NTNU

Norwegian University of Science and Technology

Towed ROV

BACHELOR THESIS

Håkon Inge Longva Haram (IIR: 10032)
Bjørnar Magnus Tennfjord (IIR: 10055)
Robin Stamnes Thorholm (IIR: 10051)

Total number of pages: 130/417

Delivered: 20.05.2019, Ålesund

IE303612 Department of ICT and Natural Sciences
Norwegian University of Science and Technology

Supervisors: Ottar Laurits Osen, Paul Steffen Kleppe

Obligatorisk egenerklæring/gruppeerklæring

Den enkelte student er selv ansvarlig for å sette seg inn i hva som er lovlige hjelpemidler, retningslinjer for bruk av disse og regler om kildebruk. Erklæringen skal bevisstgjøre studentene på deres ansvar og hvilke konsekvenser fusk kan medføre. Manglende erklæring fritar ikke studentene fra sitt ansvar.

Du/dere fyller ut erklæringen ved å klikke i ruten til høyre for den enkelte del 1-6:		
1.	**Jeg/vi erklærer herved at min/vår besvarelse er mitt/vårt eget arbeid, og at jeg/vi ikke har brukt andre kilder eller har mottatt annen hjelp enn det som er nevnt i besvarelsen.**	☒
2.	**Jeg/vi erklærer videre at denne besvarelsen:** • ikke har vært brukt til annen eksamen ved annen avdeling/universitet/høgskole innenlands eller utenlands. • ikke refererer til andres arbeid uten at det er oppgitt. • ikke refererer til eget tidligere arbeid uten at det er oppgitt. • har alle referansene oppgitt i litteraturlisten. • ikke er en kopi, duplikat eller avskrift av andres arbeid eller besvarelse.	☒
3.	**Jeg/vi er kjent med at brudd på ovennevnte er å betrakte som fusk og kan medføre annullering av eksamen og utestengelse fra universiteter og høgskoler i Norge, jf. Universitets- og høgskoleloven §§4-7 og 4-8 og Forskrift om eksamen §§14 og 15.**	☒
4.	**Jeg/vi er kjent med at alle innleverte oppgaver kan bli plagiatkontrollert i Ephorus, se Retningslinjer for elektronisk innlevering og publisering av studiepoenggivende studentoppgaver**	☒
5.	**Jeg/vi er kjent med at høgskolen vil behandle alle saker hvor det forligger mistanke om fusk etter høgskolens studieforskrift §31**	☒
6.	**Jeg/vi har satt oss inn i regler og retningslinjer i bruk av kilder og referanser på biblioteket sine nettsider**	☒

Publiseringsavtale

Studiepoeng: 20

Veileder: Ottar Laurits Osen, Paul Steffen Kleppe

Fullmakt til elektronisk publisering av oppgaven

Forfatter(ne) har opphavsrett til oppgaven. Det betyr blant annet enerett til å gjøre verket tilgjengelig for allmennheten (Åndsverkloven §2).
Alle oppgaver som fyller kriteriene vil bli registrert og publisert i Brage HiM med forfatter(ne)s godkjennelse.
Oppgaver som er unntatt offentlighet eller båndlagt vil ikke bli publisert.

Jeg/vi gir herved NTNU i Ålesund en vederlagsfri rett til å gjøre oppgaven tilgjengelig for elektronisk publisering:	☒**ja**	☐**nei**
Er oppgaven båndlagt (konfidensiell)? (Båndleggingsavtale må fylles ut) - Hvis ja: **Kan oppgaven publiseres når båndleggingsperioden er over?**	☐**ja** ☐**ja**	☒**nei** ☐**nei**
Er oppgaven unntatt offentlighet? (inneholder taushetsbelagt informasjon. Jfr. Offl. §13/Fvl. §13)	☐**ja**	☒**nei**

Dato: 20.05.2019

ABSTRACT

This report is written on the basis of the bachelor thesis "Towed ROV" and is provided by NTNU Ålesund. The purpose of this thesis, is to study the concept of a towed ROV, which includes designing and building of a prototype which shall be towed behind a surface vessel. The ROV system is designed to perform as a multipurpose platform, with the possibility to perform explorations and searches. The ROV also makes it possible to survey large areas in a short amount of time, and removes the continuous need for the attention of an operator. With the results of this project, the group was able to control the ROV through a lucidly graphical user interface, stream live video, and should be able to automatically control the depth of the ROV. The design of the ROV is designed to be modular, which the group believe they have achieved; it shows that it is possible to develop a towed ROV that uses affordable, commonly available materials.

The project is based on the previous Towed ROV thesis from 2018. To further improve the old concept, it has been developed a new ROV that has a longer design, improved software and communication. On the surface vessel it was created a system for quick and easy connection, this significantly reduces the time used to set up the system.

Preface

This bachelor is written by three automation students at NTNU Ålesund. The project was carried out during the spring semester of 2019. The purpose of the project was to improve an older design of a towed ROV, which also was created here at NTNU Ålesund. The project contains tasks such as design, software development, creating a functional prototype and a test of the prototype.

We would like to thank the following people:

- Our supervisors Ottar Laurits Osen and Paul Steffen Kleppe for support and input.
- Our previous lecturers and professors.
- The lab engineers Anders Sætersmoen and Øyvind Andre Hanken for assistance.
- Staff engineer Andrè Tranvåg and his lab apprentices for assistance.
- Lastly, to everyone who has contributed with our project.

X 

Håkon Inge Longva Haram

X 

Bjørnar Magnus Tennfjord

X 

Robin Stamnes Thorholm

Ålesund, 20.05.2019

Contents

Terminology

Concepts

Char Primitive data type for characters

Gnd Ground in electronic circuits

IP address Internet Protocol address

Optocoupler A device used to optically isolate circuits

Real-time Real-time programs shall guarantee a response within a pre-specified time constraint

String A class that represents a string of text

Bar A metric unit of pressure

Units

A Ampere

Ah Ampere hour

Hz The derived unit of frequency

Kg System International unit for Kilogram

Knots Nautical unit for speed (1 knot ... = 0.5144 m/s)

Nm Newton Meters

Notation

K_p Proportional term of a PID controller

K_i Integral term of a PID controller

K_d Derivative term of a PID controller

Abbreviations

API Application Programming Interface

ASCII American Standard Code for Information Interchange

CSV Comma-seperated values

DC Direct Current

DOF Degrees of Freedom

FEM Finite element method

FTP File Transfer Protocol

GPIO General-purpose input/output

GPS Global Positioning System

GPX GPS Exchange Format

GUI Graphical User Interface

I^2C Inter-integrated Circuit

IC Integrated Circuit

IDE Integrated Development Environment

IEEE Institute of Electrical and Electronic Engineers

IMU Inertial measurement unit

I/O Input Output

KML Keyhole Markup Language

NACA National Advisory Committee for Aeronautics

NMEA National Marine Electronics Association

OS Operating System

PCB Printed Circuit Board

PID Proportional Integral Derivative controller

PLC Power-Line Communication

ROV Remote Operated Vehicle

RPi Raspberry Pi

RTC Real Time Clock

TCP Transmission Control Protocol

TP Twisted Pair

UDP User Datagram Protocol

USB Universal Serial Bus

List of Figures

List of Tables

Chapter 1

Introduction

The purpose of this Towed ROV project was to develop a prototype that can perform various subsea tasks. This includes tasks such as exploration, surveys, mapping of the ocean floor, tasks within search and rescue, etcetera. The ROV will be towed behind a surface vessel, which removes the need for self-propulsion on the ROV. Its characteristics are automatic depth control, either from the surface to target depth, or from the ROV to the sea floor, manual control, and a user-friendly GUI which displays a live video stream. It also has the functionality to take high resolution images which can be used to map the sea floor. This thesis is based on the previous Towed ROV thesis from 2018, and the main focus will be to improve the ROV. However, since it was not possible reuse the ROV from the previous project, we ended up with building a new ROV from scratch. This includes designing the ROV and its components, a plug-and-play suitcase to simplify the operation setup, development of software and communication, while keeping the cost at a minimum.

Figure 1.1: The design of the ROV

1.1 Topic

The topics to be addressed

- Study the concept of a towed ROV.
- Develop a functioning prototype with software which includes a control system to further study the concept.

1.2 Objectives

The objectives of this bachelor thesis are:

1. Develop a prototype with focus on simple assembly (plug-and-play), good hydrodynamics, and a modular design.
2. Implement automatic depth control, live video stream, and fast and reliable data transfer.
3. User-friendly and functional graphical user interface.
4. Maximum operating depth of 50 meters

1.3 Project specification

In the preliminary report, a table of specifications were made. These specifications are what the project must have, should have and what it could have. The table is shown below:

Must have	Should have	Could have
Functioning and user-friendly GUI	Anti-collision system that can detect sandbanks/hills	FPV-goggles to watch live feed
A control system that can keep the ROV at level, dive, and rise.	Sensor and signaling rod on ship	Heater to prevent moisture and condensation in camera house
Live Video Stream	Manual Control of ROV	Auto adjust external lights based on video feed
Gimbal on camera with manual controls	Calculate approximate GPS position of ROV	
An electrical housing that is not filled with oil	Alarm system (detect leak, moisture, etcetera)	
Live PID calibration in GUI	A plug and play system on all to ease the use of the ROV	

Table 1.1: Must-, should-, and could-have list

1.4 Approach

During the scope of the thesis, the project was divided into several phases. The first phase involved planning the approach for solving the tasks. This included a preliminary report, where a detailed Gantt diagram was created (C). The Gannt diagram worked as a project plan, where each of the members have been given a set of tasks and deadlines.

The second phase was taking the old ROV apart and salvage components that could be reused. This phase also included researching other solutions, gathering data, and ordering of necessary parts. During this phase, different designs were considered and studied.

The third phase was designing and building of the ROV, support vessel equipment, and code layout.

The final phase was testing of communication, full test at sea, and finally finish writing the thesis.

1.5 Structure of the Report

The rest of the report is structured as follows.

Chapter 2 - Theoretical basis: Chapter two gives an introduction to the theoretical background needed for this project.

Chapter 3 - Method: Contains a description of the methodology and materials that were considered throughout the project.

Chapter 4 - Result: Contains a description of the finished prototype, the software developed for the project and the results of the tests completed.

Chapter 5 - Discussion: A discussion of achieved results, changes done, and areas for improvement.

Chapter 6 - Conclusions: This chapter presents an overall conclusion of the project and experiences gathered throughout the course of this thesis.

Chapter 2

Theoretical basis

The theory necessary for completing this project is presented in this chapter. Multiple areas are presented throughout the chapter, such as material, hardware, software, communication, and calculations.

2.1 Remotely Operated Vehicle

A Remotely operated vehicle is an unmanned mobile device utilized below the surface of the ocean. The ROV is often controlled from a ship or a platform. These types of vehicles are used in many different types of operations such as service and repair on subsea installations, inspections, and detailed mapping of the seafloor, to name a few. The normal operating depth range of an ROV is between 0 to 3000 meters. There are multiple types of ROVs, such as towed ROVs and self-propelled ROVs. Common for all types of ROV's is a cable connection from the ROV to the operating station onboard the surface vessel [5].

2.2 Software

The software section describes the different development theories used for developing software and concepts that had to be taken into concern.

2.2.1 Executor and Scheduler

The executor is an interface that provided a way of decoupling task submissions. The executor is used for managing threads within an application. It forms the basis of a flexible and powerful framework for asynchronous task execution. The executor is based on the producer-consumer principle, where the producer submits the tasks and the consumers execute the tasks.

Whenever a thread is in use, one should consider replacing the standard single thread operation with an executor. This way multi-threaded programs become more durable, as the executor offers protection for unforeseen events that make threads stop. The executor restarts the thread if this happens. If there is resource exhaustion, the executor then limits the number of active threads. It also enables to set the priority of the threads, which gives the threads run-time[17]. These threads managed by the executor are thread-safe as long as the methods of the tasks are implemented correctly. With the scheduler, the tasks can be scheduled to run one time or periodically.

2.2.2 Communication

In our project, the ROV system is built on a GUI, Raspberry Pis, and several Arduinos. These components need to communicate with each other to send and receive essential data. It is therefore necessary to use different communication protocols.

UDP

UDP (User Datagram Protocol) is a network protocol for transferring data. UDP works in a way where it has no guarantee that a package has been delivered, and there is no state that ensures the package to be stored with the sender after the package has been sent. The reason for this is that it is not necessary to fill the sender port before reading. The receiver does not need to know where the package is coming from, since it is not expected a response in return of a received package [18].

TCP

TCP (Transmission control protocol) is a more complex protocol than the UDP protocol. The TCP is more reliable for data transfer than UDP since the receiver needs to send a response to the sender if a package has reached the destination. If not, the TCP will retry sending the missing data. If the data still remains undelivered, the receiver is notified of this failure [4].

Port

For a package to be sent over a network to the correct destination, you need a receiver address which need to contain the name of the server/computer and a port number. If a computer is running multiple processes simultaneously and communicates through the same communication medium, then the package needs to know where it is supposed to go after it has arrived the receiver. This means that the package needs to know what program to connect to. The port-numbers task is to define this. Normally, port numbers between 1-1023 are reserved to system applications, but above this, you can freely choose what port number to use up to 65535 [24].

SSH

The Secure Shell protocol (SSH) was developed to provide secure communications over an unsecured network. It is based on a client-server principle, and in order to get access via SSH, a password and username are required. The main advantage of using SSH is the ability to remotely log in to a computer system using an Ethernet connection [22].

I^2C

I^2C (Inter-integrated Circuit) is a bus system invented by Philips in 1982. I^2C is mostly used for integrated circuits, processors, and microcontrollers in short distances. The I^2C bus has a multi-master and multi-slave system, meaning that one or more devices can be masters, and one or more devices can be slaves. The communication between the devices uses two wires called SCL (synchronized clock) and SDA (synchronized data). I^2C use SCL to synchronize time against each device on the bus where the master devices clock is the reference. SDA is for transferring data. I^2C supports up to 1008 devices as slaves. Compared to other bus systems, I^2C has a relatively slow bit transfer range, ranging from 100 kHz to 5MHz. I^2C is also restricted to shorter distances for communication.

The master is the one that needs to decide which slave it wants to send data or request data from. The master also decides how much data will be sent, and the data is sent in the form of 8-bit frames. The clock speed is determined by the speed of the master. This might lead to an issue with slave units, as the speed can be too high for the slave. When this happens, the slave device tries to slow down the clock speed, and this is a term called clock stretching. When this

happens, the slave reduces the bus speed in order for it to send or receive data. The result of clock stretching with slave devices is that it can significantly reduce the overall bandwidth of the bus [12].

Power-Line Communication

Power-Line communication (PLC) is a technology used for sending data over a power cable. PLC use the well established IEEE-1901 standard for broadband over power lines. This standard is theoretically able to transfer up to 500 Mbit/s. This means that with just power cables running to a device, we can both power it up and also send and read data. PLC can be both sent via AC and DC [6].

NMEA0183

The NMEA0183 is the name of the standard developed by the National Maritime Electronics Association. This is used for interfacing marine electronic devices and has become the standard interface for GPS receivers whether they are used at sea, land or in the air. NMEA 0183 data transmission use plain text with the characters coded using seven-bit ASCII (American Standard Code for Information Interchange). To make full eight-bit data bytes, an additional bit is added to each seven-bit character and set to binary 0. The NMEA 0183 data are transmitted in asynchronous serial form, meaning that bits are sent one at a time. The data is transmitted at a rate of 4800 bits per second. NMEA recommends the use of an optocoupler between talkers and receivers for noise immunity. The data that is sent is in the form of NMEA sentences with a maximum length of 82 characters [16].

2.3 Control system

This section presents the theory behind developing, tuning, and controlling the control system for the ROV.

2.3.1 PID

A PID controller is a control loop feedback mechanism widely used in industrial control systems. The PID controller calculates an error value as the difference between the desired set-point and a measured process variable, and then it applies a correction based on proportional (K_p), integral (K_i) and derivative (K_d) terms. This means that the PID automatically applies an accurate and responsive correction to a control function. The controller attempts to minimize the error over time by adjustment of a control variable, to a new value determined by a weighted sum of the control terms [14].

Ziegler-Nichols' closed-loop method

The Ziegler-Nichols' closed-loop method is a method for deciding the parameters in a PID controller through experiments, either by a simulator or in the physical system. The method is based on the requirements that the system should be as fast as possible and at the same time, keep acceptable stability in the system. Ziegler and Nichols defined acceptable stability as the amplitude of the oscillations in the system response after a jump in the set point decreases by a factor of four, which means that the amplitude of the second peak in the response should be about $1/4$ of the amplitude of the first peak. One of the weaknesses of using the Ziegler-Nichols method is that it causes overshooting in addition to dampen the fluctuations in the response [1].

2.4 Calculations

All necessary calculations needed for the project are presented in this section.

2.4.1 Buoyancy

The buoyancy is the upward force applied to an object submerged in a fluid. The equation is as follows:

$$B = V \times \rho_f \times g \tag{2.1}$$

Where:
B = buayant force in N

ρ_f = Fluid density (1029 kg/m^3)
V = displaced body volume of liquid in kg/m^3
g = gravity (9.81 m/s)

2.4.2 Resistance in cable

$$R = \rho \times \frac{L \times 2}{A} \tag{2.2}$$

Where:
R = resistance in cable
ρ = Material resistivity
L = Lenght of cable
A = Cable cross section

2.4.3 Voltage drop in a cable

$$\Delta U = R \times I \tag{2.3}$$

Where:
ΔU = Voltage drop
R = Resistance in cable
I = Current draw

2.4.4 Transformers turnover ratio

The formula for an ideal transformer with no transmission loss

$$n = \frac{N_P}{N_S} = \frac{U_P}{U_S} = \frac{I_S}{I_P} \tag{2.4}$$

Where:
n = Turnover ratio
N_P = Windings primary side
N_S = Windings secondary side

U_P = Voltage primary side
U_S = Voltage secondary side
I_P = Current primary side
I_S = Current secondary side

2.4.5 Battery capacity

$$H = \frac{Ah}{I} \tag{2.5}$$

Where:
H = Hours
Ah = Ampere hours
I = Current draw

2.4.6 Nyquist-Shannon sampling theorem

$$Fs = 2 \times f_h \tag{2.6}$$

Where:
Fs = Sampling frequency
f_h = Highest frequency of the signal.

Chapter 3

Materials and methods

In this section a detailed description of concepts considered, what was decided along the way, and how it was implemented to develop a functional prototype.

3.1 Project organization

The group consists of three students with different areas of responsibilities within the project. The project leader's responsibilities are to ensure that tasks are completed within the deadline, and delegate tasks. The secretary's responsibilities are writing summaries of the group meetings with the supervisors, and writing the weekly reports. The final member is the project engineer, which does not have any specific responsibilities, other than completing the tasks according to the project plan. The project is planned after the Lean development principle, which means that every person is expected to work independently even though there is a project leader as described in the preliminary report (B). Decision making will be done by the group, and each members opinions are weighed equally.

Group meetings will be every Monday at 9:00 AM. Here the group will discuss problems encountered, how the progress is going, and the plan forward. Meetings with supervisors are held every other week. In these meetings, the progress, problems, and solutions are discussed.

During the preliminary report phase, a project plan was created. This is represented with a Gantt diagram. The diagram is used as a schedule for what needs to be done throughout the project phase. The Gantt diagram has multiple tasks with deadlines, and if any of the group members cannot finish the tasks within the deadline, the group must be informed of this. This way, a decision can be made for solving the tasks within deadline.

3.2 ROV terminology

In this section, we are going through the different terminologies used to describe various parts of the ROV.

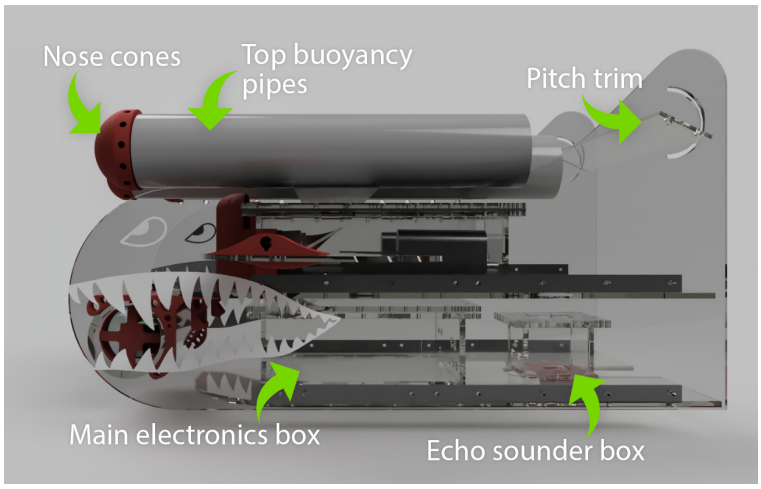


Figure 3.1: The port side of the ROV

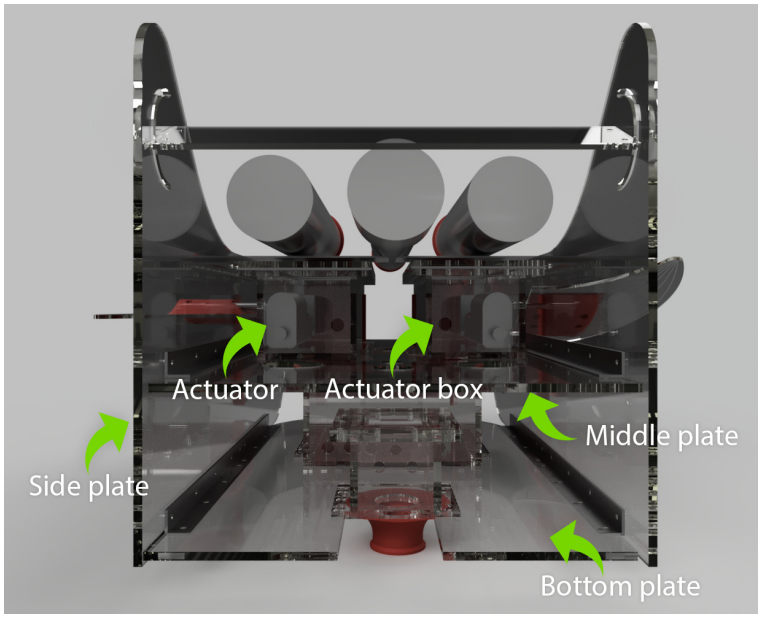


Figure 3.2: The stern of the ROV

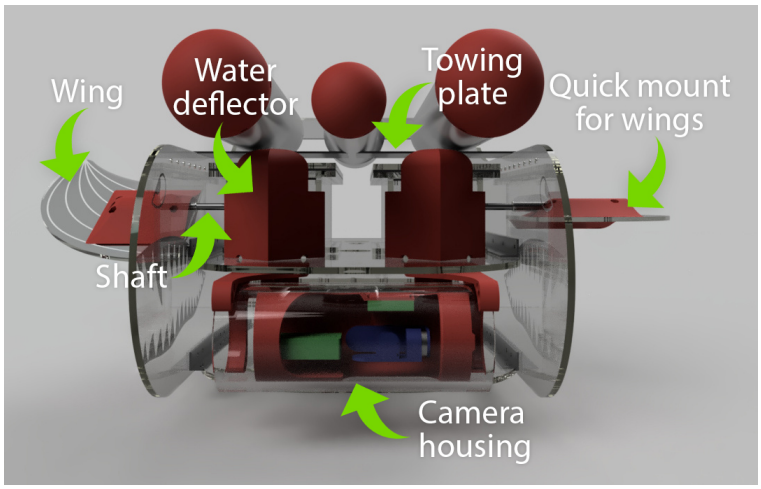


Figure 3.3: The front of the ROV

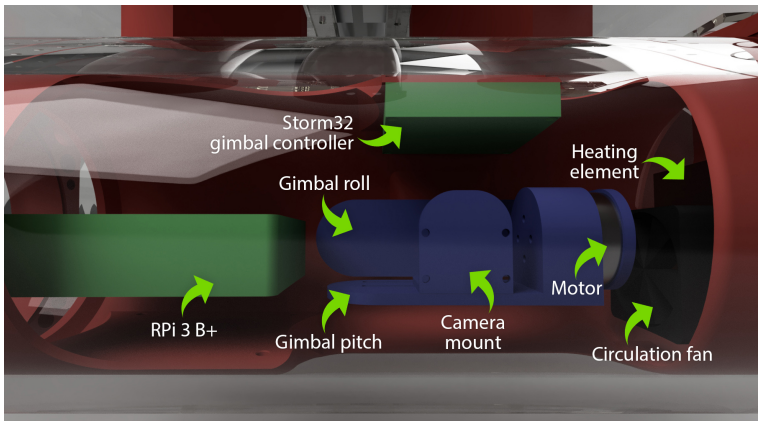


Figure 3.4: Camera housing

3.3 Concepts

3.3.1 Frame

When designing the frame for the ROV, several factors need to be considered, such as drag of the ROV, simple production and assembly, modularity, low cost, and weight.

What types of operations the ROV will be doing is a crucial factor for the frame design. There are different types of ROVs, such as work class ROVs, and ROVs designed for more streamlined operations. The working class ROVs are usually used for subsea operations at low speed. There

is often a lot of equipment and sensors on them, where hydrodynamic design is not so important. The streamlined ROVs are mostly used for operations at higher speeds. These often have few or none additional equipment connected to them. They are mostly used for operations for capturing video. For this type of project, a combination of both types would be the optimal choice.

3.3.2 Material selection

The first things that have to be determined, is the materials to be used for the different parts. Most common materials for ROVs are plastics and metals since these have proven to be well suited for underwater operations.

Aluminum and titanium are the most considered material for these types of operations. The aluminum has a high strength-to-weight ratio and a low price. With Titanium there is no corrosion on the material when in contact with saltwater, but the price is much higher. Other commonly used materials are stainless steel, brass, and bronze. Bronze and brass are usually used for plumbing and fittings for screws and bolts, while stainless steel is used in hydraulics, pneumatic and electrical fittings as well as fasteners.

Most commonly used plastic materials in subsea operations are PVC, ABS, and acrylic. PVC and ABS are mostly used when the ROV will operate in shallow waters, as both materials are very fragile. Acrylic is commonly used as viewports for cameras because it can be completely transparent. Acrylic has a good strength-to-weight ratio, but the drawback is that it is quite brittle.

3.3.3 Linear actuator

Linear electric actuators cast a motion in a straight line, instead of circular motions like the conventional electric motor. These actuators are usually used in machine tools and industrial machinery. They typically operate by converting rotary motion into linear motion. There are also other types of linear actuators, which are using hydraulics or pneumatics instead of an electric motor. In this particular project, a linear electrical actuator has been used [23].

Figure 3.5: GLA750-P 12V DC Linear Actuator with Position Feedback [20]

3.4 Design

The original plan with the project was to keep the ROV as it was designed in the previous project and only add components to it. We knew early on that the ROV was leaking oil and water had penetrated the 3D printed boxes. Based on this, we started to dismantle the ROV to find out what state it was in and get an overview of what was needed to be fixed.

When we started the disassembly, we quickly discovered that the screws that were holding the 3D printed boxes and the lids had sunken into the 3D print. This had happened because the boxes were printed with PLA, and the petroleum-based motor oil they had used to fill the boxes with altered, the characteristics of the PLA, making it soft. This resulted in an ROV that was impossible to open without destroying it. We therefore, decided to only harvest the electrical components inside of it. Here we managed to salvage the two actuators, some Arduinos and a Raspberry Pi, but some components were coated with epoxy cement which were unusable.

Because of the state the ROV, we decided to start from scratch. We looked at what alternatives we had and what the previous group had been suggesting for improvement. We reduced the ideas to three realistic designs based on price, availability, and quality.

Figure 3.6: A bullet design.

Figure 3.7: A flat wing design. [11]

Figure 3.8: Old ROV design.

The bullet design

The bullet design was very early a concept we believed would fix many of the problems from the old ROV. We planned to get an empty industrial propane tank, around 17-20 kg bottle and use this as the ROV body, and fix it up so it was waterproof. With a cylindrical shape in metal, it would be able to withstand the underwater pressure without filling it with oil. Not filling it with oil would make maintenance and further improvements much more manageable, along with reducing the overall weight of the ROV. We would also limit the number of cable glands needed, thus reducing the risk of leakage, see figure (3.6).

Pros:

- Not filled with oil
- Body can be made from tanks that are not usable for propane anymore
- Can withstand high pressure
- Few critical points where water can enter
- Very hydrodynamic

Cons:

- Hard to make the shaft seal for the wings waterproof
- If a small leak occurs it will be filled with seawater quick and destroy the electronics
- The camera dome will be a challenge and probably expensive

The flat wing design

The flat wing design is a design that combines the bullet design and the old design. It would be hydrodynamic and easy to control due to its similar characteristics to an airplane. It would also be possible to build the most of it out of PVC pipes, and it would be quite modular, since we could make a sliding system for the electronics boxes that could be placed in the PVC pipes, see figure (3.7).

Pros:

- Hydrodynamic with airplane-like characteristics
- Body can be made out of PVC pipes

Cons:

- Complete redesign of the control methods of the wings
- To get the desired body a lot of 3D printing would be needed
- The design would be so different that the experience the last group had gathered would not be transferable

The old design

The old design was a design that was already tested. We could take inspiration from the old design and probably reuse some of the parts. A huge time saver would be that we could use the exact same mechanically construction to steer the wing. The main advantage of this design is that it is very modular. If a component in one of the boxes failed, we would only need to open that box instead of the whole ROV, see figure (3.8).

Pros:

- A tested system
- 3D files for the design should exist
- We could reuse some of the parts
- The entire actuator wing system could be reused

Cons:

- Only slightly better hydrodynamics
- Metal part has to be custom made and welded
- A lot of exposed entry points for water

3.5 Hardware

This section contains a description of the hardware used in the project, and what they are used for.

Raspberry Pi 3 model B+

The Raspberry Pi (RPi) single board computer is used as a main computer within the main electronics box for running the main Java program, controlling the non-latching relays through its GPIO pins, the actuators through its I^2C pins, as well as reading serial data through its USB ports. In the camera housing there is also a RPi which is used for broadcasting the video stream and controlling various sensors through its I^2C and GPIO pins. The operating system used is Rasbian Stretch [9].

Figure 3.9: Raspberry Pi model 3B+.

Raspberry Pi camera module V2

The RPi camera module V2 is an official product by the Rasberry Pi Foundation, designed for the Raspberry Pi. This camera has 8 megapixels and can take video in 1080p at 30fps, 720p at 60fps. The sensor resolution is 3280 x 2464 pixels. The connector is specially made for the RPi, and there is no need for a USB connection. Used for the live video stream and high-resolution photos [8].

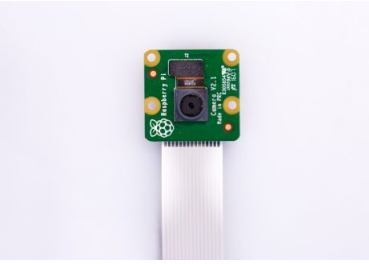


Figure 3.10: Raspberry Pi Camera V2.

Razor IMU 9DoF SEN-14001

In the Towed ROV project, a SparkFun 9DoF Razor IMU has been used to read the roll and pitch of the ROV. This IMU has a total of 9 degrees of freedom, where the accelerometer, gyroscope, and magnetometer has 3 degrees of freedom each.

Humidity and Temperature Sensor - HIH6130

The SparkFun Humidity and Temperature Sensor is based on Honeywell HumidIcon's HIH6130. The HIH6130 is a combined temperature and humidity sensor capable of communicating over I^2C. The temperature range is -40° to 85° and the relative humidity range is 10% to 90% [21].

MS5837-30BA pressure sensor

The MS5837-30BA pressure sensor is a high resolution pressure sensor, with a I^2C interface. The sensor measure both pressure for depth and temperature. The main function of the MS5837-30BA is to convert the uncompressed analogue output voltage from the pressure sensor to a 24-bit digital value, as well as the 24-bit digital value for the temperature sensor. This sensor is used on the camera housing, and is placed on the lid. The sensors measures the temperature of the water, and measures the pressure to determine the depth of the ROV [3]. The pressure sensor showed reliable and accurate data when we tested the ROV in the ocean. See figure (4.4).

SparkFun RedBoard

Sparkfun RedBoard is a microcontroller based on Arduino UNO. This microcontroller can be programmed with a USB Mini-B cable and powered through the same cable. The circuit board

has in total 14 digital I/O, six PWM pins and six analog inputs.

Used for reading sensor values from GPS and echo sounder in the suitcase.

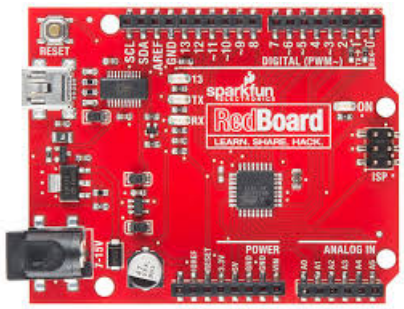


Figure 3.11: SparkFun RedBoard.

Arduino UNO

Arduino UNO is a microcontroller with 14 digital I/O. It is capable of reading up to six analog inputs and. We chose this because it can be programmed with a USB Mini-B cable with power supply through the same cable. For our project, it does not differ anything from the SparkFun RedBoard. The Arduino UNO is used to handle the feedback from the actuators and is also handling all the extra I/O connections.

Arduino Nano

The Arduino Nano board is a compact microcontroller similar to the Uno. The Nano is based on ATmega328, with a total of 22 I/O pins where 6 are PVM and 8 Analog in. Used in the ROV to read sensor values from the echo sounder.

Figure 3.12: Arduino Nano.

Echo Sounder

The echo sounder used in the project is an Airmar DST800. This echo sounder is available with NMEA 0183 and NMEA 2000 versions. This sensor can measure depth, speed, and temperature all in one unit, with a maximum depth range of 70 meters or 100 meters depending on what version is used. The echo sounder used in this project is the NMEA0183 version, since this was salvaged from the previous project.

Fathom-x Tether Interface

The Fathom-X tether interface is a Power-Line Communication module. The interface provides a high speed, long-distance Ethernet connection to an ROV or other remote platforms. It is designed to transfer data at 80 Mbps with 300m+ tether length capability. Used for transferring data between ROV and surface vessel.

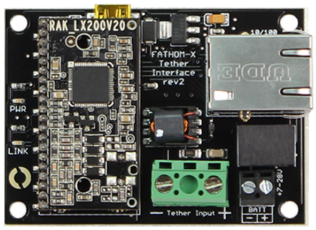


Figure 3.13: Fathom X-3 tether.

Umbilical tether ROV cable

The ROV cable is a neutral buoyant copper cable with 14 conductors used for transferring data and to power the ROV. The cable is waterproof and reinforced with Kevlar which enables it to be used to tow the ROV.

Network cable Nexans LANmark cat.6A S/FTP

This network cable is designed to be outdoors and is shielded. We chose this cable since the cable was stiff and would, to some degree, not be compressed by the water pressure. This is important, so the cable does not shrink in the cable gland and causes leaks. The internal pairs are also individually shielded, which we could use for the I^2C conductors and other signals that need to travel between the boxes.

Rubber cable Nexans LINEAX

The rubber cable is used to connect the battery pack to the suitcase. The cable is chemically tolerant to oil, can withstand high physical forces and is very flexible.

Instrumentation cable TCX (I), Nexans

The instrumentation cable is used for powering the actuator boxes, echo sounder, and camera housing. The cable consists of two $0.75mm^2$ pairs. We can use this to send 12V and 5V through the same cable.

3.6 Communication protocols

3.6.1 Serial communication

Serial communication is used for communication from the surface computer to multiple Arduino devices in the suitcase, and from the Arduinos for actuator feedback and extra I/O to the main RPi in the ROV. Serial communication can be transmitted with a USB cable. This way, we could use a USB cable on each of the Arduino directly to the computer. To prevent connecting multiple cables into the computer, a USB hub has been used to connect all the devices.

NMEA 0183

The NMEA 0183 protocol will be used for reading data from the GPS and echo sounders. These devices had to be connected to different Arduinos, since both the GPS and echo sounder are

talkers (2.2.2). It had to be done this way because only one talker can send sentences to multiple listeners; data corruption could occur if multiple talkers are connected to the same Arduino. After the sentence has reached the Arduino, the Arduino will convert the NMEA sentence to a string and send it to the Java application (GUI).

The NMEA sentences used in the project:

- SDDPT - Depth of water
- SDDBT - Depth below transducer
- VWVHW - Water speed and heading
- YXMTW - Mean temperature of the water
- GPGGA - Global positioning system fix data

3.6.2 Ethernet communication

Ethernet communication will be used for communication between the ROV and surface vessel. Power-Line Communication is a simple method for Ethernet over two wires. The Fathom tether interface is the module used in this project for communicating over PLC.

Implementation of this is done with two circuit boards, where two wires of the utility cable are used for communication on the interfaces. A computer is then connected to the RJ45 socket. The signal voltage on these cards can be from 7-28V.

3.7 Programs and libraries

3.7.1 Programs

Github	Github is a free open source program which is used for version control and update versions in projects. This is used so multiple people can cooperate on the same project simultaneously.
SourceTree	SourceTree is a graphical user interface for simplifying version control. This program shows all branches in the git repository and how they are connected. It also makes it simpler to switch between different versions of the code.
Netbeans 8.2	Netbeans is a development platform that is based on Java. It allows applications to be developed by a set of modules. It also has a built-in graphical designer for developing and generating code of graphical user interfaces in Swing.
Arduino IDE	This is a Java-based programming tool, for programming and uploading of code to Arduino microcontrollers. The IDE runs on Windows, MAC OS, and Linux. The programming language used for Arduino is Arduino C.
VNC	VNC is a remote access software for desktop and mobile. This software is used to access the remote computers via WiFi or Ethernet.
Siemens NX	Siemens NX is an advanced CAD-program used for designing, simulating, and analyzing of 3D models.
Autodesk Fusion 360	Fusion 360 is a simple CAD-program used for designing, simulating, and analyzing of 3D models, where smaller changes can be made quickly.
Cura	Cura prepares the 3D models for printing. It is custom for the Ultimaker 3D printers and has many different settings which we can adjust to achieve the best result for the 3D model.
Overleaf	Overleaf is an online editor for LaTeX. This editor makes it possible for collaboration in real time.
Google Earth Pro	Google Earth is a computer program that renders a 3D representation of Earth based primarily on satellite imagery [26].
Garmin Virb	Garmin Virb combines video footage with GPS and other data from any camera or a compatible Garmin device [10].

Creately	Creately is an online flow diagram builder.
EasyEDA	EasyEDA is an online tool for designing PCB.
Geany	Geany was used to code in phyton.
Matlab	Matlab is an advanced computer program capable to handle huge data files and set them in system and visualize the data.
Gource	Gource is a tool for visualizing a Git repository. The program shows how each user has worked on a repository and how the code increases over time.
CorelDraw	CorelDraw is a vector based drawing software designed for editing two-dimensional images [25].

Table 3.1: Programs utilized

3.7.2 Libraries

Libraries used for the Java code:

- Pi4J
- jSSC - Java Simple Serial Connector
- Commons IO
- Commons Net
- JFreeChart
- JCodec
- MiniPID

Pi4J enables full access to the Raspberry Pi's I/O capabilities, including the GPIO pins and I^2C. The jSSC library is a simple cross-platform serial port communication library. This library makes it easier to get the serial communication working with Java. The Commons IO library is used for easy file handeling in windows. The Commons Net library enables FTP server/client functionality. JFreechart is used for plotting data into a graph inside the GUI. The AWTSequenceEncoder

class in JCodec enables us to encode images into a video file. MiniPID makes it easy for implementing a PID controller in our project.

Libraries used for the Arduino code:

- Wire.h
- Adafruit_GPS.h
- NMEA.h
- Razor AHRS v1.5.7

The Wire.h library enables communication through I^2C for Arduino devices. The Adafruit GPS library extracts NMEA format information from the GPS unit. NMEA.h is used for parsing NMEA sentences. The Razor AHRS code is used to read the sensor values from the Razor IMU.

Libraries used for the Python code:

- MS5837
- RPi.GPIO
- SMBus
- PiCamera
- OpenCV

The MS5837 library is used to interface with the MS5837-30BA waterproof pressure and temperature sensor from Blue Robotics through I^2C. It also converts the incoming data to human readable data. The RPi.GPIO library enables the possibility to control the GPIO pins on a Raspberry Pi controller. SMBus is used for communicating through I^2C in Python. PiCamera provides a simple Python interface to the Raspberry Pi camera module. The Open CV library for Python is used for handling the images from the PiCamera interface.

3.8 Software

The software developed for this project are written in Java, Python and Arduino C. These languages have been core subjects during our engineering education, which has given the group

a good amount of experience prior to this project. There are two main Java applications; one for the ROV and one for the graphical user interface. These were programmed in Java since it supports well-known mechanisms for real-time programming. Since Python has the necessary libraries for controlling the Raspberry Pi camera, Python has been used for controlling the camera and sensors connected to the camera housing. Arduino C will be used to program the microcontrollers.

The software also needs to be multithreaded. This is because of the different tasks that work simultaneously, which utilizes the CPU more efficiently. With the use of multiple threads, the program can work with their tasks without waiting for other tasks to be completed, and also without being an obstacle for other threads.

At the beginning of the project the group discussed how to develop the software. It was either to develop a completely new software and GUI or reuse the software from the previous thesis. Factors such as time to rebuild, compared to reusing and modifying the old software, were taken into account. The GUI is based on the previous "Towed ROV" project, but it was decided to rebuild the whole ROV application completely.

The challenges the group had to solve can be divided into the following

- Fast and reliable communication between server/client (TCP/UDP)
- Read, sort, and store data from the various sensors (Serial, I^2C, NMEA)
- Visually/functionally expand and improve the graphical user interface from the previous thesis
- Rebuild the video stream from scratch to improve the frame rate
- Logging of various data
- PID controller for steering the actuators

Under the software development, we focused on low coupling and high cohesion, and divided tasks into smaller classes making sure each class has a clearly defined area of responsibility. For communication between threads, we made a shared resource class. We did not synchronize the methods in this class, because when a synchronized method is called the rest of the object's methods are inaccessible since the object is locked. Instead, we have designed our applications to compensate for this by only using the set-methods of a variable from one place at a time.

The applications will be of a client-server architecture. We have two different TCP servers; one on the main RPi, and one on the RPi in the camera housing. The TCP clients are running on the operating computer (GUI). The clients ask the servers for feedback data with a frequency of 10 Hz. A UDP server is also implemented in the GUI for receiving the video stream frames from the client on the camera RPi. In the GUI, key data will be displayed, such as the various depth data, position, pitch and roll of the ROV, and also the live video stream from the ROV. The GUI application will also send the necessary data to the ROV application whenever it changes, such as depth from the echo sounder on board the boat and various commands from the GUI.

3.8.1 Communication

Serial communication

The Arduino microcontrollers we are using is communicating over serial communication. This has multiple advantages:

- After sealing the main electronic box, we can still reprogram them through the Raspberry Pi using VNC.
- Serial communication is less affected by noise than I^2C.
- We can power the Arduino through the same USB cable.

However, from earlier projects, we have experienced that it is not guaranteed that the Raspberry Pis are receiving all data, or that the received data is corrupt. To counter this, the ReadSerialData class was made. This class will keep a defined serial port constantly open when the ROV is operating. This eliminates the need of opening and closing the port, and speeds up the reading process. By holding the port constantly open, the buffer of the port is also filled. A 50 ms delay is implemented to make sure the buffer is filled before reading, meaning there is plenty of data to take from when the buffer is read. To eliminate the possibility of reading old data, the latest complete data set that is received, is the only one that is read. A data set with just one command will look like this: <key:value>. A data set with multiple commands looks like this: <key1:value1:key2:value2:key3:value3>. To eliminate corrupt data or misspelled commands, the class checks for the following:

- Does the data set contains the symbols < and >? If they do, delete everything else and store the data in between the <> symbol.

- Is the key present in the switch case? If it does, store the value in its respective variable in the shared resource class.

By doing this, the serial communication worked flawlessly.

One important note is how often the Arduino is sending out new data versus how big the delay for building up the buffer is. Here we can take inspiration from the Nyquist theorem, which says that the sampling rate has to be at least twice the speed of the highest frequency. This would in our case mean that if the Arduino is sending out data every 10 ms, then the data should be read every 20 ms in Java. In our case, the Arduino is freewheeling. This means that the Arduino is sending data as fast as it can. Since the buffer is read every 50 ms, we can be completely sure that there is data to read. The buffer is read every 50 ms because most of our executor threads are running every 100 ms, as described in section (3.8.1).

Internal communication - Surface vessel

Communication between the components on the surface vessel is done with serial communication. The echo sounder and GPS data are originally presented as an NMEA sentence, but this is parsed in the Arduino code, where the variables for the different values are compressed into one string, which is presented as "<key:value>". The key is for example Temperature and the value is the temperature of the water. This string is then sent via serial communication over a USB cable to the computer, and the string is then parsed in the Java code. The USB hub connects all the Arduinos to the operating computer. In the GUI application, a class responsible for reading the data and control what com port is connected to which Arduino was created.

A problem encountered early in the project, was reading of sensor data from the echo sounder. The NMEA sentence worked perfectly, but the problem was with parsing it to a string and sending it to the Java application (3.8.1). From earlier projects, we have created a class in Java that reliably reads serial data from Arduino. To prevent making changes in the Java class, we instead edited the Arduino code to build the string piece by piece manually, ending up with a complete string of data that is equal to the original format "<key:value:key:value>".

Internal data update rate

The update rates in the GUI is set to 10 Hz. At this speed, the system is operating fast enough to successfully send and receive data without the user feeling the GUI is unresponsive or slow.

The update rates in the ROV is faster than the GUI. This is because the code needs to act quickly when new data is received. An example is the logic class, which is running at 200 Hz. This class is (among other tasks) responsible for stopping the actuators if they have reached their desired position. Therefore the class has to be run as often as possible or else we risk that the actuator is passing their target and the targets hysteresis (3.14.3). The reason why the code is not freewheeling is that it will use additional CPU time and delay other tasks.

Communication and data transfer

The communication between the ROV and surface vessel is done over TCP and UDP, via the utility cable towing the ROV. UDP is used for the video stream and TCP for communication and other data transfer. For the video stream, it is not crucial to receive every frame, and this is why UDP is used. Data that need to reach its destination such as depth, I^2C and leak detection, is sent over TCP. This is important data, and we could not risk that this data was lost in the transfer.

The way TCP communcation is implemented is similar to the serial communication described in section (3.8.1), where the same start and end brackets are used to make sure it has received valid data. This is a bit overkill since TCP already is a reliable protocol which provides error-free data transfer. However, since we have used this method in multiple projects earlier, we know it provides us with a method that also minimizes the chance of human error while coding. A TCPClient thread is created when the GUI connects to either the camera RPi or the ROV RPi TCP server. This thread runs every 100 ms and checks if the connection is maintained - if not, it waits for five seconds before trying to reconnect to the server. Data is transferred is by calling the *sendCommand(String command)* method. An example of a command would be *cmd_targetMode:2* or *fb_allData*. The *sendCommand()* method then adds the start and end brackets, before sending the string through a *PrintWriter*. The server then sends a response, depending on which command is received. If the server receives a set-command, a short OK response is sent back, and if the *fb_allData* command is sent, the server then builds a string of the feedback data and sends it back to the client which then splits up the string and sets the values to the corresponding variables in the shared resource class *Data* by calling the *handleDataFromRemote* method.

3.8.2 GUI

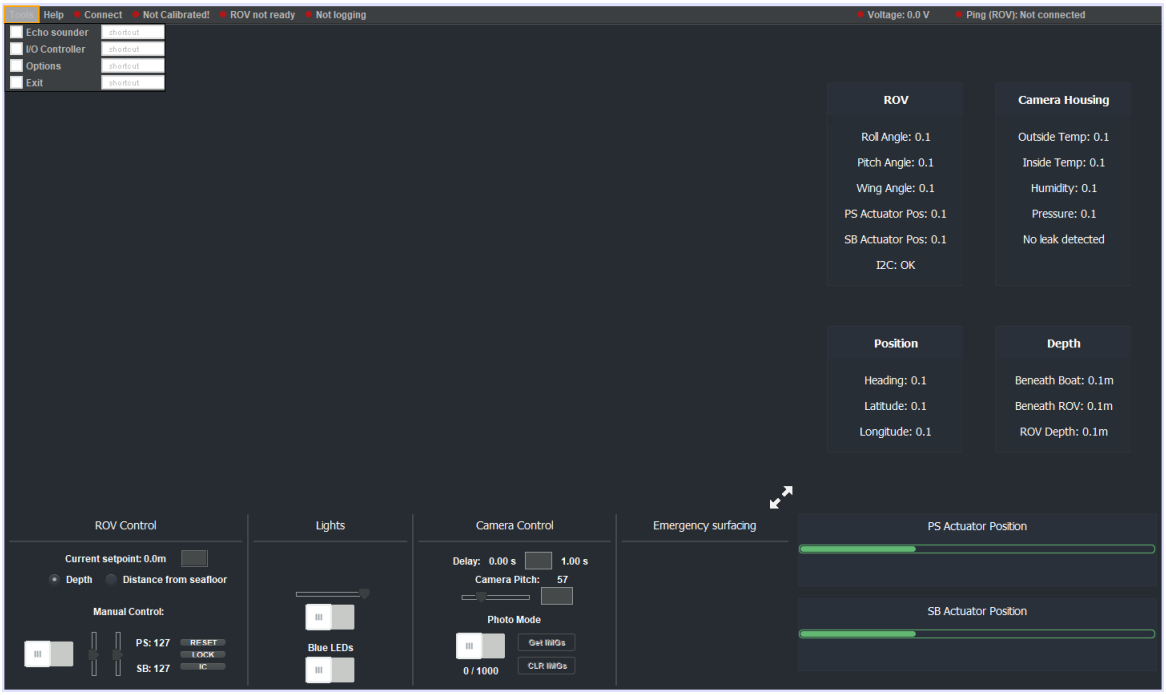


Figure 3.14: The graphical user intereface concept in the GUI builder

Figure (3.14) shows the concept of the main window of the GUI where the video stream and most of the data is displayed. For our bachelor thesis, we originally wanted to rebuild the whole GUI in JavaFX, but this was later discarded to save time. We therefore based the GUI on the one from the previous project (built in Swing) as this already had some of the functionality we wanted. In general, we have improved, both visually and functionally, the GUI to have a more modern design and implemented new functionality. In the bottom left corner, the ROV Control panel is located. From the previous thesis, the functionality for both depth mode and distance from seafloor mode was added. We have expanded this by adding an option for manual control. With this mode activated we can either control each wing separately by moving the sliders, or both wings simultaneously by switching the LOCK button on. We have also implemented an Xbox 360 controller which can be used to control the wings of the ROV wirelessly. This can be enabled by pressing the IC button. The next panel to the right is for controlling both the blue LED in the boxes and the LED lights mounted on the front of the ROV. The intensity of the front lights can be controlled by moving the slider. Next is the Camera Control panel; here it has been implemented the possibility to control the pitch of the camera's gimbal. We can change to Photo Mode (explained in section 3.8.4), choose the desired delay between each photo is taken, and

also retrieve and clear the images from the RPi SD card. The last panel holds the emergency button. If this button is pressed, the ROV will begin surfacing immediately, and a loud alarm sound will go off on the operator computer. On the right side of the GUI, the most essential data is displayed. Here it displays the depth beneath the ROV and the boat, the depth, roll, pitch and wing angle of the ROV, the speed and position of the wing actuators, the temperature inside and outside of the camera housing, relative humidity in the camera housing, the outside pressure, leak detection, I^2C status, and the position data (heading, latitude and longitude). At the top right corner, the voltage of the batteries and the ping to the ROV RPi is displayed. In the top left corner in the Tools menu, the Echo Sounder, I/O Controller and Options frames can be opened in a new window. To the right there is added menus for connecting to the TCP servers on the RPis, calibrating the actuators, display the ROV status, and start/stop data and video logging.

The update rate of the incoming data is set to 10 Hz. The GUI communicates with the Raspberry Pis in the camera housing and the ROV over TCP (explained in section 3.8.1), and the video stream over UDP (explained in section 3.8.3).

3.8.3 Video stream

The video stream is built in Python using the PiCamera library together with OpenCV. Since OpenCV is such a big library, a typical installation takes around 4-5 hours. It took several tries before the installation was completed successfully. It was also neccessary to mount an extra heat sink in addition to a small cooling fan to prevent the Raspberry Pi from crashing due to overheating under the installation process.

Each raw frame from the camera is captured with a resolution of 680x420 pixels. Each frame is then converted to BufferedImage and compressed using OpenCV before the frame is sent in a DatagramPacket. With this resolution, the video stream has a varying delay of 2-5 ms (20-50 frames per second) between each frame is received in the GUI. This is more than enough to get a smooth and stable video stream with decent quality. In the same Python program, a *UdpListener* (server) thread is implemented to receive commands from the GUI. This is mainly to control the Photo Mode, its delay between each photo is taken and resetting the Photo Mode. The video stream will also be saved to disk if logging has been enabled.

3.8.4 Photo mode

The Photo Mode enables us to capture high-resolution photos from the camera. If the Python program receives a command from the GUI to enter Photo Mode, the capture resolution changes from 680x420 to 3280x2464. This resolution makes each frame too big to be sent over just one DatagramPacket, so the images are saved to the SD card of the Raspberry Pi. The image folder only holds up to 1000 images because the SD card only has around 5 GB of free space. If the folder is full, the GUI automatically sends a command to exit Photo Mode. The folder then has to be cleared from the GUI in order to enter Photo Mode again. The photo folder on the Raspberry Pi has been set up to an FTP server, which the GUI retrieves the images from whenever the "GET IMGs" button is pressed. It was also added a button to clear the images from the RPi FTP folder.

3.8.5 Emergency mode

To secure safe operations, we have implemented an emergency mode. When an alarm for either I^2C error, leakage in the camera housing or if the emergency mode button is pressed, a visual red warning label will appear, and an alarm sound will go off. It also sets the control mode to *manual*, the target depth to 0 meters, and sends a command to the ROV to set the wings in rising position.

3.8.6 Multithreading and responsibility-driven design

Making the ROV multithreaded was crucial to get the desired speed and reaction time. It was essential to keep the response time from the GUI to the ROV as close to real time as possible. In this project, we defined real-time to be a command sent from the GUI to the ROV, processed, and sent back to the GUI in under 10 ms. A picture of the GUI from the sea trial can be seen in figure (4.3). From the top right corner, we can see that the ping is 3.24 ms. This varied a bit, but it is within our definition of real-time for this project.

This was achieved by being aware of responsibility-driven design and how it was implemented into real-time programming from the start. Both the GUI and the ROV application follow the same mindset. An instance of the shared resource class called Data is created; this is only responsible for setting and getting stored values. This instance is shared by almost every other class in the project, but by doing so, we are in danger of not making the variables thread safe.

To counter this we could use synchronized, but when a synchronized method is accessed, all the other synchronized methods will be queued up until the first one is processed. This would make the program slow, so it was decided that this method would not fit our project. Instead, we decided not to make anything synchronized, but to keep the application thread safe we made sure that the set-methods would only be used from one place at a time. Here the responsibility-driven design was important. When the *ReadSerialData* class is gathering data, that class is handling the data and sends it to the correct set method in the *Data* class. By doing so, all the other classes in the program will have access to the shared resource all the time. The only drawback by doing this is the danger of writing to a variable at the exact same time as the variable is read. The result of this can be wrong read data. We tested this multiple times, but could not see any corrupt data, and since we are updating the data so frequently, one or two corrupt readings will not have any catastrophic effect on the ROV. We decided that this was good enough for our application.

3.8.7 Data logging

Data logging is a crucial part of prototyping. Therefore we made sure that we logged all necessary data from the ROV and GUI. The class *LogFileHandler* is responsible for handling the log files. The *LogFileHandler* is an executor task that runs every 100 ms. This means that the log interval also will be 100 ms. The log is especially crucial for understanding how the ROV moves through the water, as we are losing visual contact as soon as it dives. The log files will only generate when the "Start logging" button in the GUI is pressed. For each time the log is restarted a new set of log files will be generated. Three files will be generated in "C:/TowedROV/Log/". The log files are called "Data_Log", "ShipPos_LOG" and "Telementry_LOG". At the end of each log file name, a date and time stamp will be added when they are generated. The logs are stored as CSV files; this way files can be directly imported to Matlab. Many other softwares also support CSV files, and it exists numerous converters online for conversion into different file types. The time of each log entry is also stored within the log itself. We are also storing the number of log entries in the log. This is important since the executor task does not necessarily run with a specific interval. If the CPU is extremely busy, it can miss the 100ms interval it was supposed to log data. Say that it missed three intervals, this will mean that the log is 300 ms behind real time. When the CPU is available again, the executor task will try to reach real-time again by doing all the missed tasks as fast as possible. This means we can get three log entries at the exact same time. Depending on what information the user wants, both the time stamp and the entry

number are included in the logs.

We are writing to the CSV files on the fly, meaning that the *LogFileHandler* is constantly having each log file open. This speeds up the logging process since we do not have to open and close the file each time we are writing to it. Another advantage of this is in case of a blackout in the ROV, or the GUI stops working for some reason, the log results will still be available on the operating computer.

ShipPos_Log

The ShipPos_Log is storing Point, Time, Latitude, Longitude, Speed, ROV Depth, GPS heading. This file can be converted to a Google Earth KML file using this online converter http://www.gpsvisualizer.com/convert_input. The CSV is sorted, so the converter places the correct data in the correct column in the generated KML file. By doing so, we can directly import the file into google earth and inspect the traveling route of the boat.

Data_LOG

The Data_LOG is storing most of the data from the ROV and GUI. They are not sorted to fit any special program but can easily be opened and analyzed in Matlab or Excel. Some of the variables are IMU, depth, actuator command, actuator feedback, temperatures, and voltage data.

Telementry_LOG

The Telementry_LOG file is following the GPX 1.0 standard. By doing this, we can use programs like Garmin Virb to visualize the data and overlay them over a video feed at a later time.

3.8.8 Serial data handler

Connecting a serial device to a computer does not guarantee that the devices are connected to the com port every time. Therefore it was developed a class that would able to determine which com port the IMU, Feedback Arduino, GPS and echo sounder used. The class determines what is connected to each com port by opening up the port, and try to read the data the device sends. A feature in the Arduino is that when the com port is opened, they will reboot. This means that the Arduino will be running through the setup section, and in that section, we can send a unique

identifier over the serial line. The identifier follows the same string setup as described in section (3.8.1) and will send for example <GPS:0>. When the serial data handler receives this, it will store the com port and the name of the connected device in a list. If the com port cannot be read from, it will then be marked as Unknown. As the devices use different baud rates, serial data handler has to search through multiple baud rates to find all the devices. Since every com port has to be opened, waiting for the Arduino to reboot, waiting for the buffer to be read, analyzing the data and finally closing, the search process will take a long time. It will especially take a long time on computers with many virtual com ports. Two methods was implemented to speed up the search process. The first method is checking for unusual com ports. Both in windows and Linux the standard com ports have a standard prefix. In windows there is "COM", and in Linux, there is "dev", only these will be added to the search list. This speeds up the process by excluding virtual com ports. The second method is removing known ports from the search list. When a device is found, for example COM3 is GPS, that com port will then be excluded when searching through the comports with a different baud rate.

3.8.9 PID

The PID controller is used to control the position of the actuators. The actuator positions are based on a setpoint depth and a measured depth. The result of a properly tuned controller is a steady system with no steady-state errors. By receiving a setpoint depth and a measured depth, the controller can then calculate the difference and use it for the proportional and integral control. For a quick development of a PID controller, miniPID has been used. The miniPID is a open source code developed to simply implement a PID controller in Java applications.

3.8.10 Start-up scripts

Start-up scripts were created on the RPis; this way we could remove the need of starting programs manually. These scripts start the necessary programs for running the ROV when it boots up. The scripts were made as a shell script (.sh), and added to the Raspberry Pis /etc/init.d folder, where all the start-up scripts are located. It is important to add the ampersand behind the command for running each script. This makes sure the program runs continuously. Finally, the start-up scripts had to be added to the Raspberry Pis start-up sequence by editing the /etc/rc.local file and add the path of the two scripts to the end of it.

```
# !/bin/bash
cd /home/pi/Desktop/WorkingCode/
python3 udpvs.py &
python3 TcpController.py &
exit 0
```

Start-up script for the camera RPi.

This script starts the video stream and the TCP server applications.

```
# !/bin/bash
cd /home/pi/Desktop/TowedROV/dist/
java -jar TowedROV.jar &
exit 0
```

Start-up script for the camera RPi.

This script starts the main application for the ROV.

Remote access Raspberry Pi

To make software changes on the Rasberry Pis, one must use VNC and connect to 192.168.0.101 for the RPi located in the electrical housing, or 192.168.0.102 for the RPi in the camera housing. Before this is done, make sure the power supply, ROV, and the Ethernet cable are connected to the suitcase. It is also important to set a static IP address of the operating computer to 192.168.0.20.

3.8.11 Version control

In this project, we have used version control to backup all of our project files. The changes done have a time stamp to identify each revision made, and the person responsible for the changes.

This tool is essential for software development when multiple people are working on the same code. There exists two types of version control; central version control, and distributed version control. Central version control is stored on a single server. When this is used a user checks out the file, and uploads the file back to the server again when he is done working. The distributed version control also stores the files on the server. However, the users that are working on the project must clone the whole project to their computer in order to work with it. By doing this, if

a server crashes, there will still be a copy of the project locally with every user [2].

3.9 Calculations

In this section, we will go through the calculations we needed for the project.

3.9.1 Power demand

Calculating the power demand is important to ensure the electronics and fuses are correctly dimensioned. It is important to note that the ROV contains three transformers. We are referring to them as "12V heavy consumer", "12V utilities" and "5V supply". The 12V heavy consumer is dedicated to the actuators. 12V utilities contain all other electronics that run on 12V, for example, the echo sounder. The 5V supply contains all 5V devices such as the Raspberry Pi. For more details about the power distribution, see section (3.11.1).

3.9.2 Worst case power demand

Worst case power demand is the max power each device will use. It is worth noticing that the ROV will never draw this amount of power, but it is important data for further calculations.

Device	Heavy cons (12V)	Utility (12V)	5V supply
2x Actuators	10 A		
External lights		2.5 A	
Echo sounder		5 A	
Internal lights		0.3 A	
Sensors and small consumers		0.7 A	0.3 A
2x Raspberry Pi 3b+			3 A
Ethernet switch			1 A
Total consumption on secondary side	10 A	8.5 A	4.3 A

Table 3.2: Worst case power demand

This would result in a worst case power demand of 22.8 A.

3.9.3 Realistic power demand

The realistic power demand is an estimate based on the normal operating power of each device. It is doubtful that we will run everything at 100 % of its capacity. The stall current of each actuator is 5 A each, as shown in the worst case power demand table 3.2. However, when they are operating, they are measured to only use 40%, which is 4 A.

There are multiple voltages and transformers in the system, and it is therefore essential to know that the devices rated current draw are based on their rated voltage. Because of this, will we have to calculate the turnover ratio of the transformers and their efficiency to get the current draw on the primary sides of the transformers.

The ROV is supplied with 36V. The 36V->12V transformers turnover ratio can be calculated using the equation (2.4). This gives a turnover ratio of 3. The 12V - 5V transformer has a turnover rate of 2.4 by using the same equation. To find the efficiency of the transformer we use the graph in figure (3.15) and figure (3.16) for the 12V -> 5V transformer. We are using 36V, and we can therefore, use this graph to calculate the current draw by using equation (2.4), but dividing the result with the efficiency, we get from figure (3.15 and 3.16).

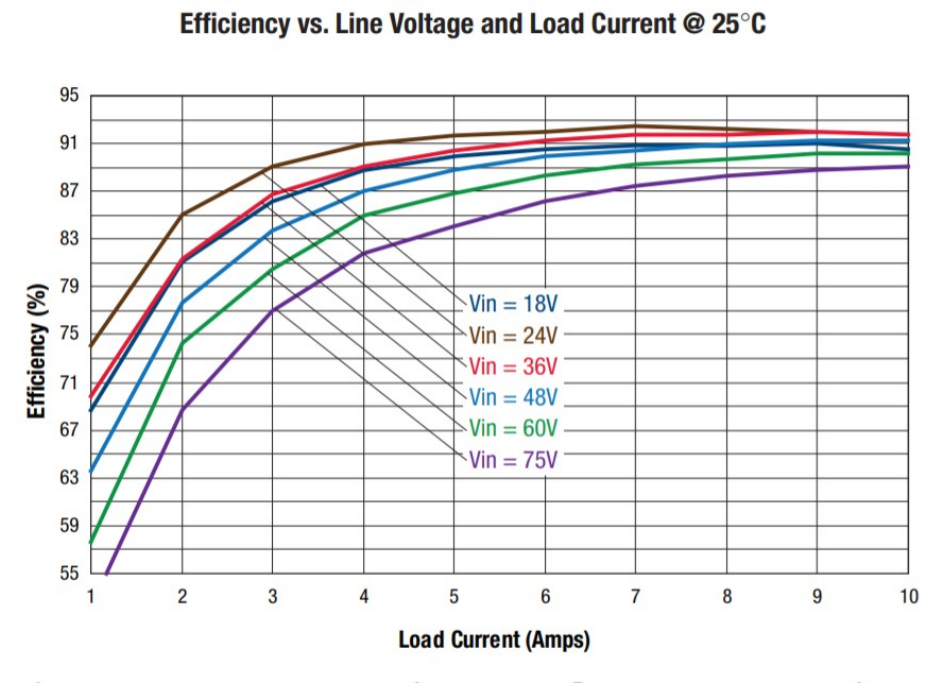


Figure 3.15: UWE-12/10-Q48NB-C transformer efficiency

Models

Order code	Input voltage range	Output voltage	Output current max.	Efficiency typ.
TEN 25-2410WI		3.3 VDC	5'500 mA	82 %
TEN 25-2411WI		5 VDC	5'000 mA	85 %
TEN 25-2412WI	10 – 40 VDC	12 VDC	2'500 mA	89 %
TEN 25-2413WI	(24 VDC nominal)	15 VDC	2'000 mA	89 %
TEN 25-2422WI		±12 VDC	±1'250 mA	89 %
TEN 25-2423WI		±15 VDC	±1'000 mA	89 %
TEN 25-4810WI		3,3 VDC	5'500 mA	82 %
TEN 25-4811WI		5 VDC	5'000 mA	85 %
TEN 25-4812WI	18 – 75 VDC	12 VDC	2'500 mA	89 %
TEN 25-4813WI	(48 VDC nominal)	15 VDC	2'000 mA	89 %
TEN 25-4822WI		±12 VDC	±1'250 mA	89 %
TEN 25-4823WI		±15 VDC	±1'000 mA	89 %

Figure 3.16: TEN 25-2411WI transformer efficiency

Device	Heavy cons (12V)	Utility (12V)	5V supply
2x Actuators	4 A		
External lights		2 A	
Echo sounder		0.8 A	
Internal lights		0.2 A	
Sensors and small consumers		0.7 A	0.3 A
2x Raspberry Pi 3b+			0.8 A
Ethernet switch			0.3 A
Supply for 5V		0.686 A	
Total current draw on secondary side	4 A	4.386 A	1.4 A
Total power consumption on secondary side	48 W	44.4 W	16.8 W
Total current draw on primary side including transformer loss	1.498 A	1.624 A	0.686 A
Total power consumption on primary side including transformer loss	53.928 W	58.46 W	8.232 W

Table 3.3: Realistic power demand

This gives us a total current draw of 3.122 ampere on the ROV.

3.9.4 Power loss in cable

The utility cable has been reused from the previous ROV. We could therefore, measure the resistance in the cable to 6.1Ω. From this, we can calculate the power loss when the ROV is using

3.122 amps, which is calculated in section (3.9.3). By using equation (2.3) we get a voltage drop of 19.044V. This is too much since the remaining voltage is 16.956 V and therefore, below the minimum operating voltage of the ROVs transformers, which is 18V, we will get an unstable system. The max current draw on this cable and battery setup can be found by using equation (2.3) and using the difference between the supply voltage and the minimum operating voltage for the transformer as ΔU. This gives a max current draw of 2.786 A. This is too little, and has been a problem throughout the project. We based our work on the calculation from the previous Towed ROV thesis, and they were not accurate, see (4.2.1).

Unfortunately, there was not enough time to do anything about the low max current draw, but we were interested to see if it would be possible in the future to operate the ROV at full capacity. To use the same cable but get enough power to the ROV, we need to turn up the voltage. The ROV is designed to accept a voltage range between 18V -> 75V. If we raise the voltage to 70V, we can recalculate the power loss and max possible current draw. By using equation (2.3) and using the difference between the supply voltage and the minimum operating voltage which gives us a ΔU on 50 V, we get 8.06 A. This is sufficient for the ROV since it is 2.89 times higher than what the utility cable could deliver with 36V and well over the realistic power demand calculations.

3.9.5 Calculated operation time

The ROV battery pack has three 12V batteries in series. This gives us 36V. Each battery has the capacity of 75Ah, and since the batteries are connected in series, the total capacity does not change. If we use the realistic power consumption from section (3.9.3) we can calculate the max operating time by using equation (2.5). This gives us an operation time of 24.023 hours.

3.10 Design and prototyping

This section will describe the process of designing, and building the prototype.

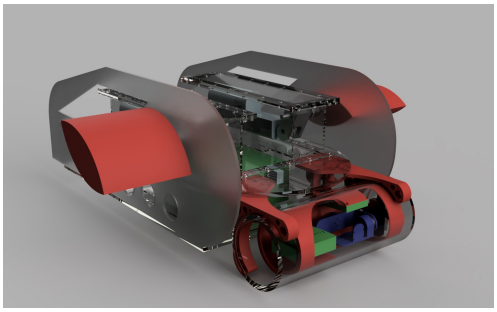


Figure 3.17: First prototype design.

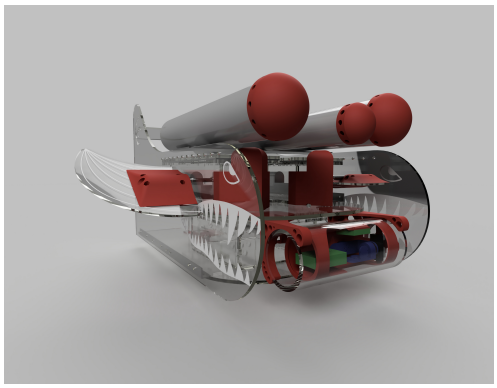


Figure 3.18: Final design.

3.10.1 Budget

To get a price estimate and overview of the components needed for this project, a budget was created in Excel. It was essential to keep the budget at a minimum, since we are developing a prototype.

			Color defenitions:
Today's currency exchange USD/NOK	8,469		Ordered
			Recieved
Personal expenses	kr 1 965,30		Already in stock
			Backorder
			Not recieved
Total:	kr 16 160,21		Not ordered

Figure 3.19: Total expenses and color definitions

3.10.2 3D modeling

To save time and work, we wanted to use the 3D files that were made in the previous ROV project. This turned out to be more difficult than expected as the 3D parts were not handed in as attachments in the previous bachelor thesis. This meant that we had to remodel the entire ROV.

We had learned by reading the previous bachelor thesis that many of their design calculations was wrong due to forces that they did not anticipate. They therefore, decided not to use any of their calculations, but instead continue by trial and error. Because of this, we discussed if we should redo their calculations and simulations for the hydrodynamics part, but the group decided that it would take too long and not give any valuable information. Therefore we decided to

go for a design that was easy to build and maintain, and also use trial and error like the previous group did.

As the modeling was complex and we wanted the ability to easily alter the size of the ROV at a later time, we decided to use Siemens NX to model it. We started with modeling the actuators; this would give us a good indication of how big the surrounding boxes should be. It was also important to start with the actuators since they are the most mechanically complex parts. In Siemens NX we could assemble the actuators, boxes, shafts, and wings. Moreover, simulate the model to inspect what angles we would achieve with the different designs.

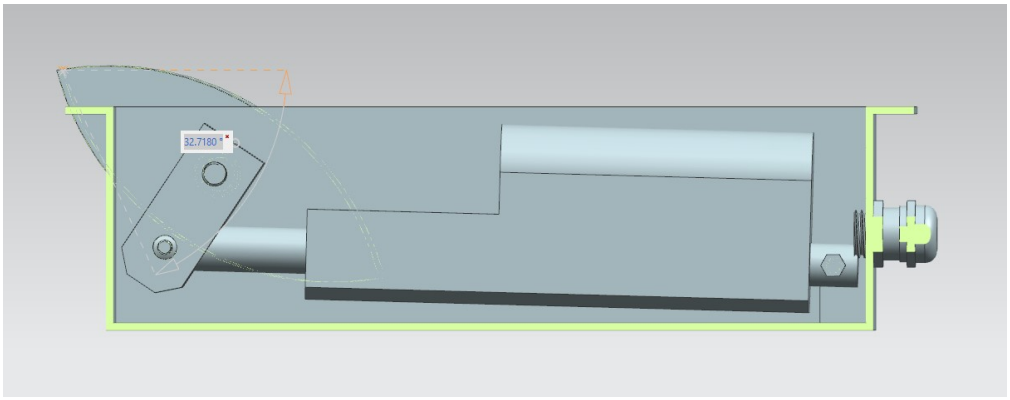


Figure 3.20: Actuator movement simulation 32.7 degrees

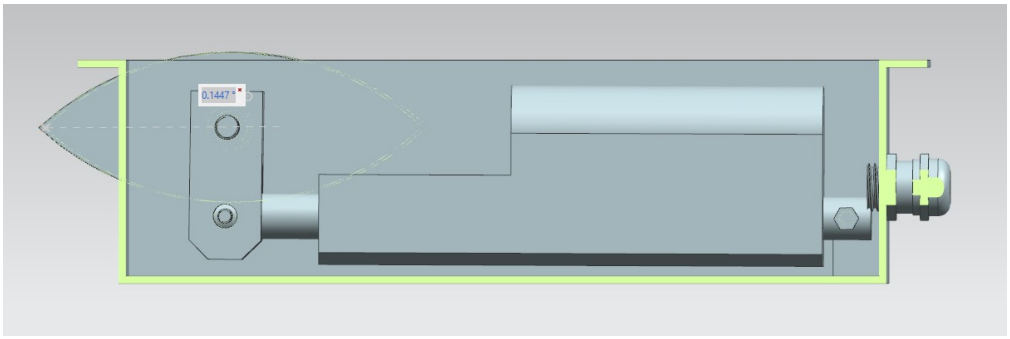


Figure 3.21: Actuator movement simulation 0 degrees

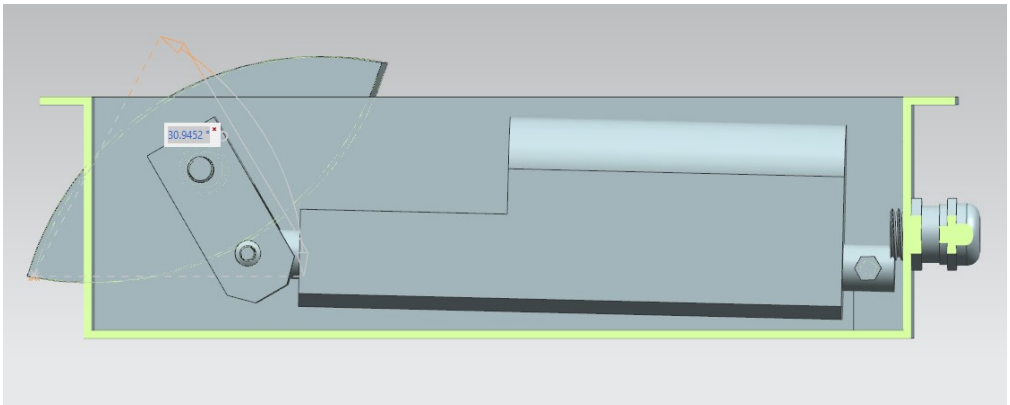


Figure 3.22: Actuator movement simulation -31 degrees

This allowed us to make the boxes just long enough to get 60 degrees of movement on the wings, see figure (3.20, 3.21 and 3.22). We were also able to simulate the assembly process; this saved us a lot of time, since we could detect future assembly problems. One aspect we focused on was making the boxes large enough to fit future components. By doing this, we could add unforeseen components without the need of rebuilding the boxes.

The process of the custom metal boxes turned out to take a lot more time than expected. It took a long time for us to design all the boxes in 3D and be sure that they were just the right size. After that, it took a long time to get a price estimate. The price estimate ended up to be roughly 16 000.- NOK. This was too expensive, especially for a prototype. We lost a great deal of time waiting for this, and had a huge impact on the total time used to build the ROV.

The university had bought a new laser cutter which would arrive after some weeks, and it was then decided to make the boxes out of 8mm thick Plexiglas and cut them on the new laser. This led to a redesign of the boxes since they were initially meant to be welded. To improve the structural strength of the Plexiglas boxes, boxed joints were designed. This way, we increased the areas we could glue and made the box stronger. Since we now are planning on using Plexiglas boxes instead of metal boxes, we would need to fill the boxes with oil to withstand the pressure in the depth of the ocean. The risk of leaks is also higher, but the oil will protect the electronics to some degree. To make the joints, we imported the designs to Autodesk Fusion 360. This program is a lot simpler than Siemens NX and made it faster to make the joints. Each of the walls, roofs, and floors of the boxes were then exported to Autodesk AutoCAD to make them two dimensional. When they were two dimensional, we could add them to Corel Draw, which is a software used to optimize the layout of the cutting arrangements and send them to the laser cutter.

The internals of the camera housing was also designed in Siemens NX, where a frame was designed to fit precisely inside the housing and a custom gimbal to fit within the frame. The internals of the camera house was done before the Plexiglas cylinder arrived, since we had the dimensions of the cylinder we were able to recreate the cylinder and then model the components we needed. We were able to assemble all of the components in 3D and test the degrees of movement the gimbal would achieve with different mounting options.

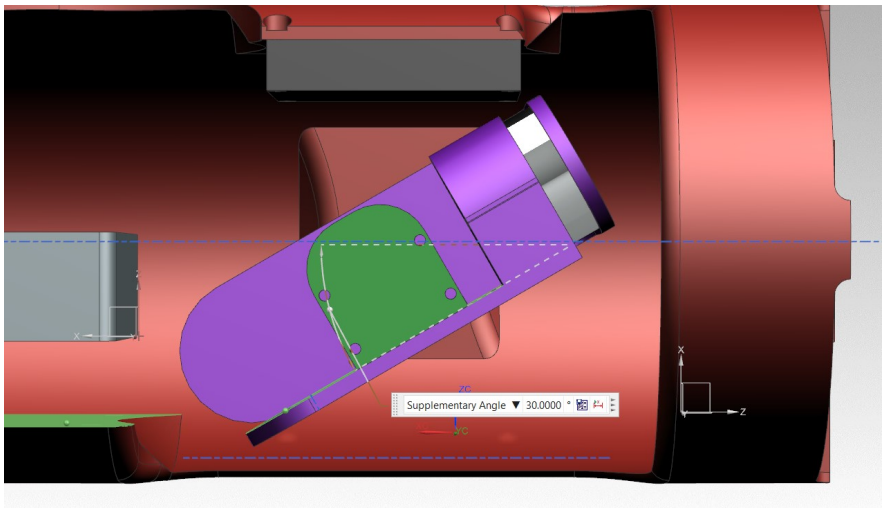


Figure 3.23: Gimbal's movement freedom

3D printing

A problem that could occur was how the PLA would react to saltwater. PLA is a biodegradable material and will over time, rot away. After doing some research, we found out that PLA is not as biodegradable as we thought. According to a research study produced by the California State University in Chico, when PLA is exposed to a marine environment, the PLA will not noticeably disintegrate after 365 days [13]. For PLA to biodegrade, it need to be placed in landfills with the right conditions. We concluded that it was not going to be a problem since the ROV will dry on land, and the PLA would not degrade in sea water.

We tried to keep 3D printing to a minimum, since it takes a lot of time to model and print. Some of the parts will be on the outside of the boxes; this means that they will be in direct contact with sea water. Therefore, it was decided to print all parts with 100% infill. This would minimize the amount of water it will soak up and reduce unnecessary weight when the ROV is above water. All components are also designed with draining holes so the water can move freely in and out of the 3D printed components. This is necessary since we will get the same pressure

on the inside as the outside and therefore reducing the thickness and strength needed for the components.

3.10.3 Choice of design

As we discussed in section 3.4, each of the designs had their strengths and weaknesses. We had to take into account the price of materials, and the construction time. The flat wing design was discarded because we wanted to avoid as much 3D printing as possible. The group has experience from earlier projects with 3D printing, and knows that both design and printing is time-consuming. In cooperation with our supervisors, we went through the two remaining designs. The bullet design was discussed but the risk of water leak would be too high. We ended up with the old design but making it longer, to make it more stable in the water.

3.10.4 Choice of oil

The boxes had to be filled with oil to even out the pressure on the inside of the boxes. When the boxes are filled with oil, the pressure on the inside of the boxes would be the same as on the outside, and the oils we are considering are non-conductive, and will provide extra security for the electrical components since the oil will, to some extent, work as a protective layer around the electronics. Different factors need to be considered when choosing the type of oil. The most important feature of the oil is that it must be environment-friendly. There are different types of oil such as motor oil, mineral oil, and vegetable oils. The motor oil is not an option, because it contains a good amount of chemicals that are not good for the environment. Therefore the two types of oil we could consider, are mineral oil and vegetable oil. The pros and cons of these oils are listed below.

Mineral oil:

Pros:

- Certain oils are transparent
- Non-drying oil
- Won't become rancid

Cons:

- Expensive
- Hard to find local supplier

Vegetable oil:

Pros:

- Some types are cheap
- Edible and won't cause harm to the environment

Cons:

- Can become rancid and dry out
- Non transparent

Based on the information above, the mineral oil would be the best option. However, because of the high prices for this type of oil, vegetable oil was the preferred option. The type of oil we finally decided to use was canola oil. This is because of the low price and the amount of polyunsaturated fat. Canola oil has a lesser amount of fat than many other types of vegetable oils. The lesser amount of polyunsaturated fat, the longer time it takes for the oil to become rancid.

3.10.5 Old vs. new design

In the following list is a summary of the major changes we have done to the ROV:

- The new designed is based on the previous ROV. As they wrote in their thesis page 114 [15] they had a theory that a longer ROV would be more stable. Our supervisors and the group agreed that this would make the ROV more stable, so our primary goal with the design was to make it longer. We also wanted the design to be more open; this would let the water flow through the ROV and improve stability.
- We also wanted more room in the boxes so it would be easier to add future components. Because of this, the whole ROV got significantly bigger.
- All the boxes are made out of Plexiglas, this would be an improvement from the old ROV, since the 3D printed boxes where weak and oil leaked through the print itself. Since the Plexiglas is stiffer, we would not have the problem with the lids on the boxes warping as they struggled with on the old ROV.
- We also designed PCB cards to reduce the number of loose cables, see section (3.11.1). This is a considerable improvement since the number of wires is decreased, and therefore, the amount of components that can fail is lower. By doing this, we are also getting a more straightforward and more comprehensive system.

- The suitcase and the plug and play system has significantly lowered the setup time for the ROV. Because of this, we did not need any tools for setting up the ROV in the boat, only a wrench in size ten to mount the echo sounder rod.
- The gimbal is a new feature on the ROV; this will level the camera no matter if the ROV is pitching or rolling. The gimbal will also give more consistent pictures.
- The wings on the old ROV is cambered. This was not needed, and in the discussion section from the previous project they discussed that the wings could be replaced with symmetrical ones. We therefore changed the wings to be flat Plexiglas and changed the quick mounts for the wings. This way, we did not risk the wings and quick mounts to be fused together, as they did on the old ROV.
- Threaded holes with blanking plugs are placed on top of each box, which gives us the ability to refill the boxes with oil if necessary.

3.10.6 ROV frame

The frame of the ROV is designed mainly of Plexiglas and angled aluminum brackets for holding sides and floors together. The difference between the new and old ROV is mostly size; the new is wider, taller and longer; this to improve stability. All flat areas on the front side of the ROV have a 3D printed nose cone created for improving hydrodynamics.

3.10.7 Wings

The wings on the ROV is used for controlling the depth of the ROV. The previous ROV had 3D printed the wings in a modular design, meaning that they could extend the length of wings if needed. When we disassembled the old ROV, we struggled with taking apart the wings. The PLA had soaked in water and expanded, so it was impossible to take them apart. We ended up destroying the wings to get hold of the shafts. The existing wings were also designed as a NACA airfoil profile; this would give the wings an upward push when they were in a neutral position. Because of the problems we had with the 3D printed wings, we decided to use Plexiglas instead. This meant that the wings would be completely flat and not have a NACA profile. The group had discussed this with their supervisors and concluded that it was not necessary with a NACA profile.

Since we did not have any standard to follow for the wing design, we took inspiration from shark fins. We chose this design since the look fits the ROV, and they are tilted backward, which meant that the ROV would not get proportionally wider with the increased surface area of the wing. This would make it much easier to transport. Two sets of wings were made, one small and one big. The small one had a surface area of 34.4 cm^2. The big one had a surface area of 54.64 cm^2. If the small wings could not generate enough force to make the ROV dive, we could easily change the wings in the field.

Since the ROV was wider than before, new shafts were also needed. The new shafts followed the same design as the old ones but they were 20mm longer.

When we tested the small wings in the ocean, they worked excellently. They were able to produce enough force to drag the ROV underwater at a controlled rate, as we can see in section (4.2.2).

Quick mounts for wings

To make the wings easy to change we 3D printed quick mounts for the wings. The quick mounts are held on the shaft with a bolt. If this is unscrewed the whole wing including the quick mounts can be slid off the shaft. The wings are connected to the quick mounts with four M6 bolts. They are easily accessible for changing wing type.

3.10.8 Towing plate placement

The placement of the towing plate is vital for the ROVs diving capabilities. The placement is directly correlated between the center of gravity and the center of lift. The center of lift can be roughly said, is straight above the shafts. This is not entirely accurate since other parts of the ROV is also producing lift at certain angles. This is ignored since a trial and error approach would be faster than calculations. If the towing plate is placed too far behind the center of lift, the ROV will flip when it is diving steeper then a certain angle. However, placing the towing plate too far in front of the center of lift, and the ROV will struggle to dive. Therefore the side plates of the ROV was designed with multiple holes so that the mounting plate can be moved back and forth. Since the placement of the towing plate will force the ROV down, the ROV has a trim pitch, which is placed at the back of the ROV to compensate for this (3.10.15).

Since the side plate will drag the entire weight of the ROV we were not sure if the Plexiglas would

be strong enough. We wanted the side plate only to be 6mm thick to reduce the total weight. Therefore a FEM analysis was conducted.

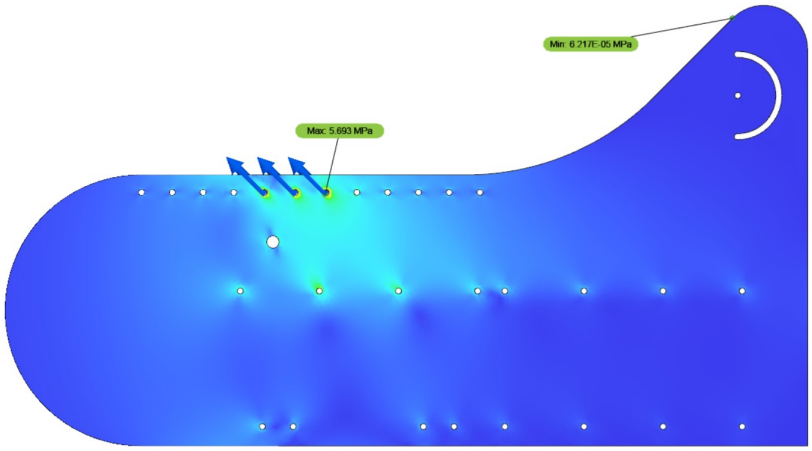


Figure 3.24: FEM analyze 150Kg, safety factor 7.026

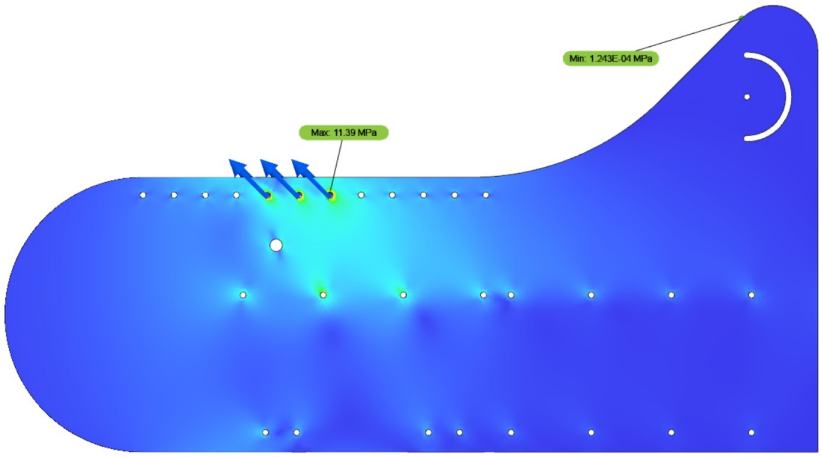


Figure 3.25: FEM analyze 300Kg, safety factor 3.513

As we can see in figure (3.24) the safety factor is 7.026 when the forces on the plate are 1500N. In figure (3.25) the safety factor is only 3.513 when the force on the plate is 3000N. Since the weight of the ROV is circa 47 Kg, we do not think that the weight of the ROV, when towed with 6 knots, will exceed 1500N. The FEM analysis shown in figure (3.24) and (3.25) are only done on one plate, since the port side and the starboard side plates are both connected to the towing

plate; the forces will more or less be equally divided between the plates. Because of the FEM analyses, we decided to use 6mm thick Plexiglas plates.

After we towed the ROV in the ocean, we could not see any damage to the Plexiglas. Which means that the calculation gave us a good enough estimate on the forces exceeded on the plate. The placement of the towing plate was also decent since we had no problem with getting the ROV to dive and rise.

3.10.9 Hydrodynamics and buoyancy

The design has to be as hydrodynamic as possible. We also needed the ROV to have positive buoyancy to prevent it from sinking. The ROV has to be hydrodynamic in order to relieve stress on the towing cable, the brackets it is mounted to, and the surface vessel. 3D printed nose cones were designed to be glued on to the buoyancy pipes.

Figure 3.26: Nose cones.

The weight of the ROV was 46.5 kg when the boxes were filled with 17 liters of oil. The volume of the boxes were calculated manually, and was 18245 dm^3, we could find the buoyancy of the ROV (2.1). The ROV itself has a buoyancy of 18,7 Kg. The pipes on the bottom were designed to be just too little to make the ROV positive buoyant. The top pipes will lift the remaining weight. The reason we have pipes on the bottom and the top is because of the wings. If the bottom pipes lifted all the weight, then the water line would have been on these pipes. With the pipes on the top taking the remaining weight, one could make sure the wings were below surface, since the water line would have been on the pipes above the ROV. The volume of the bottom pipes are

244.44×10^6 mm^3, and by using equation (2.1) we get a buoyancy of 25 Kg. The pipes and boxes combined had a buoyancy of 43,7 Kg. The top pipes would then need a buoyancy of a bit more than 2.8 Kg in order to have positive buoyancy of the ROV.

Figure 3.27: Bottom part for buoyancy.

Figure 3.28: Top part for buoyancy.

3.10.10 Mounting of Plexiglas

The Plexiglas plates are screwed together with M6 bolts and nylon nuts. We tested a lot how tightly we could tighten the bolts on small test squares of Plexiglas. It was crucial that we used a torque wrench, so we did not over tighten the Plexiglas and crack it. Especially on the box lids, we wanted to have an even distribution of pressure over the rubber gasket. We used the torque wrench to test 8mm Plexiglas until it cracked. We found out that with 13Nm, the Plexiglas cracked. Therefore we decided only to use 10 Nm to be on the safe side. A problem we discovered with this method was the choice of nuts. When we did the crack tests, we used regular nuts and got a stable result. However, the nuts we are using on the ROV is nylon nuts. This meant that the torque would be different since some of the torque is lost in the friction between the nylon and the bolt. We tried to redo the crack test with the nylon nuts, but depending on the wear and tear on the nylon we could not get a stable result. The Plexiglas would break somewhere between 5-13 Nm, and this was not accurate enough. In the end, we tightened the bolts manually with a wrench to avoid cracks. This worked well, and we did not experience any further problems.

A problem with the 4 mm Plexiglas is microcracks when we tighten the bolts, see figure (3.29 and 3.30). The 4 mm is only used for extra waterproofing, and the microcracks did not have any effect on the waterproofing.

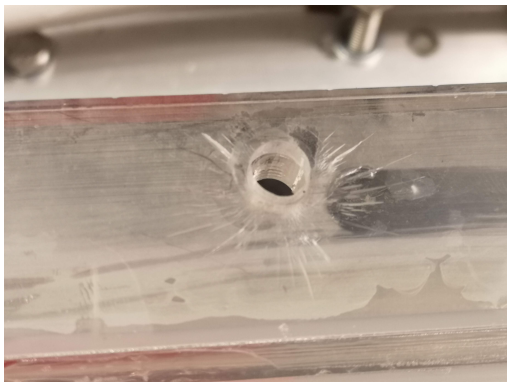


Figure 3.29: Cracks, 4mm plates.

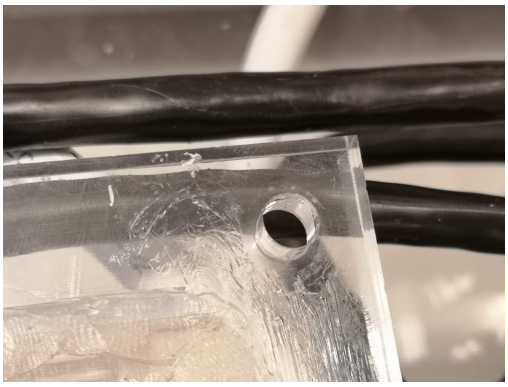


Figure 3.30: No cracks, 8mm plates.

3.10.11 Plexiglas boxes

The ROV is equipped with different boxes for different uses. On the ROV there is a total of four boxes; electrical box, echo sounder box, and two actuator boxes. Most of the equipment used within these boxes are not water resistant and is therefore placed in these boxes. It is therefore important that the boxes are waterproof. From the previous bachelor thesis, it was used 3D printed boxes. These boxes were not waterproof, and a different solution had to be considered. Therefore a decision was made of using 8mm thick Plexiglas as the material for the boxes since this was the thickest plates the supplier had and therefore had the highest material strength. Another advantage was the increased area to apply glue on. These boxes were 3D modeled, and then cut with the laser cutter. The pieces were then placed together and glued together with a compound called ACRIFIX. The ACRIFIX is a mixture that chemically melts the Plexiglas and welds it together. After the ACRIFIX had dried, a few layers of silicone was applied around the edges inside of the boxes as an extra layer for waterproofing. Because of the thickness on the laser beam, the final cut plates were a tiny bit smaller than designed, which created a small gap between each plate when assembling the boxes. Because of this, the type of ACRIFIX we used (ACRIFIX 1S 0117) was not the best option, as there is a thicker type (ACRIFIX 1S 0109) of ACRIFIX, which is designed to fill gaps like these.

The gluing process was very time consuming since the ACRIFIX compound needs time to dry, and the pieces also need to be held tightly together with clamps while drying. After the boxes

were complete, each of the boxes had to be tested to see if there was any leakage. The rubber gaskets are made from natural rubber, which enables us to cut them with the laser cutter. The gaskets were mounted between the box flange and the box lid. We also applied some gasket sealant for extra safety. After all the boxes were proved to be waterproof, they were mounted on the ROV.

The boxes are also designed for refilling them with oil. On top of all the boxes, there is a hole with a blanking plug. These blanking plugs are accessible even after everything is assembled.

Our design was based on Plexiglas plates that were 8mm thick, and when the boxes were ordered and delivered we noticed that the plates were 6mm thick. This was because the order was wrong, and new plates had to be ordered. The construction was therefore delayed.

Cable glands

Holes for the cable glands were cut with the laser cutter, and threaded by hand to fit M16 and M20 cable glands. The M16 cable gland is for the cat6 cable used for communication, and the M20 is for the power cable between the electrical box and the other boxes. To seal the threading a layer of silicone was added before mounting the cable glands. The cable gland used in the project is rated for 9 bar.

Echo sounder box

The echo sounder box has a hole to mount the echo sounder, a 3D print was designed to cover the neck of the echo sounder. The manufacturer of the echo sounder has also cast it in epoxy for an enclosure and is therefore waterproof.

The echo sounder box is designed for easy removal. The box is held in place with six bolts but is easily accessible, and the whole box can be slid out of the ROV. The cable for the extra I/O is located here and connected to the Arduino UNO in the main electronic box. The orange pair is in use for the echo sounder NMEA signal. The blue is connected to analog pin 4. The white/blue is connected to analog pin 5. Green is connected to digital pin 6 and white/green is connected to digital pin 7. The brown pair is spare and is not connected to anything in the electronic box.

3.10.12 Camera housing

The camera housing is the only box that is not filled with oil, as the oil would have interfered with the cameras picture quality. Therefore a cylindrical waterproof enclosure was purchased from BlueRobotics. The enclosure is rated for approximately 90 meters of depth and is suitable for this projects usage.

Inside the enclosure, there is a 3D printed frame to make it easier to remove and place the electronics inside the cylinder. In the camera housing there is a Raspberry Pi with a camera, a heating element, a circulation fan, multiple leak sensors and motors for a gimbal to stabilize the camera.

To prevent condensation within the camera house, a heating element was added, a fan was placed inside and also some silica-gel. The heating element was used to heat the inside of the housing, while the fan was used to circulate the air flow. And lastly, the silica-gel is used to remove any form of condensation within the housing.

Since the camera housing did not have any oil, it was important to seal the cables in a way, that no oil from the other boxes could travel through the cables and enter the camera housing. The way this was done was with a layer of glue and self-amalgamating tape. Because of the camera house is not filled with oil, there is a much higher chance for water or condensation will short circuit the 5V. Because of this, we did a consequence analysis of the camera house filled with water:

If a short circuit on the 5V supply occurred, the following critical components will fail:

1. The 5V fuse on the main electronics card will blow.
2. The Raspberry Pi in the camera house would lose its power
    (a) Video stream is lost
    (b) ROV depth information is lost
    (c) External lights will shut down
3. The Raspberry Pi in the electronic box would lose its power
    (a) Depth to seafloor information is lost
    (b) Actuator control is lost

The result of this can be the ROV colliding with the seafloor. The risk of this is too high for the group to accept, so we decided to address this issue. By installing an additional fuse in the main electronics box for the camera house supply, we can protect the 5V in the main electronic box. This means we will still have controls over the actuators and can start an emergency ascend, avoiding the seafloor.

3.10.13 Camera

A Raspberry Pi Camera V2 is used for video capture. The camera has high resolution and decent picture quality, at least if it has enough light. It is very compact and takes little place, and can be powered by the Raspberry Pi, which makes the camera ideal for filming and taking pictures for our project.

Figure 3.31: The colour loss under water at 20m depth, filmed with a GoPro [19]

As figure (3.31) shows, a problem with cameras filming underwater is the lack of light, especially red light. This can be compensated for by using a red filter in front of the lens. We tried this, but it seems like the RPi camera has some automatic night mode function, when the red filter was placed in front of the camera, the video stream went gray. The filter was therefore removed, which is why most of the videos we logged turned out very green. However, it seemed like when the camera was close to the sea bed, the "night mode" would get activated. This was something we did not have control over, and even if we did find a way to control this manually, the camera quality turned out to be insufficient in dark environments.

3.10.14 Gimbal

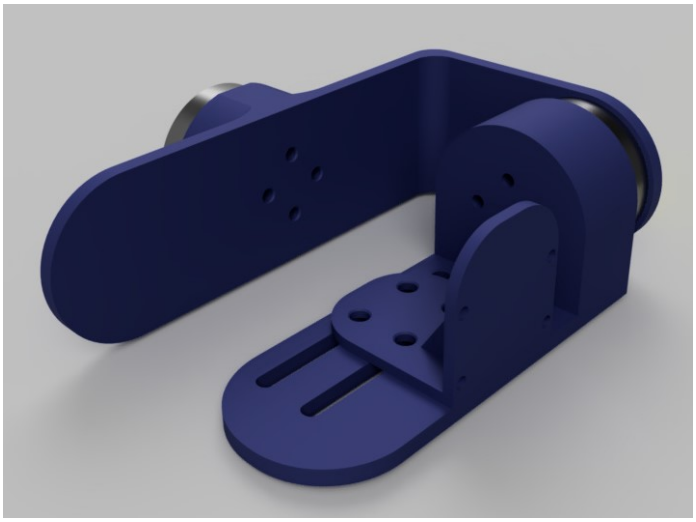


Figure 3.32: The gimbal design

In this project, we have designed a gimbal for removing unwanted shaking and sudden movements from the camera, as well as compensate for the pitch and roll of the ROV. We designed a gimbal, and 3D printed it to fit on to the gimbal motors and RPi camera. Since we only need to compensate for the pitch and roll of the ROV, we only needed a two axis gimbal. The gimbal, therefore, consists of two brushless DC motors controlled by a STorM32 gimbal controller. We configured the controller by using the O323BGCTool software. The PID parameters (K_p, K_i, and K_d) were found by the Ziegler-Nichols method described in section (2.3.1).

Since the Raspberry Pi V2 camera is so small and light, the gimbal will need to be balanced to counteract the weight of the pitch motor. This is done by adding weights to the gimbal until it is leveled without power.

3.10.15 Pitch trim

On the rear part of the ROV, there is mounted a spoiler. The purpose of this spoiler is to help with trimming the pitch of the ROV. Since the bracket connected to the wire that is responsible for dragging the ROV is mounted in the front of the ROV, there will be a force in the front causing an upward tilt; therefore this spoiler will help to adjust the pitch of the ROV while it is being towed. It is therefore, possible the spoiler will help stabilize the ROV. It is static and must be adjusted manually.

3.10.16 Towing arrangement

A solution had to be made for fastening the utility cable to both the ROV and the boat. For the ROV, we reused the mounting bracket which was made last year, only making it a bit wider to fit our ROV. The cable is then fastened to the mounting bracket with a cable thimble and a shackle.

On the previous Towed ROV project, they suspected that the reason for the constant tilt on the ROV was because of the incompatibility between the thimble and the shackle. The result of this incompatibility was that the thimble could jam itself askew in the shackle. To prevent this, we modified the thimble and made it smaller so it could move freely within the shackle.

To attach the yellow utility cable to the boat, we needed a system that did not interfere with the outboard motor. Therefore we made a rope system that is attached to each side of the boat and goes behind the motor, see figure (3.33). On this cable a pulley is attached so it can move freely on the rope, the utility cable is fastened to this pulley. By doing this, the force on the rope that is attached to the boat will be more equal to the port and starboard side of the boat. To the pulley, there is also attached a small fender to prevent the whole rope system from sinking.

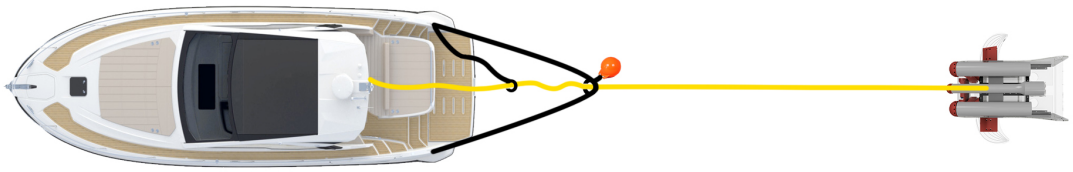


Figure 3.33: The mechanism for fastening the utility cable to the boat.

3.10.17 One-line diagrams

The one line diagrams which shows how all of the components in our system are wired are presented in figure (3.34 to 3.38).

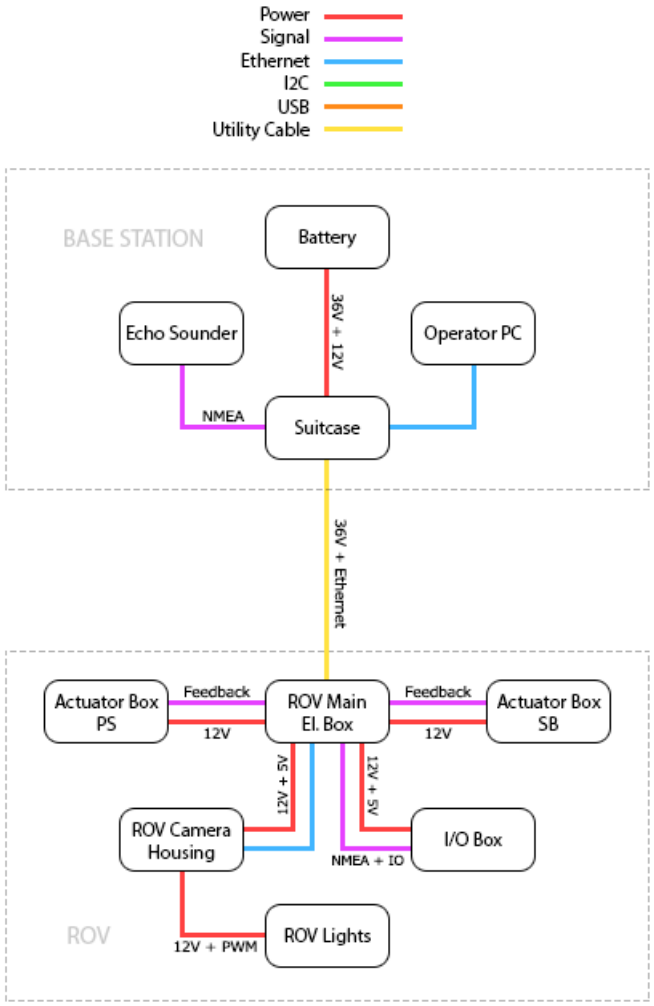


Figure 3.34: One Line Diagram - MAIN.

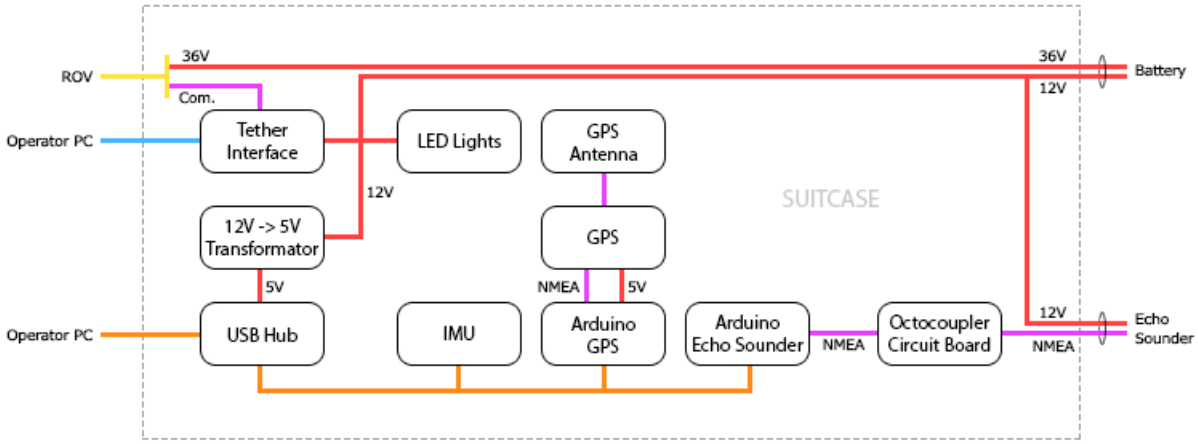


Figure 3.35: One Line Diagram - SUITCASE.

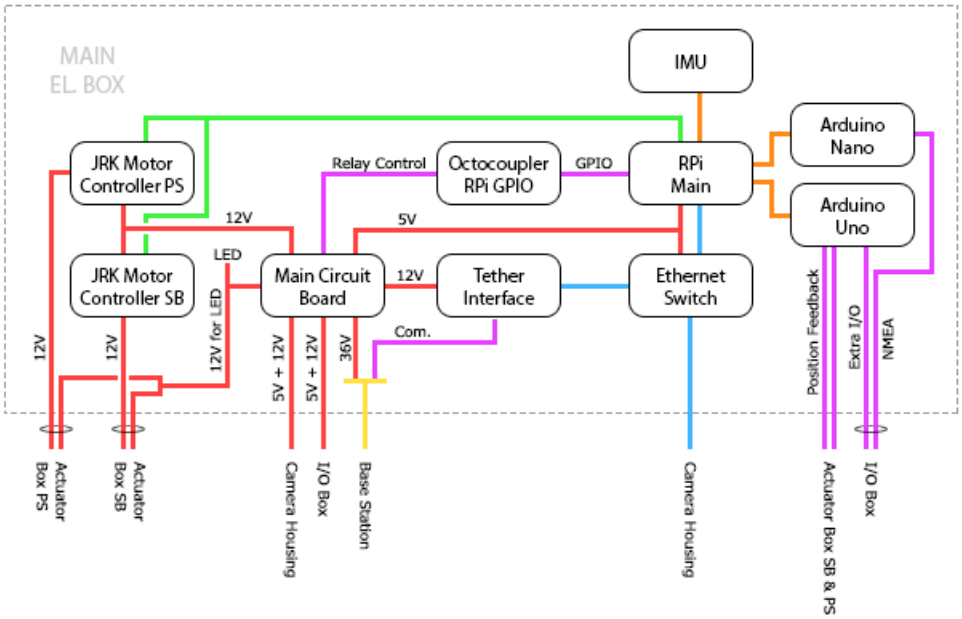


Figure 3.36: One Line Diagram - MAIN EL BOX.

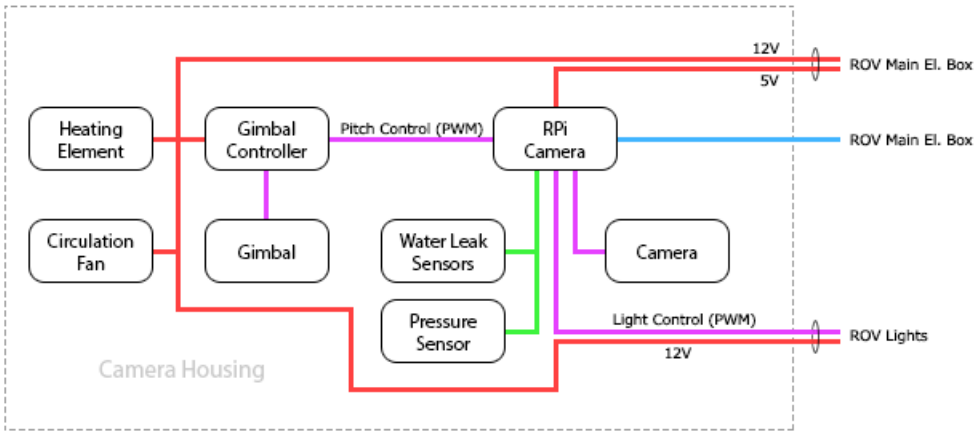


Figure 3.37: One Line Diagram - CAMERA HOUSING.

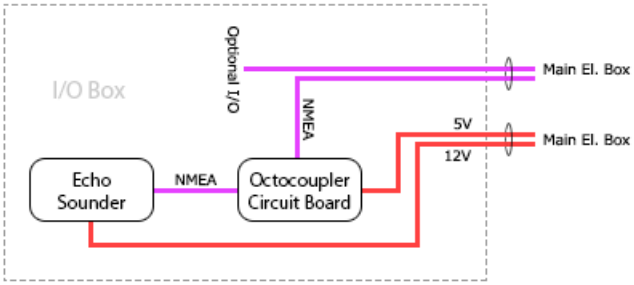


Figure 3.38: One Line Diagram - I/O BOX.

3.11 Circuit boards

In order to connect everything in the different boxes, have we made our own PCBs. By doing this, we do not have to use as many loose wires, and we will get more compact and orderly circuits. We designed the PCBs in a program called EasyEDA. We later ordered them from jlcpcb.com. We have three different PCBs:

- Main electronics card
- Optocoupler card
- I^2C extender card

3.11.1 Main electronics card

The main circuit board has three DC/DC converters, two 18-75V -> 12V transformers and one 10-40V -> 5V transformer. This means that the ROV is capable of handling up to 75V. There are several other electrical components on the PCB. For a complete list, see the budget and order list in the appendix (9) :

- 2X Transformer UWE-12/10-Q48NB-C
- 1x Transformer TEN 25-2411WI
- 2x RS PRO 10000μF Capacitor
- 1x Panasonic 4700μF Capacitor
- 2x RS PRO Resistor 100 ±5% 2W
- 1x Resistor 330
- 5x SPDT Non-Latching Relay 5V, 5 A

The main circuit board has gone through some changes since its initial design. In V1.0 the positive and negative output for one of the transformers where flipped. We fixed this and ordered V.1.1. Unfortunately in V.1.1 one of the PCB lines overlapped each other. We fixed this by cutting off the line, and soldering two wires over the other line. See the red circle in figure (3.40). In V.1.1 the Aux 4 and Aux 5 label was switched, this was corrected in V.1.2. In case we decided to use another microcontroller that did not share common ground with the relays, we implemented a cut line and a spare ground to circumvent this scenario, see figure (3.40). We also implemented a connection if we wanted battery backup for the 12V, this was not needed at the time we built the ROV but could be a necessity if bigger loads were added to the ROV.

The fuses and transformers are dimensioned based on the calculations in section (3.9.2). This is the worst case power demand, and the fuses and transformers should be able to handle this. The main electronics card has three transformers, and therefore we refer to them as "12V heavy consumers", "12V utilities" and "5V supply". The 5V supply is supplied from the 12V utilities. The 12V heavy consumer and 12 utility transformer is the same type. They have a capacity of 10A each; therefore we use 10A fuses on these. The 5V supply transformer has a capacity of 5A and is protected by a 5A fuse. The main fuse in the design should be a 20A fuse, but we did not see this as necessary, because of the realistic power demand calculations (3.9.3). We therefore changed it to a 15A fuse.

The main PCB card also includes three status LEDs. Each of the LEDs are connected on the secondary side of the transformer. We can use these lights to determine if one of the transformers is not giving out power or if one of the internal fuses has blown.

The main supply for the card can be anywhere between 18V to 75 Volt as the transformers are capable of converting this range to 12V.

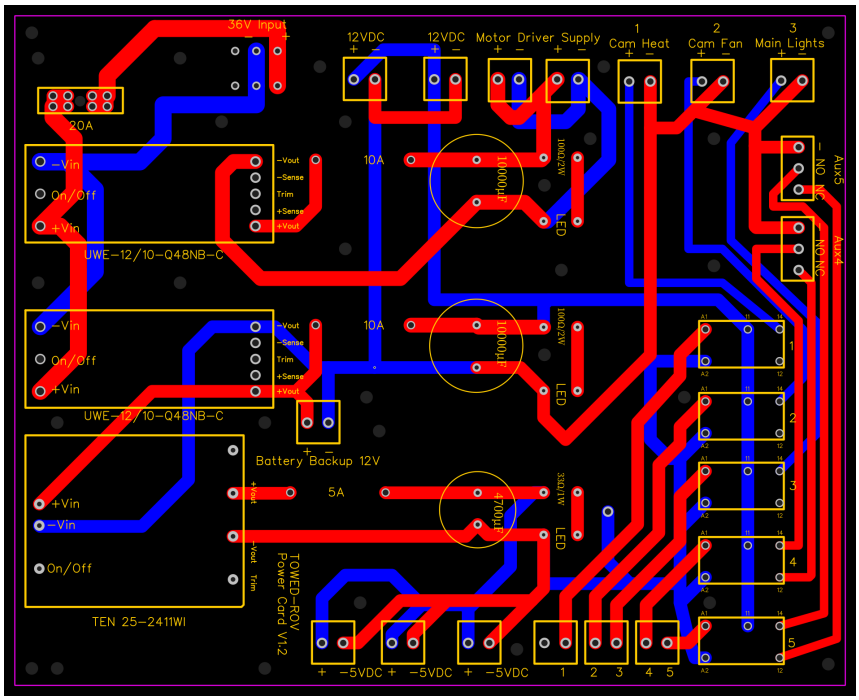


Figure 3.39: Main Circuit Board V1.2.

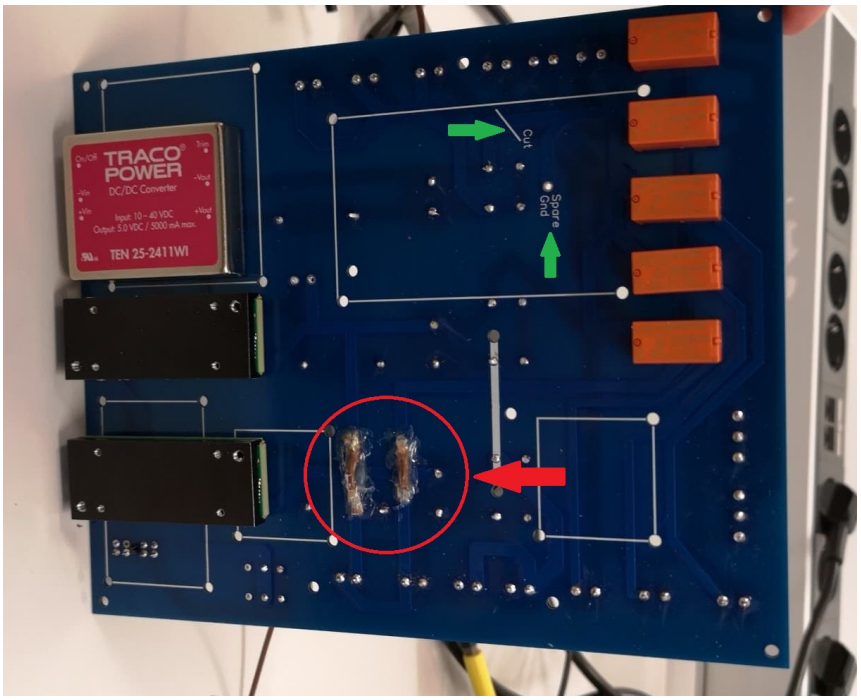


Figure 3.40: Main Circuit Board V1.1 back side.

3.11.2 Optocoupler card

The optocoupler circuit board is used to simplify the connection of an optocoupler. We used these optocouplers between the Arduinos and the echo sounders. The logic of the echo sounder NMEA signal is 12V DC. The Arduino uses 5V DC as input and will, therefore, be damaged if connected directly to the echo sounder. The optocoupler card was necessary for connecting these two devices. A 4 channel optocoupler was used in case we needed more than two channels for the echo sounder.

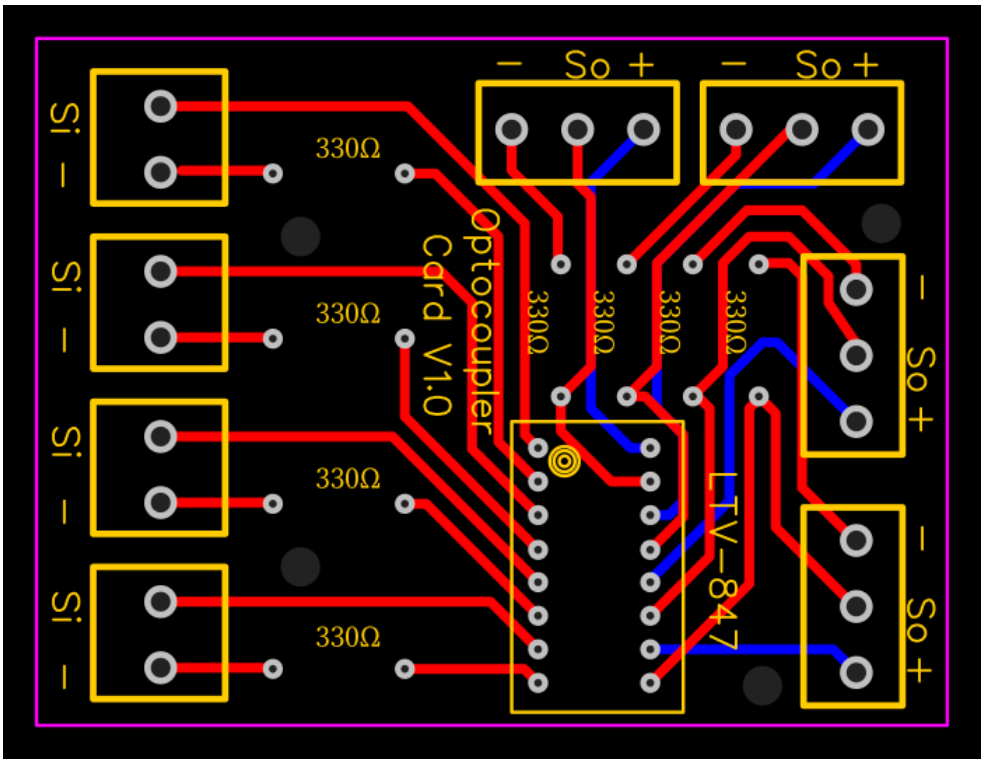


Figure 3.41: Optocoupler circuit board.

We discovered a design fault with the Raspberry Pi and the relays on the main circuit board. Since the Raspberry Pi logic is 3.3V, the relays would not activate since their coil is designed for 5V. We did not discover this until the whole ROV was built and it would take time to order relays that was 3.3V compatible as they were not in stock at RS-Online.com. We realized we could tweak the optocoupler card by using different resistors on it. By doing so we could use the 3.3V logic and control the 5V relays. The drawback of using this card is that there are only four channels available. And we have 5 relays on the main circuit board. Therefore only four first relays are in use.

3.11.3 I^2C extender card

The I^2C circuit board was designed to boost the I^2C signal between the boxes. By boosting the signal, we could send the I^2C signal over a length of 3-25 meters, and the signal would not drown in noise. The distance could be changed on the onboard switch. We ended up with not using them since they caused more trouble than what we gained from them as described in section (3.12).

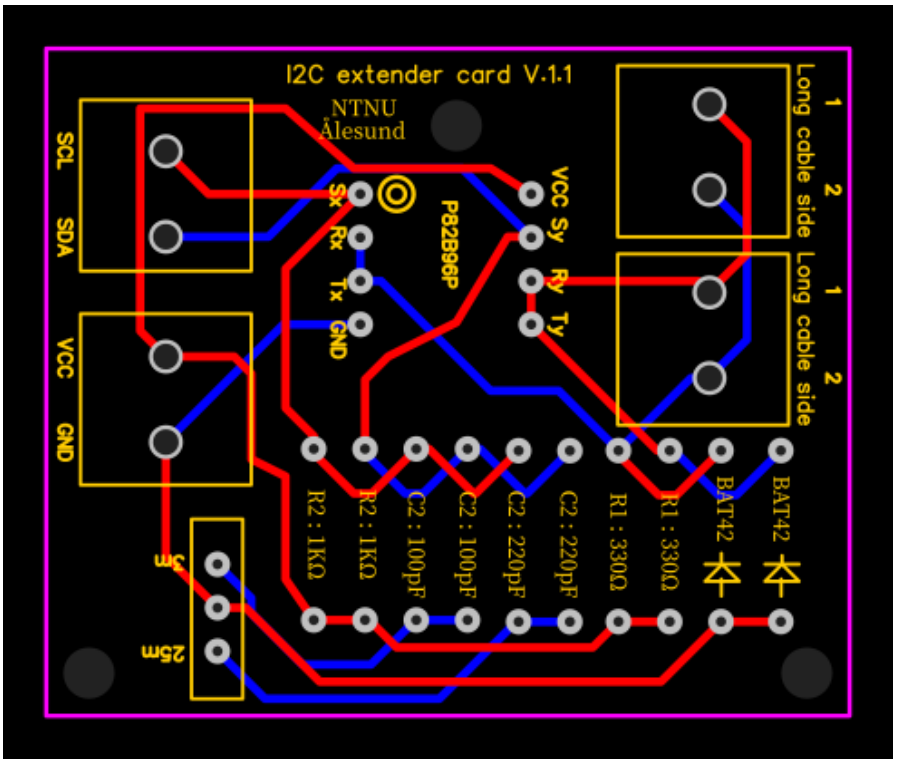


Figure 3.42: I^{2}C Circuit Board.

3.12 I^{2}C and noise

I^{2}C was designed to use on internal circuits with small distances. Over time the I^{2}C protocol has become more and more popular. By using I^{2}C we could connect multiple devices on the same bus:

Main electronic box:

- JRK cards
- Razor IMU

Echo sounder box:

- Additional I/O Arduino
- Pressure sensor
- Temperature and humidity sensor

The group that worked on the old ROV also tried to use I^{2}C without any luck. They struggled

with noise and could not effectively read the I^2C signals. As we have experience with noise and shielding of sensitive equipment from our earlier projects, we decided to give it a try. As we went through the previous bachelor, we could not find any systematic testing or results of the shielding job they had done. Therefore we suspected that they did not have correct experience with a noisy environment to make the shielding good enough. We decided to use shielded network cables, and a chip called P82B96P. This chip is a dual bus buffer designed to carry I^2C signals over a distance of up to 200 meters. As we only need to carry the signal between the boxes, we decided that with this chip and shielded cable it would be worth it to try I^2C.

We tested the I^2C signal by sending it over a 3 meter long cable. As we can see in figure (3.43) the capacitance in the cable is so high that the signal is no longer squared. After inserting our I^2C extender card we got a much better result as we can see in figure (3.44).

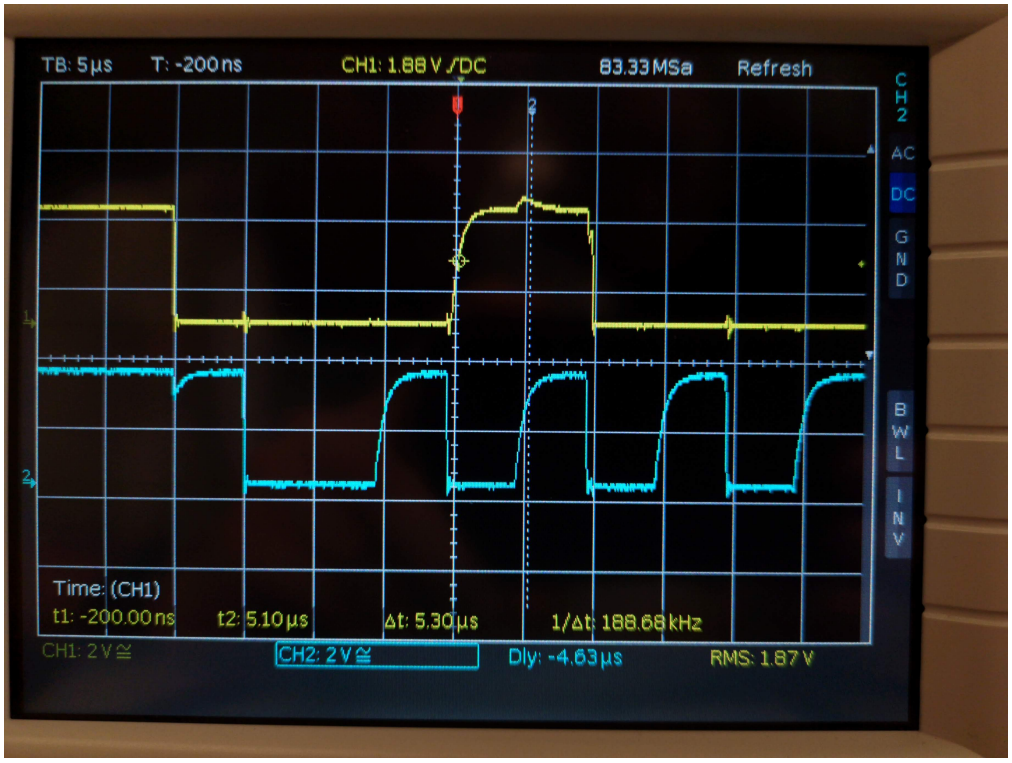


Figure 3.43: Before I^2C Circuit Board.

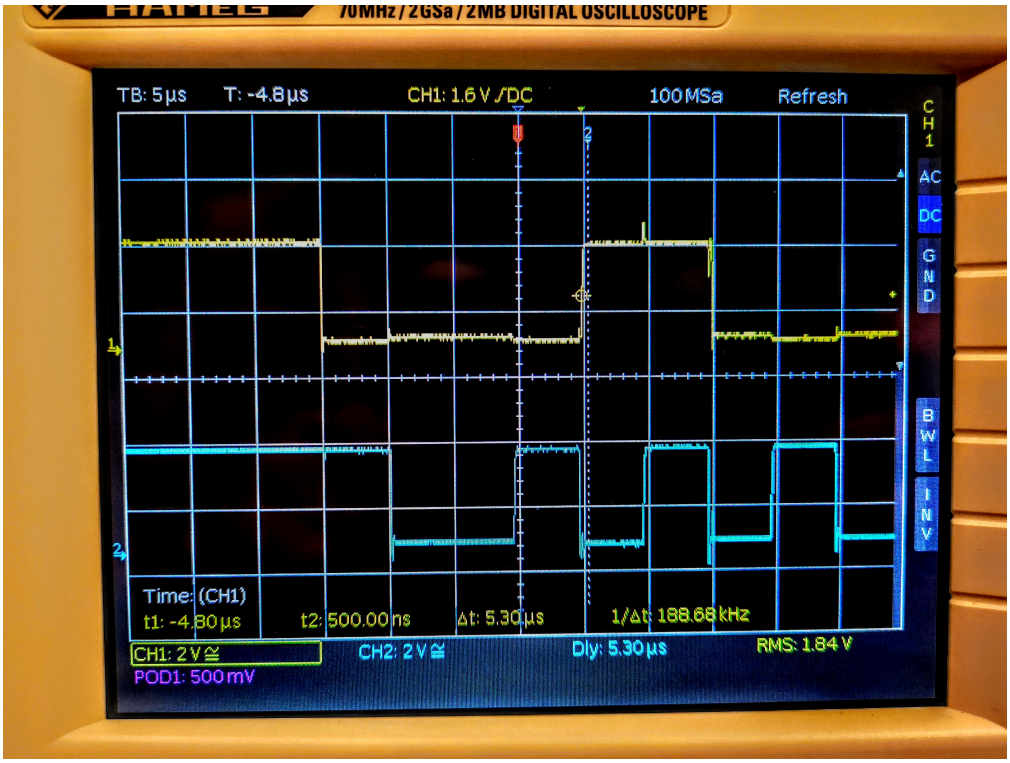


Figure 3.44: After I^{2}C Circuit Board.

The I^{2}C extender card was connected to every I^{2}C device. However, when we powered the ROV through the suitcase with the battery pack connected, we could not find any of the I^{2}C devices when we scanned for them. It turned out that when all the I^{2}C devices was connected on the same bus together with the I^{2}C extender card, we needed to add pull-down resistors to the bus. This was not planned, and therefore, not implemented in the I^{2}C extender card. We also saw that the total length of the cable was only 40 cm. By using the I^{2}C extender card the additional cable length would make more problems than the I^{2}C extender cards would fix. The I^{2}C extender card is also prone to noise since the PCB itself is not shielded in any way. Therefore we decided not to use the I^{2}C extender card. When we removed the I^{2}C extender cards, we were able to find all the I^{2}C devices.

The next problem was the reliability of the I^{2}C bus. Sometimes, the devices would disappear from the I^{2}C bus and sometimes ghost devices would appear on the bus. This was unreliable and could not be fixed in the software, and therefore, needed to be fixed at a hardware level. Therefore we separated the bus. The pressure sensor and temperature/humidity sensor was moved to the camera house and connected to the Raspberry Pi there. The two JRK cards, Razor IMU and the Arduino I/O card was still in the main electronics box. By doing this, we shortened

the cable length, and the I^2C devices in the camera house worked flawlessly. We tested the I^2C in the electronic box by sending commands to the JRK cards. This worked, but at a later point we found out that we were not able to get complete feedback data from the JRK cards. Since the deadline for the project was closing in, we decided to make some radical changes as we describe in the following sections.

Arduino for extra I/O and echo sounder feedback

The Arduino Nano for the extra I/O was supposed to be placed in the echo sounder box for easy access. It was also supposed to receive the NMEA data from the echo sounder, and send it to the Raspberry Pi over I^2C. However, because of the I^2C problems, we decided to move the Arduino Nano inside of the main electronic box and use serial communication instead. We also decided to add an Arduino Uno and connect it to the Raspberry Pi with a USB cable and move the extra I/O to this. By doing this, we got a better system in many ways:

- The Arduino Nano could handle the NMEA parsing from the echo sounder alone. Since the parsing was extremely slow, it would have affected the extra I/O.
- We could reprogram the Arduino through the Raspberry Pi, making it easy to change the echo sounder to another type or change the code.
- We would still be able to add extra I/O in the echo sounder box since the signal cable, 12V and 5V are still present in the echo sounder box. For the wiring guide of the extra I/O, see section (3.10.11).

JRK input and feedback

The position feedback from the actuator is a potentiometer that changes the resistance determined by the position of the actuator. The Pololu JRK G2 18V19 has direct input for this and can send the feedback back to the Raspberry Pi over I^2C. Since the feedback did not work reliable enough, we therefore moved the feedback from the JRK cards to the Arduino UNO. This was a better solution because:

- The Arduino UNO is powerful enough to handle inputs from multiple sources. This gives us a higher update rate for actuator feedback.
- The JRK cards are not remotely programmable. To change the feedback or calibrate it, we would have to dismantle the whole ROV to reach the cards. Therefore it was better to

leave the feedback on the Arduino. This meant that if the actuators are changed, we can remotely calibrate the feedback.

- We could use the Arduino to map the feedback to the same scale as the actuator position commands (0-254). Making the Java code more simple. We, therefore, mapped max upward position to 254, max downward position to 0 and neutral to 127.

Razor IMU

The Razor IMU is capable of sending its data both over I^2C and serial communication. We also connected this to the Raspberry Pi with USB and used serial communication. The advantages of doing this are:

- Faster data transfer
- The ability to reprogram the IMU after the boxes are sealed.

3.12.1 Solution for writing to JRK cards

The only remaining device on the I^2C bus inside the main electronics box was the JRK cards. We are only sending the JRK cards commands through the I^2C, since there was no other suitable method for this purpose. When thoroughly testing the ROV on land we noticed that we sometimes got an I^2C write error. We got this error since the Java code was not able to write to the JRK card. After some troubleshooting, we found that the Raspberry Pi 3 b+ has a documented bug in its hardware I^2C (GPIO pin 2 and 3). This meant that it was not able to handle clock stretching; this was a problem since the JRK cards are using clock stretching. This was fixed by using the Raspberry Pi's software I^2C instead (GPIO pin 23 and 24). To change this we added *dtoverlay=i2c-gpio,bus=3,i2c_gpio_sda=23,i2c_gpio_scl=24* to the config.txt in the Raspberry Pi boot folder. This resulted in better results, but when we scanned after addresses on the bus, we sometimes got ghost addresses, which means addresses that showed up, but did not actually exist. Because of this, we decided to shield the I^2C. We did not want to use the stiff and shielded cat 6 cable because, as the single-stranded wires were prone to break. Instead, we made our own shielding. We took two multi-stranded cables and twisted them together. Then we used aluminum tape around the twisted pair and connected the aluminum to the Raspberry Pi's ground. To protect the cable we used a shrinking tube around it. By doing all of this, all our communication problems with the JRK cards was resolved. The clock stretching issue and the

lack of proper shielding was most likely the cause of all our I^2C issues. The group discussed if we should move all the sensors back to the I^2C bus, but all the advantages with serial communication outweighed the advantages with I^2C. We therefore, decided to only use the I^2C bus for sending commands to the JRK cards.

3.13 Instruments

Different sensors are installed in the ROV. The sensors that are installed is the foundation of the control system and is necessary to make the ROV work as intended.

3.13.1 Razor 9-DOF IMU

The Razor 9-DOF IMU SEN-14001 from SparkFun was our preferred choice of IMU. The group had worked with an older type of the Razor IMU (SEN-10736) before and was familiar with the code and product. The Razor IMU contains a magnetometer, gyro, and accelerometer. All are connected on the same board which includes a SAMD21 microprocessor that is programmed with the Arduino IDE and uses Arduino C. Because of this it works like any other Arduino and is programmable over a mini USB type B. It is also highly accurate since all the onboard sensors are working together to give more precise data. The main advantage is that the data the Razor sends can be reprogrammed to fit our needs, we only need the pitch and roll from the gyro as there is no need to send accelerometer or magnetometer data to the RPi. This speeds up the data transfer. The code is created by SparkFun but needed some modifications to send the data in the format of <Key1:Value1:Key2:Value2>.

3.13.2 Echo sounder

The echo sounder is an Airmar-DST800 NMEA0183. We have used two of this type, one placed in the ROV and the second on the surface vessel. This echo sounder can measure depth, speed, and temperature in the water. It can measure up to 70 meters of depth below it, with an angle of 22 degrees. In the ROV the echo sounder is connected to an Arduino Nano through an optocoupler. In the surface vessel, the echo sounder is connected to the suitcase and internally in the suitcase through an optocoupler that is connected to an Arduino RedBoard.

3.13.3 Leak sensor

There is a total of three leak sensor probes inside the camera housing placed in different locations. The leak sensor has a bright red LED that lights up in case a leak are detected. The leak sensor itself is mounted in front of the Raspberry Pi, so the LED will be visible from the outside.

3.13.4 Internal temperature and humidity sensor

These sensors are located internally in the camera housing. The temperature/humidity sensor was added to see the temperature and humidity inside the camera housing while testing it in the ocean, see figure (4.15). The sensors proved to be accurate and gave us crucial information about the internals of the camera housing.

3.13.5 Anti condensation measures

To prevent condensation, we mounted a heating element and circulation fan inside the camera housing. The heating element is self-regulating to 40°C. The fan is used to circulate the hot air around the camera house. We also put in some silica gel packs. All the electronics are also coated with a layer of clear paint. This will help in case of condensation on the electronics.

3.13.6 Voltage measure circuit

To measure the voltage of the battery pack we made a small circuit in the suitcase. The circuit is based on the voltage dividing principle. A complete guide was found on the website Starting-electronics.org [7] which designed a circuit for measuring 50V with an Arduino. This was perfect for our use, so we recreated the exact same circuit.

3.14 Actuators

3.14.1 Actuators and oil

The actuators were submerged in oil. After we had drained the old ROV for oil and cleaned the actuators, we tested them and found out that they were slow, but working. After some weeks, we tested them again and noticed that they performed much worse. We disassembled them completely and found out that the oil had washed away almost all the lithium grease. We also found out that the actuators where using brushed DC motors to drive the actuator. We have a theory that the oil works as an insulator between the brushes and the commutator, resulting in a poor connection and therefore lower speeds. This is a problem that does not have an easy fix. We cannot redesign the actuator to work on brushless motors, and we cannot avoid filling the actuator boxes with oil. We decided to leave this as it was, and focused on the other systems of the ROV.

3.14.2 Actuator controller

To control the actuators, we used the JRK G2 18V19 from Pololu. This is capable of handling 30V and 19A, and also PID. To communicate with the JRK controller, we decided to use I^2C. The advantage of this was that we could communicate directly with the JRK from the RPi. The *I2CRW* class is used to handle the communications with the JRK. The commands for controlling the JRK are well documented in Pololus documentation for Phyton, but the huge number of options and possibilities made it hard to get a system that was so simple that it just sent position data to the JRK with Java.

The initial idea was that the JRK should handle the feedback, and it should not have any form of PID regulation; this should be handled in Java. Unfortunately, we found out that it could not be completely turned off when it also handled feedback. We therefore, decided that the K_p should be 1, K_i should be zero and K_d should be zero. This worked great, when manually controlling the actuators through the JRK G2 configuration utility, we could see that the actuators almost hit its desired position. We changed the K_p to 3 so the actuator was closer to its target, we did this because when the ROV is in manual mode the PID in Java would be disabled. The commands to the JRK was based on high and low bytes. This means that if the position of the actuator were lower than the desired target, we would need to send a high byte. The feedback was later

changed to be handled by an Arduino instead, because of the problems we encountered with the I^2C. 3.12.

3.14.3 Actuator logic

The logic for the actuator has to as flexible as possible. Meaning that manual control, distance from seafloor and depth mode, use the same logic. This is important for avoiding code duplication, but also for making the code easier to handle between the different modes. This is solved be creating a class in Java called *Logic*. This class is an executor service which runs at 200 Hz. Every time it runs, it checks if the current actuator position has reached the hysteresis of the desired position. The hysteresis is ±5 steps of the desired position. The hysteresis is needed to compensate for imperfections in the feedback reading, delays in the actuator, logic, and I^2C commands.

The I^2C commands are mapped from 1 to 254, where 1 is down, 254 is up, and 127 is the middle point of the wings. In the Arduino, the feedback is mapped to the same values.

Chapter 4

Result

This chapter will present the results achieved in this bachelor thesis. This projects main objective was to develop a prototype of a Towed ROV. This includes design, hardware, and software to control the system. This was achieved with satisfying results.

A full-scale test of the system was done at the end of the project. The towing was done by a boat at the island of Store Kalvøy, near Åleasund. All functional systems were tested while doing this, such as communication, sensors, gimbal, and the camera and video stream.

4.1 Final product

4.1.1 ROV

Figure 4.1: The final product.

The final product is a functional ROV that is stable when diving, waterproof, and is positively buoyant. Test results proved that the boxes and camera housing had no signs of leakage. Also, when the wings were locked in diving position, the diving abilities of the ROV proved to be stable while being towed. When we stopped towing the ROV, the ROV ascended back to the surface, proving that it was positively buoyant. The communication between the ROV and surface vessel worked as intended, the data was received and later plotted into various graphs, see figure (4.15, 4.19, 4.20 and 4.21).

In addition to the ROV a system on the surface vessel was created. The purpose of the system was to simplify the connection of the overall product while testing. It was built a battery case which is connected to the suitcase. This suitcase has all the electrical components needed in the surface vessel, and is responsible for distributing power from the batteries to the ROV, and connecting the communication between the ROV and operating computer.

For more photos of the final product, see section 4.5.

4.1.2 Software

The result of the final software design for the ROV will be displayed and explained in this section. As mentioned earlier, the ROV application was the server, and the GUI application was the client. The software was implemented using Java, while the microcontrollers are programmed using Arduino C.

Data flow diagram

Figure (4.2) shows the data flow between each of the components in our system.

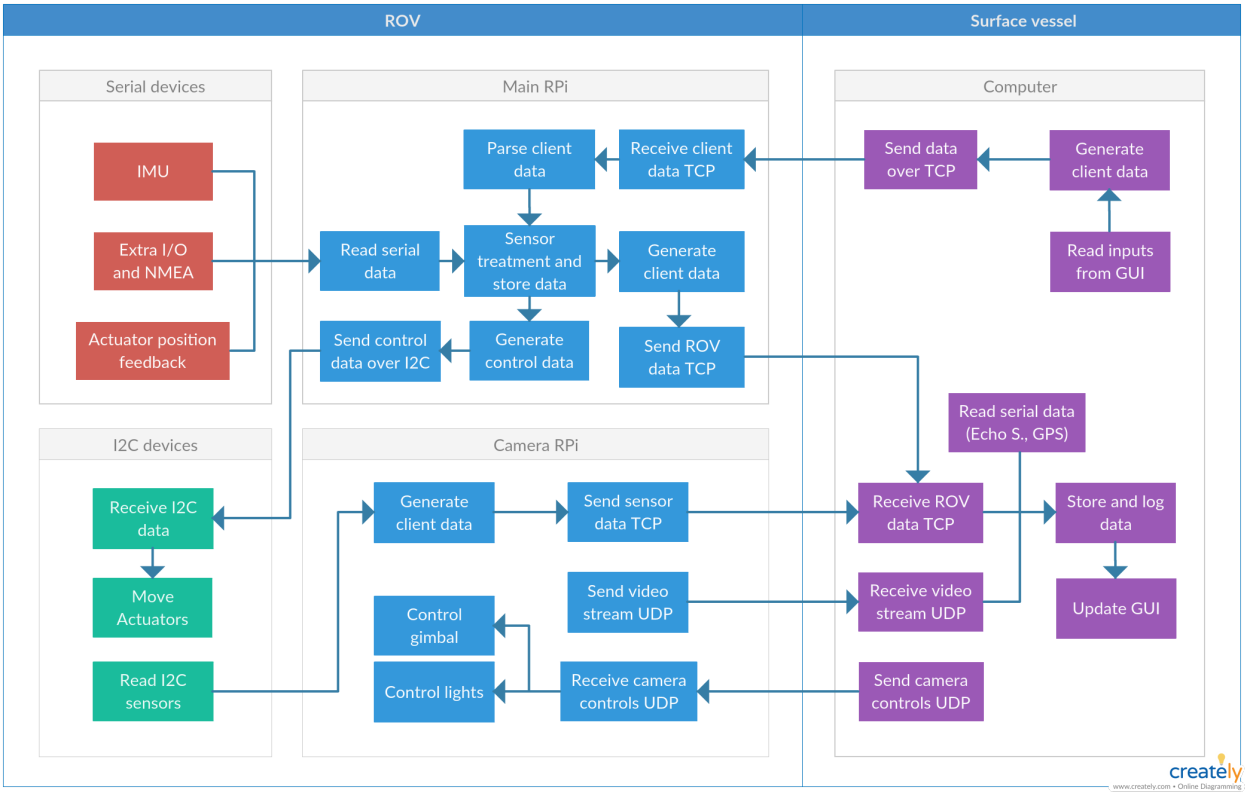


Figure 4.2: The data flow diagram.

GUI

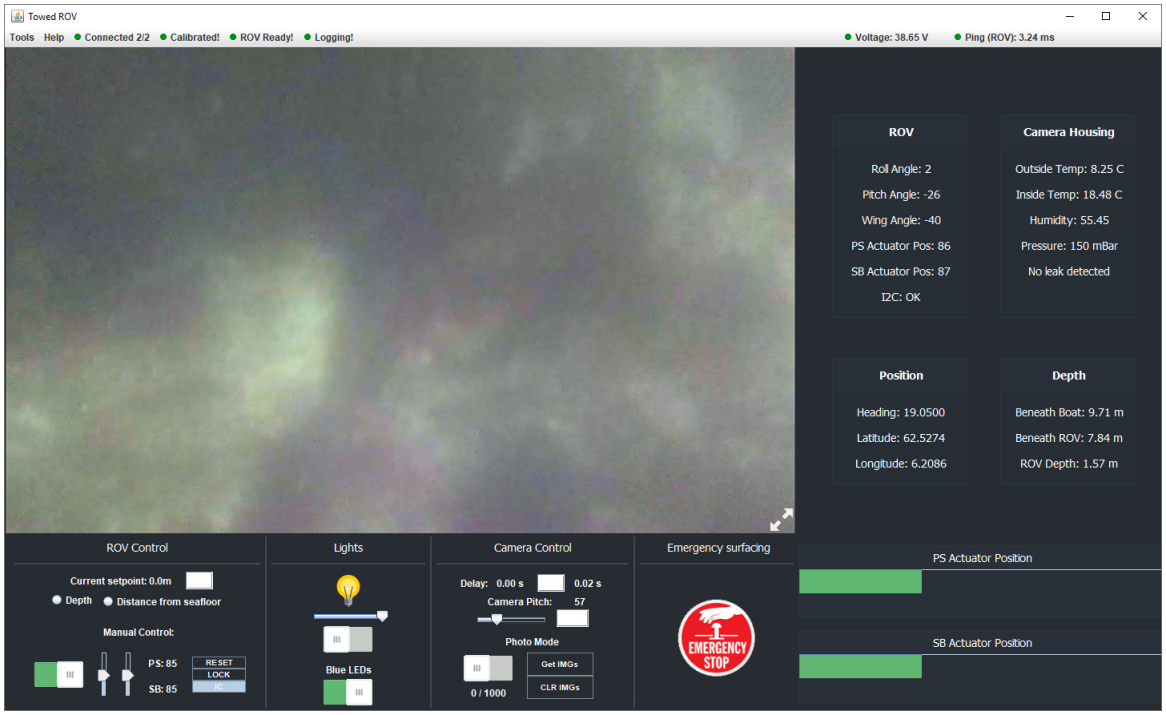


Figure 4.3: The graphical user interface

Figure (4.3) shows the final GUI in operation. One of our main goals was to build a lucidly and user-friendly GUI. We achieved this by focusing on only displaying the most necessary controls and data in the main window, and placing other tools and functionality in drop-down menus at the top left corner. The GUI fulfilled our requirements when it comes to the set objectives of this thesis.

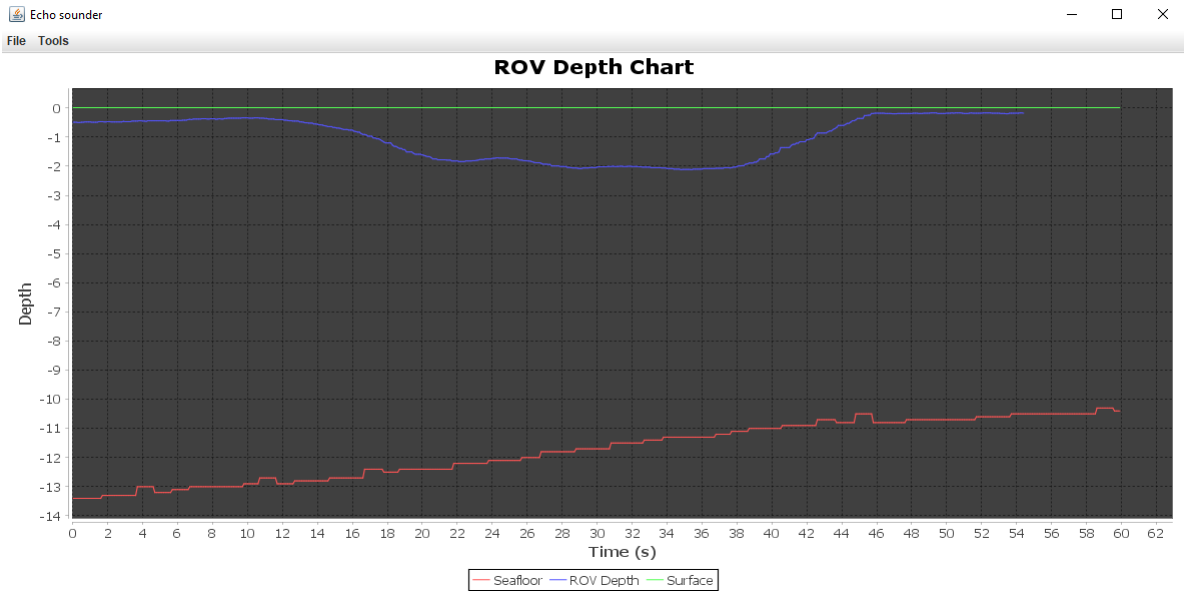


Figure 4.4: The live depth graph

From the Tools menu, a live graph which plots the depth beneath the ROV, and the ROV depth can be opened. This has been improved from the previous project.

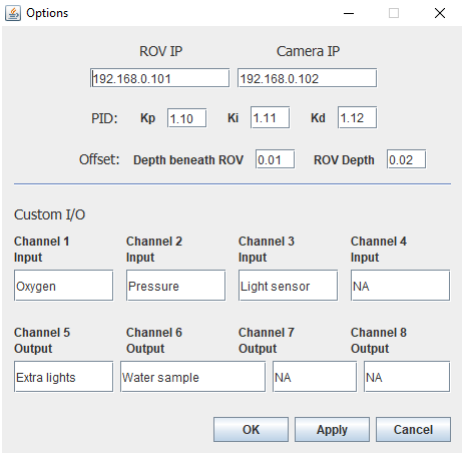


Figure 4.5: The Options window of the GUI

The options window of the GUI has been expanded from the previous project. Since we use two Raspberry Pis, we added the option for changing the default IP addresses of those. We also added the option for setting the PID values and the offset for calibrating the depth sensors. These values are sent to the main RPi controller when the apply/OK button is pressed. The rest of the options window for changing the labels of the custom I/O was left unchanged.

Figure 4.6: The I/O Controller window of the GUI

The window shown in figure 4.6 is almost similar to the original design from the last year's thesis. The only thing we did was to change the colors and images to a more modern look and feel.

4.1.3 Surface vessel

On the surface vessel, there are multiple key components used, in order to make this prototype work properly. In this section, a description of the various systems implemented on the surface vessel will be described.

Battery case

Because of the weight of the batteries, which was a total of 60 Kg, we made a case to place the batteries in, in order to transport the batteries supplying the instruments on the surface vessel and ROV easily. By doing this, we could have the batteries connected to each other at all times, and also quickly connect to the suitcase on the surface vessel. There are three batteries in total to supply the components on the surface vessel and the ROV. The batteries are connected in series, which gives us 36V DC and 75Ah. We also extract 12V DC from one of the batteries.

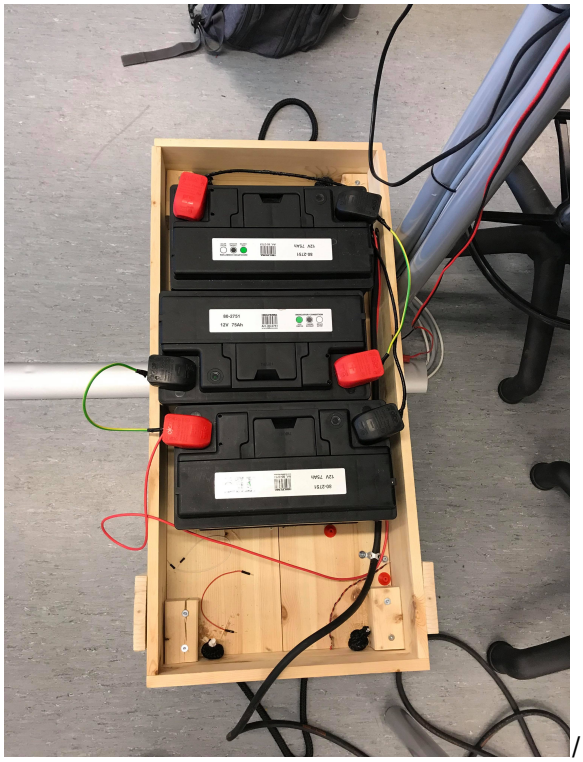


Figure 4.7: Battery case.

Base station suitcase

On the surface vessel, we have made a suitcase for quick and easy connection. On the outside of the suitcase, there are three connectors marked by color, to connect batteries, ROV, and the echo sounder in the surface vessel. There is also a USB connection and a Ethernet connection. On the inside of the suitcase, there is two RedBoards, a USB hub, a Razor 9DoF IMU, an Adafruit ultimate GPS, a Fathom-X tether interface, and an optocoupler card for the connection between the echo sounder and one of the Arduino Uno cards. By doing this, we can easily connect a USB cable to read values from the Arduinos, and also connect to the Ethernet port to read data from the ROV. Within the suitcase a Plexiglass pane has been shaped in the form of the suitcase. This would give the operator a place to put the computer while testing, and thus removing the need of a table.

Figure 4.8: Suitcase.

Figure 4.9: Connectors on the left side of the suitcase.

Different components in the case have a 3D printed part holding them in place, mainly to avoid using screws directly into the suitcase floor and to prevent using glue directly on the component. We also did not want the parts to move freely inside the suitcase, as this could result in a short circuiting. Therefore the 3D printed parts were glued in the suitcase, and if a change was necessary, we could replace the component without any issues.

Surface vessel echo sounder

The echo sounder on the surface vessel is used to measure depth below the vessel. Mounting it to the boat was done with a two meter long aluminum profile. On the profile there is fitted two pipe clamps that are fastened to the railing on the side of the boat. On the profile, there are pre-cut groves so we could easily slide the clamps up and down depending on the size of the boat. A 3D printed model was designed to hold the echo sounder to an aluminum profile, see figure (4.11).

Figure 4.10: Echo sounder.

Figure 4.11: Echo sounder Frame.

4.2 Test Result

4.2.1 Test on land

We wanted the system to work properly before we filled it with oil and took it out to sea. Here we tested the actuators, video stream, gimbal, and data transfer. We wanted to make sure everything worked properly before we tested the ROV in the water, to avoid any major problems while out on the sea.

All working functions were tested separately before everything was tested together. Communication was tested with the video stream and transferring images from the FTP server. The actuators were tested in manual mode with a controller, the joystick on the controller sent a value, and the actuators moved to the correct location. For the echo sounders, we used a tube filled with water to see if it gave us feasible results. Everything for testing the ROV in manual mode was working, but we had some issues which will be addressed in the following sections.

Optimistic power calculations

When testing the actuators on full speed while the lights was set to max intensity and all other parts of the ROV was powered on, we experienced some issues with the power. Sometimes the Raspberry Pi would reset, or the lights would flash. Firstly, we suspected that the capacitors on the main circuit board drew to much power to get filled up, resulting in power loss and the flashing. Since the batteries and cable were used from the previous thesis, we expected that the calculations that were done, were correct. *"The actuators could use up to 3A each, and the echo sounder 5A. This was the worst case scenario and was accounted for."* (see page 81 in TowedROV 2018). Since their worst case scenario was 11A, and they had taken that into account for the battery and cable choice, we should be able to use the exact same calculations. As we can see in section (3.9.3), the realistic power consumption can be up to 8.6A on our ROV. This is well under their worst-case scenario. However, since we could not find the underlying cause for the blackouts, we started to do our own calculations. If we say that the total load is 11A and the resistance in the cable is 6.1Ω we can use equation (2.3). This will give us a voltage drop of 67.1V, a much higher result than the original calculations. Since we have 36V supplying the ROV and the transformer can handle voltages as low as 18V, we can max have a voltage drop of 17V. The equation (2.3) gives us then 2.78A. This is too little capacity to run the ROV efficiently. This explains why we have been struggling with random blackouts and being unable to use the large capacitors we originally planned to use. There is no quick fix for this problem, especially when we discovered it so late in the project. The cable is under dimensioned, and we would need a power source of 86.1V to reliably supply the ROV with 11A. Our temporary solution was to lower the max speed of the actuators to 20% of max speed and lower the external lights to its lowest possible setting. By doing this, we could run the ROV as planned just a bit slower.

Overheating

One major problem we encountered was loss of the 5V supply. Losing the 5V results in the Raspberry Pi turning off and we lose all connection to the ROV. This was a problem we had to investigate before the sea trial. We noticed that the main electronic box was hot while the ROV was assembled without oil. Therefore we logged the internal temperature.

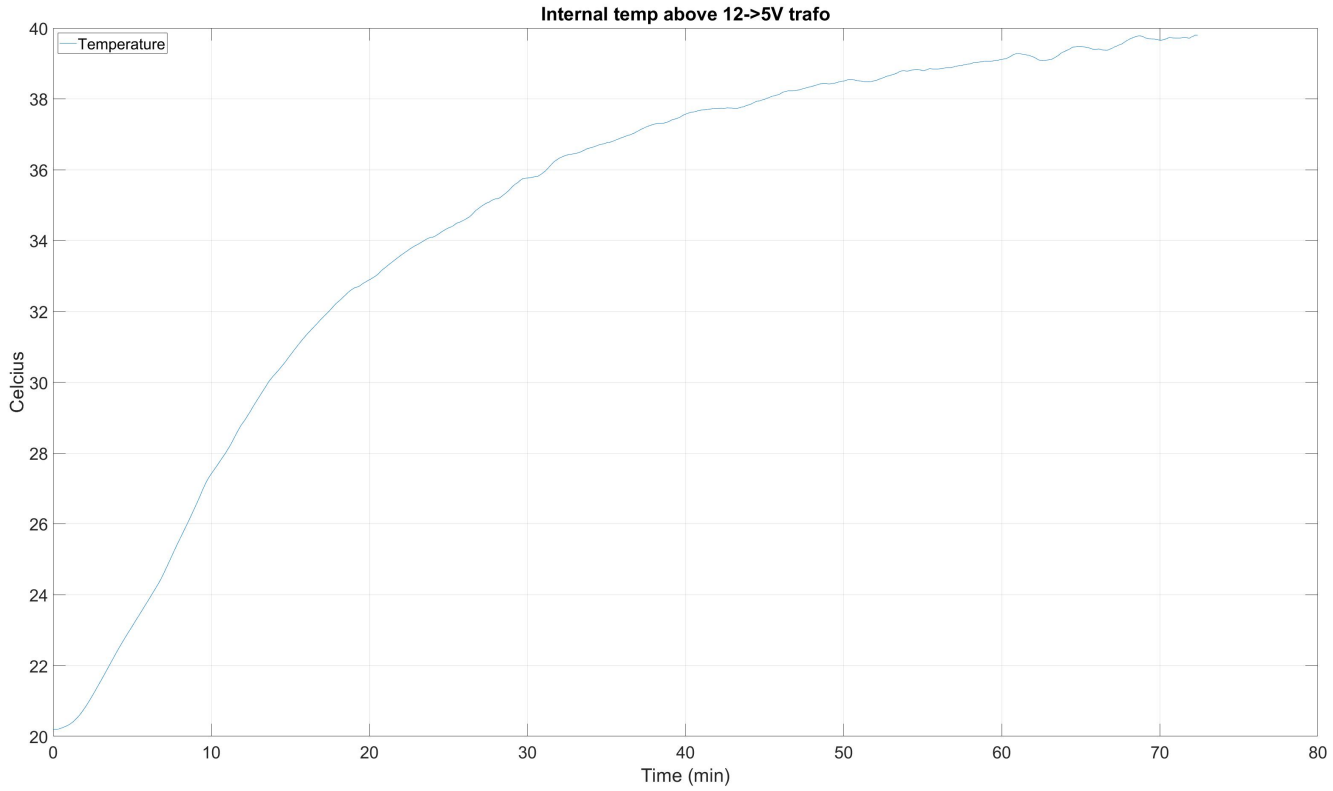


Figure 4.12: Internal temp above 12-5V transformer

We logged the temperature for 80 minutes when the ROV was connected to the power supply and all the internals systems running. After 80 minutes, the temperature had risen from 20 degrees to 40 degrees. This should not be a problem since the temperature limit for the 12V to 5V transformer is 85 degrees Celsius. Although, it is important to note that the measured temperature is not the internal temperature of the transformer but the surface temperature. Therefore we concluded that after close to 9 hours of continuously testing, the temperature limit of the transformers was reached. Since the main electronic box is filled with oil and the whole box is submerged under water when operating, and the continuous operating time of the ROV never will surpass 9 hours, the group concluded that this would not be a problem.

GPS test

The results from the GPS shows that the location is correct. Where the longitude and latitude display the correct values. The GPS took some time to get a GPS lock. After the Arduino had read the data from the GPS, it was then converted into a string.

When testing the GPS logging, we noticed that the GPS had a very long start-up time before a GPS lock was achieved, over 1 minute in the worst cases. This was fixed by adding a small battery in the GPS. The battery circuit on the GPS is designed to keep the RTC data. This means the GPS does not have to wait for satellite clock data but can start getting a location immediately. This resulted in a consistent startup time that was no longer than 20 seconds. After this, we placed the tracking equipment in a car and drove around the university to check the data.

Figure 4.13: Log from car test

As shown in figure (4.13), the GPS data gathered proved to be precise.

Echo sounder test

The echo sounder were tested in a 1.4m high tube filled with water before installation. We use the same logic for converting it to a string as the GPS (4.2.1). While testing in the tube, the depth showed a value of 1.2m. This is a 20 cm error, which was acceptable for our use. Other data from the echo sounder such as temperature and speed also showed to be working, but since we were not going to use this data, these values were removed from the data string.

$SDDPT,1.2,*7A
Depth: 1.20
Transducer offset: 7.00
$SDDBT,4.1,f,1.2,M,0.6,F*06
Depth below Transducer: 1.20
$VWVHW,,T,,M,3.55,N,6.56,K*52
Speed in Knots: 3.55
Speed in Km/h: 6.56
$YXMTW,24.6,C*12
Temperature in celcius: 24.60

Echo sounder NMEA test results.

4.2.2 Sea trial

Figure 4.14: The GPS data from the sea trial - red line.

The complete test at sea was a milestone in the project. This test was done around the island Store Kalvøy, an island located 15 minutes from Ålesund. We chose this location because one group member grew up on this island and was familiar with the waters around it. We also used his father's boat, an Askeladden P66 Weekend. This boat is big enough for three people to operate the ROV and has enough motor power to tow the ROV and maintain precise steering. We arrived at 10:45 on the island and started the final assembly. This included replenishing the boxes with oil, mounting the top buoyancy pipes, and visual inspection of the connections and screws. After this was done the ROV was placed in shallow waters, which made it possible to test the floating capabilities and hydrodynamics. This was done by manually towing the ROV while one group member was swimming beside to study its behaviour. After the buoyancy was adjusted and the group were pleased with the manual tests, we sailed to deeper water and continued with a proper towing test. During the tests, we looked at the behavior of the ROV, the floating capabilities, and sensor data.

Actuators

The actuators worked properly at the start of the testing day. However, as we suspected earlier (3.14.1), their performance was slowly getting worse. The oil had reached the brushes on the starboard actuator, and at the end of the day it was barely moving. This affected the remaining tests we needed to do. The oil also affected the feedback from the actuators. Since the feedback is based on a potentiometer, we suspect that the oil has penetrated the potentiometer and therefore increased the resistance leading to wrong feedback. For further testing, we decided to turn them to diving position and lock them in that position in the software.

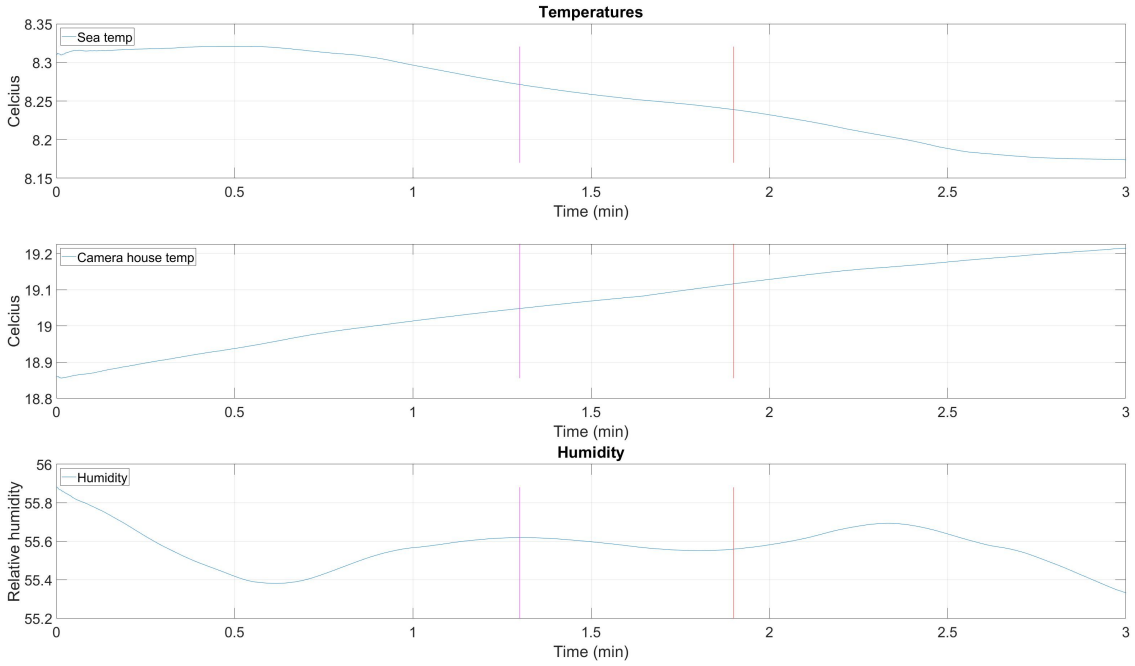


Figure 4.15: Temperatures

Too much buoyancy

The ROV began tilting when it was placed in water for the first time. This was because of the pipes we had used for buoyancy (3.10.9). The bottom pipes (3.27) held the whole weight of the ROV. These were calculated to make it a bit lighter in the water, while the ones on the top would make sure that it was floating. The problem was that our original calculations were wrong, new calculations were done after the completed test, this to see how far off we were. The first calculations of the boxes were done manually, and we were off with 6.3 Kg. The first calculations

showed that the ROV had approximately 19 kg of buoyancy. After we used Fusion 360 to measure the volume to be 21,213 $\times 10^6$ mm^3, using equation (2.1), we calculated that the buoyancy of the boxes was close to 25 Kg. The bottom pipes itself lifted as much as 25.04 Kg. Which was a total of 50 Kg, which is 4kg more than the ROV weight, and therefore, the ROV would float before the top buoyancy pipes reached the water. In order to get the wings under water, the bottom pipes had to be removed. The ROV floated just as intended when the bottom pipes were removed.

As mentioned in section (3.10.9) was the top pipes designed to lift a bit more than 2.8 Kg, since the ROV was still positively buoyant after the bottom pipes were removed, new calculations were done of the top pipes. The volume of the pipes were 14.875 $\times 10^6$ mm^3, which then gives a buoyancy of 15 Kg. The huge difference between the first calculation and the second, was because how Fusion 360 selected the pipes when we analyzed the volume. The first time we did it, not all the pipes where selected and therefore Fusion 360 only gave us the volume of the smallest top pipe.

Figure 4.16: Buoyancy before removal of bottom pipes.

Figure 4.16 shows that we had to hold the ROV in place, in order to prevent it from tilting over to its side.

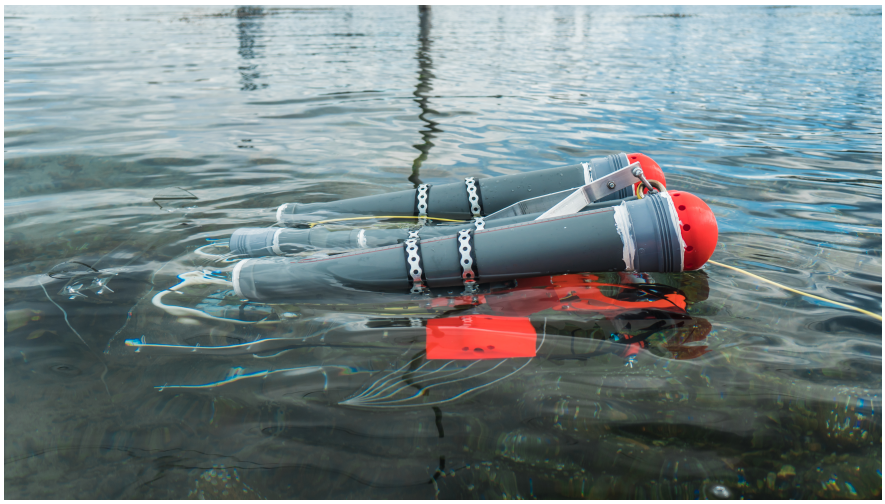


Figure 4.17: Buoyancy after removal of bottom pipes.

Figure 4.17 shows how the ROV floated in the water after we removed the botom pipes.

Condensation and high humidity

The leakage sensors were activated after some time in the water. This was because of the high humidity inside. We inspected the housing after the alarm of leakage was activated. While inspecting, we saw no signs of water. Because of this and the humidity inside the housing resulted in the leak sensor to activate. Since no water was detected, we left it as it were and continued testing.

ROV behavior when diving

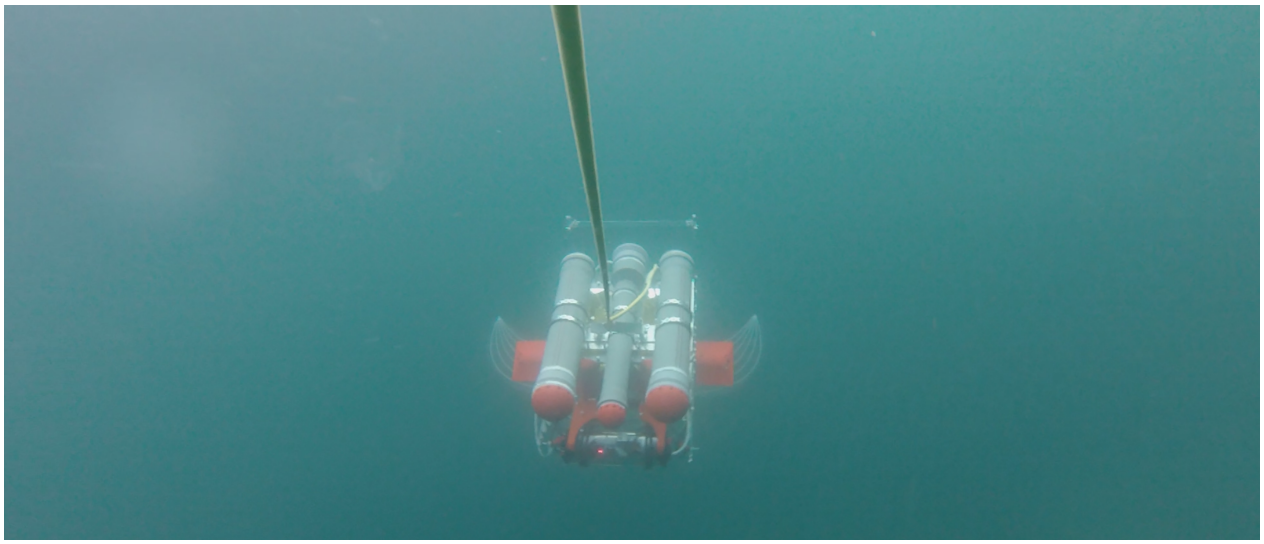


Figure 4.18: The ROV diving two meters below the surface

When we towed the ROV along the surface, it was very stable. There were no oscillations in the ROV, and it sailed smoothly through the water. We had to adjust the dive speed and depth with the speed of the boat since the wings were locked in diving position. When the boat had around one knot, the ROV was towed just in the surface. When the speed rose to between two and three knots, the ROV dived. We saw that the ROV dived smoothly. We had a theory that when the ROV dived it could pitch too much down and flip around, especially when the fins were locked in dive position and cannot compensate for pitching to much down. As we can see from figure (4.19 and 4.20) the magenta line represents when the ROV is ascending, and the red line represents a dive. Here we can see that when the ROV is diving the roll is very stable and it is pitching 51 degrees downwards. This is a bit to much pitch and should be limited by the wings, but as mentioned before, they are locked in place. When the ROV is rising the roll is unstable, up to 29 degrees. This is because we are regulating the diving with the speed of the boat. To get the ROV to rise we are slowing down the boat, but the wings are still in dive position. The result of this is that the ROV is stalling the same way an aircraft do, when the speed is too low. There is nothing to control its upward movement except its buoyancy. This would not be a problem if the boat had a constant speed, and the actuator's PID controller was activated.

Raw data from diving shows that the ROV dived 1.54 meter in 8 seconds when the speed of the boat was between 1 and 2.4 knots. This is equal to a diving speed of 0.192 meters per second. It is important that the rising and diving speed is not too high, as this would add unnecessary

strain to the ROV and the utility cable.

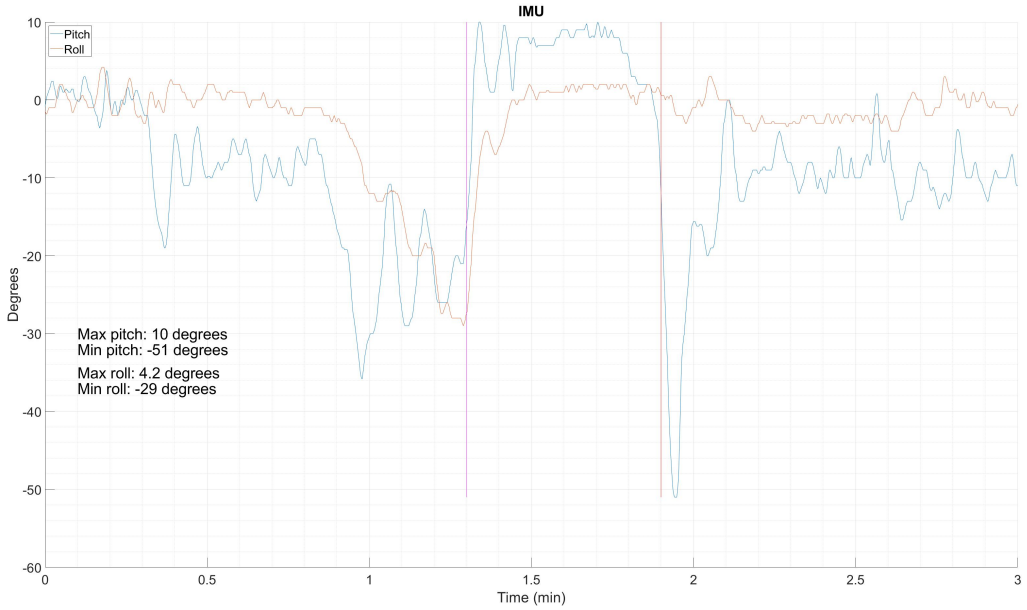


Figure 4.19: IMU data

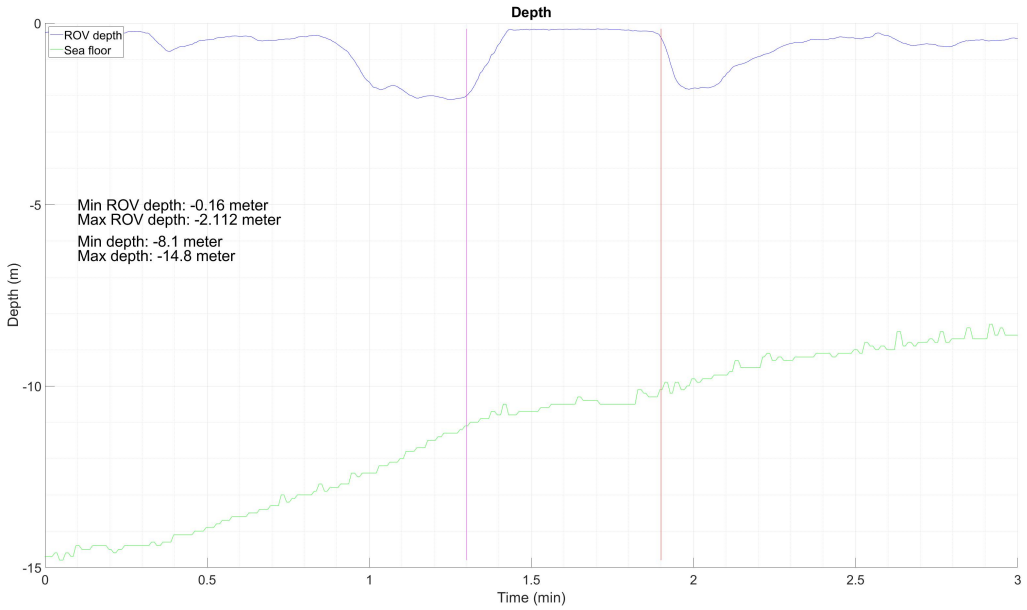


Figure 4.20: Depth data

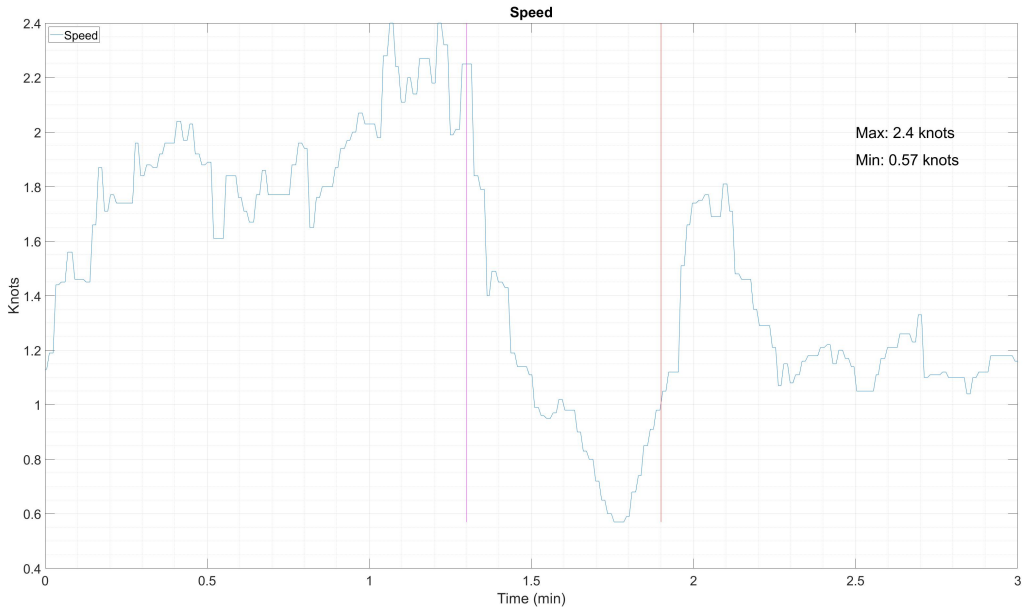


Figure 4.21: Speed data

As we can see from the normal distribution diagram (4.22 and 4.23), the ROV is only rolling about -4 degrees over the span of 3 minutes. This shows that the roll stability on the ROV is good. The normal distribution on the pitch is around -10 degrees. This does not give us that much valuable information since we were constantly diving and rising and the wings were locked in dive position. To get good data from the pitch, we need to use the wings to stabilize the ROV at a certain depth and hold it there over a longer period of time. When doing this, the normal distribution should be around 0 degrees.

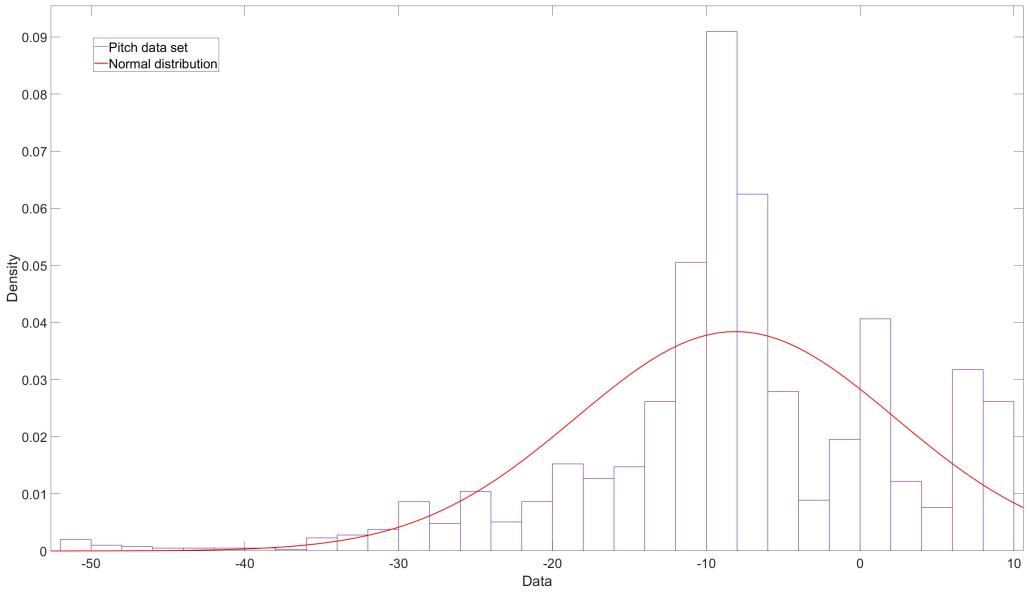


Figure 4.22: Normal distribution pitch

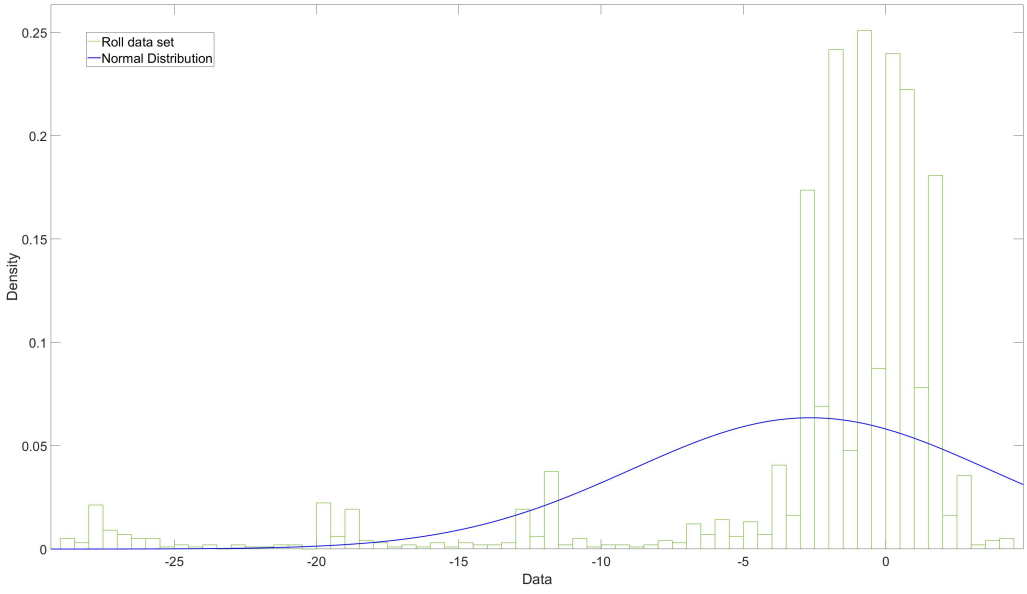


Figure 4.23: Normal distribution roll

4.3 Completed goals

The must-, should- and could-have list from the preliminary report has been our main goal to fulfill. Since we had to build the ROV from scratch, a lot of time went into building instead of working on the list. As we can see from table (4.1), the green tasks are done. 66.6% (not included the scrapped magenta ideas) of our goals has been completely reached. Based on the amount of time we have used and the size of the whole bachelor thesis (D), we are pleased with how much we were able to do and get working.

The orange tasks are started on and almost done, all necessary code and data are available in the source code. With further work, this can easily be implemented.

The magenta tasks are tasks we have discarded along the way. An electrical housing that is not filled with oil would be too expensive for this project. We did not find it suitable to implement FPV-goggles in a prototype at this stage. The auto adjustment of the lights was decided to not be implemented since it would be more bothersome than useful for the user. The GUI contains a slider to adjust the brightness on the lights manually.

Must have	Should have	Could have
Functioning and user-friendly GUI	Anti-collision system that can detect sandbanks/hills	FPV-googles to watch live feed
A control system that can keep the ROV at level, dive and rise.	Sensor and signalling rod on ship	Heater to prevent moisture and condesation in camera house
Live Video Stream	Manual Control of ROV	Auto adjust external lights based on video feed
Gimbal on camera with manual controls	Calculate approximate GPS position of ROV	
An electrical housing that is not filled with oil	Alarm system (detect leak, moisture, etc.)	
Live PID calibration in GUI	A plug and play system on all to ease the use of the ROV	

Table 4.1: Result of must-, should- and could-have list

4.4 Operating setup

In order to test the system, a computer with the GUI application is needed. The software can be downloaded at: (https://github.com/Roklo/TowedROV_GUI).

To test the ROV, follow the instructions below:

1. Fasten the utility cable to the surface vessel.
2. Fasten mobile echo sounder device to the surface vessel.
3. Start client computer.
4. Attach battery cable to the suitcase. RED connector to RED plug on the suitcase.
5. Attach surface vessel echo sounder to the suitcase. BLUE connector to BLUE plug on the suitcase.
6. Attach ROV cable to suitcase. YELLOW connector to YELLOW plug on the suitcase.
7. Connect the operator computer to the suitcase with USB and Ethernet cable.
8. Set static IP Address on operator computer to 192.168.0.20.
9. Start client application.
10. Wait for the blue lights on ROV to flash once, for the Raspberry Pi to boot and sensors to calibrate, and listen for three beeps from the gimbal controller.
11. Click "Connect" in the upper left corner of the GUI to connect to both the camera RPi and ROV RPi.
12. Deploy ROV to the water surface.

4.5 Additional photos

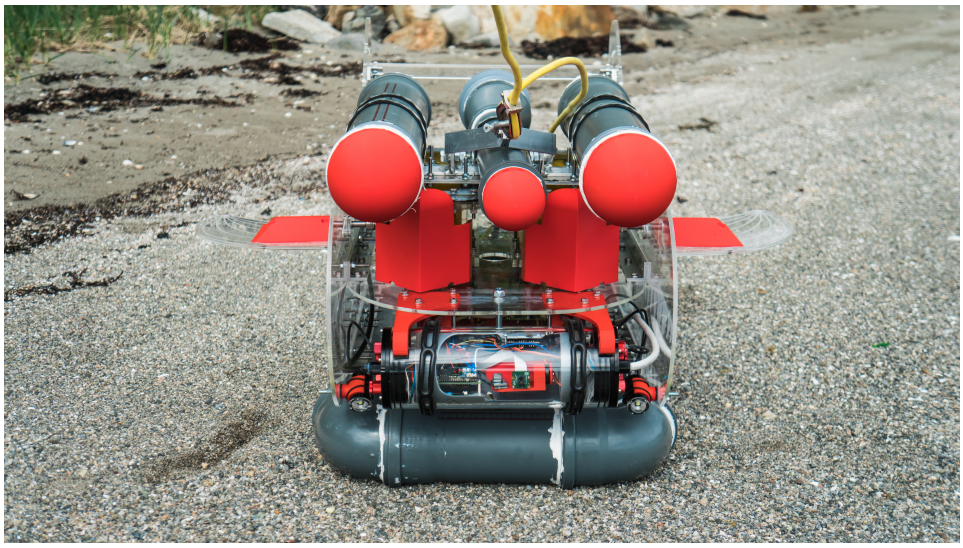


Figure 4.24: The front of the ROV.

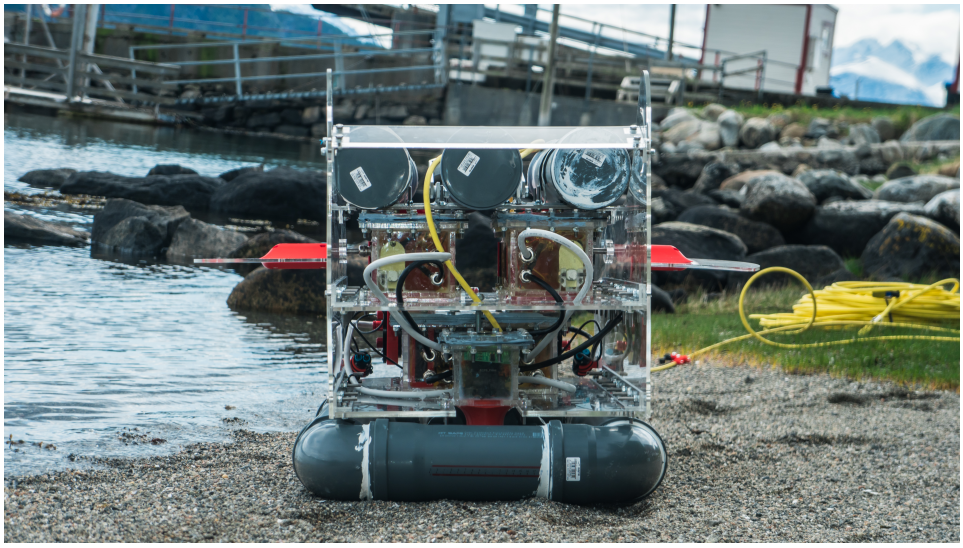


Figure 4.25: The rear of the ROV.

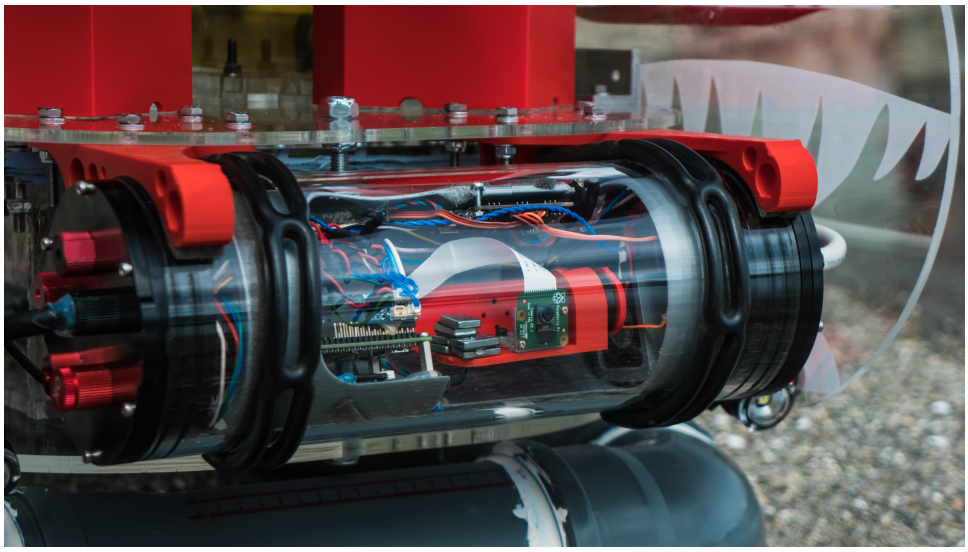


Figure 4.26: The camera housing.

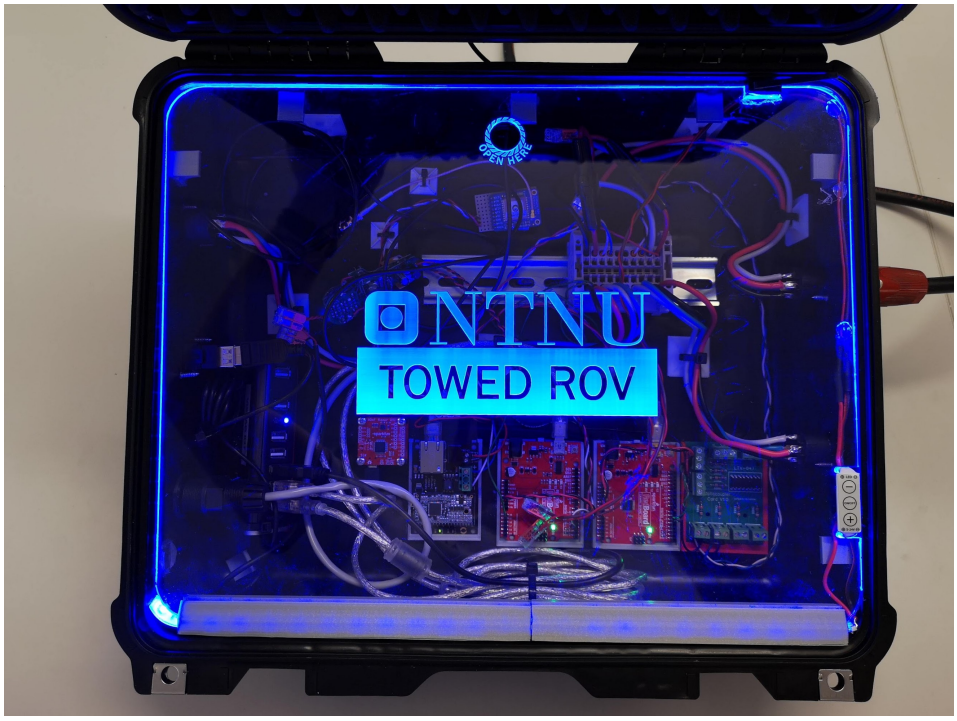


Figure 4.27: The suitcase.

Figure 4.28: Testing of the ROV in the sea.

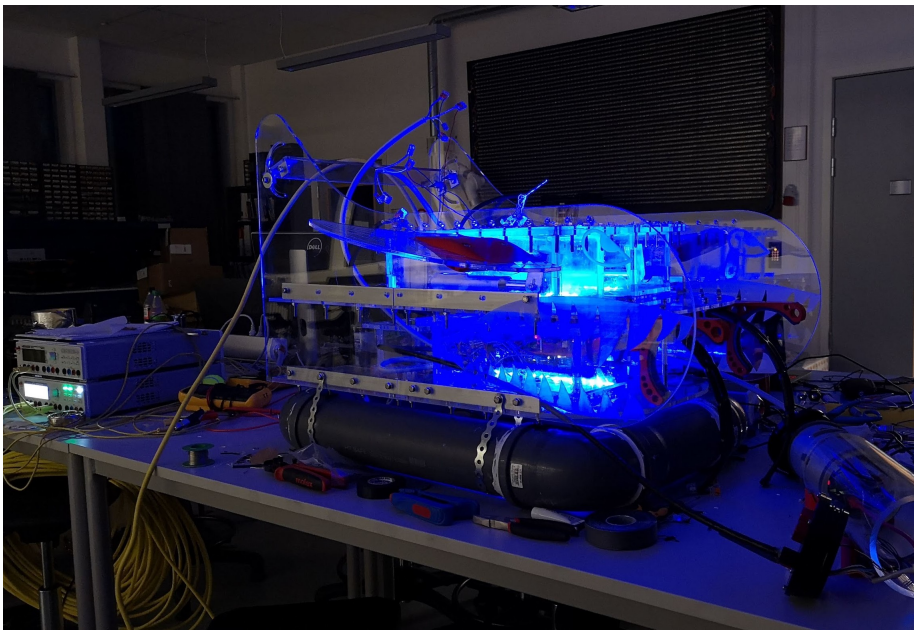


Figure 4.29: Testing of the blue LEDs.

Chapter 5

Discussion

5.1 Prototype

The prototype is both wider and longer compared to the old ROV design. We did this to increase the stability, which was proved a success while testing it as described in (4.2.2). The result of this proves that the design is a good solution. After a full day of testing, there were no signs of leakage proving that the boxes were sealed properly, and the Plexiglas was a viable material to use for prototyping.

Because of the shape of the boxes, many flat areas facing in the towing direction would cause a lot of drag. By adding the 3D printed nose cones in front of the top buoyancy pipes, and the water deflectors in front of the actuator boxes, it helped to reduce the drag. As a prototype, the current design is good enough for further testing.

A problem with the Plexiglas is microcracks when tightening the bolts. Even though it works for prototyping, it is not recommended to be used in a finished product. In the areas where microcracks have happened, it has the potential to easily crack even further, and therefore increasing the risk of water leaking in. This can be avoided with a material that is stronger, preferably a material which does not easily crack when tightening the bolts.

5.2 Hardware

We had problems with the external lights blinking when they were running at their lowest setpoint. An afterthought on this is that the 18V-75V -> 12V transformers are very inefficient on low loads as we can see in figure (3.15). Under 1A, the transformer has an effectiveness of only 70%. Since the transformer is not designed for such small loads, this may explain the blinking. If this is correct, it will be important to remember this if the voltage down to the ROV is raised. We did not have time to test this, so this is only a theory.

Otherwise, we are pleased with how the hardware aspect of the project was done. Making our own PCB cards was extremely helpful as it lowered the amount of cables needed in the ROV, making it less prone to human error when connecting the components. The PCB cards also helped us make the system more straightforward to use and understand. The drawback of using these PCB cards, is that if an error that has been made while designing is not discovered until after it has been ordered, it may be hard or in worst case impossible to fix without ordering new ones. Because of the long delivery time, this might become a big waste of time.

We are also pleased with the plug and play system. As it simplified the final solution for connecting the ROV to the surface vessel, it removed the need of connecting the wires each time we would test the system. This way our batteries could be connected to each other at all times, and supply the ROV with power by only connecting the batteries and utility cable to the suitcase.

5.3 Software

The software that has been developed throughout this project has been one of the main tasks. It has been in constant development since the start and has resulted in an improved, responsive, lucidly and more user-friendly GUI, together with software for reading various kinds of sensors, a fast and reliable video stream, and a satisfying data transfer system. Of course, there is always room for improvements in such a big project, such as removing public variables and increasing thread safety. We did not have any issues with our way of securing threads, but we believe that by implementing semaphores, we could achieve a higher level of thread safety. By implementing thread executors, it enabled us to set fixed rates for each running thread. This made our system less heavy for the processor and enabled us to control the speed of each individual thread.

Our way of sending text strings over TCP proved to work very well and showed to be a stable way of sending data. For updating the GUI we used an observing technique when a new value had been received. This worked as intended, but as it is now, it always updates the whole frame of the GUI instead of just the one variable that changed. Especially when receiving the data string from the ROV with nine different variables to be set - which is received every 100ms - the GUI then has to update the all of its elements for each of those variables. That is 90 times each second. It is not a problem, but it may be something that can be improved.

The development of the GUI in Swing made it easy for us to connect the back-end code to the front-end. On the other hand, the Swing GUI builder in Netbeans is not that great when it comes

to placing elements where you want them to be, often moving other objects to undesired locations. It was therefore very time consuming to make the GUI resizable for any kind of monitor. With our experience, it would be much easier to build the GUI in JavaFX when it comes to the structure of the elements, but then again, in JavaFX, it is not as easy to connect the back-end to the front-end as in Swing.

5.4 Test results

The dry test was conducted at the lab, which involved individual testing of each component in our system. The testing was done thoroughly to ensure that no problems were encountered while testing at sea. This ensured us that the whole system worked properly when the ROV was completed.

The software and communication were also tested during this process. If we encountered a problem while testing, we resolved them and moved on. These tests were also done multiple times to check it is stability, and to check if there was something that we missed. By doing the testing along the way, it reduced the amount of assembly and disassembling when we found problems.

The boat we used for testing was an Askeladden P66 weekend. The boat had good space to work with while testing the ROV, which was absolutely needed. All the equipment plus three persons took up more space than anticipated. A problem with this boat is its high railings. We would not be able to safely launch and recover the ROV from the boat because of the weight and shape of the ROV. Because of this, we had to tow it from the shore to the testing site and back. A launch and recovery system would have made this a lot easier, but designing such a system is hard since it should be modular to fit multiple types of boats.

Because we had to lock the actuators in diving position, we were unable to test the depth control system, but we were still able to test the diving abilities by actively changing the speed of the boat. The test results of this were good, but it was not good enough to evaluate the complete ROV. Although, it is adequate to determine that the hydrodynamics work and that the wing size is sufficient for it to operate.

To prevent condensation in the camera housing a heating element and a circulation fan was added. The camera that is used in the ROV turned out to be not good enough for operating in low light scenarios like the ocean. When underwater, the video stream is completely green from

all the algae in the ocean, and when it was close to the sea floor the details in the picture was low, and the field of view was too narrow. Because of this, it was hard to estimate the actual distance from the seafloor or to see anything of interest. The video we recorded with a GoPro of the ROV underwater was crystal clear, and did not struggle with the low light conditions. A better camera would be preferred to resolve this issue.

5.5 Areas for improvement

Collision prevention system

Getting the collision prevention system to work is a crucial part of a towed ROV. With the data and system that is implemented in the ROV already, it is possible to make such a system. With the help of the mobile echo sounder mounted on the boat, it is therefore able to detect possible collision points before the ROV reaches them. By using this data and in combination with the max rising speed of the ROV, an advanced system can be made to avoid steep hills in time.

Change actuators

As mentioned before, the actuators are not well suited to be used in oil. Over time will the oil get inside the actuator, and the brushed DC motor. When this happens, the oil will work as an insulator between the brushes and commutator, resulting in poor performance, and over time, a complete stop. The feedback potentiometer will also be filled with oil and might give wrong feedback. A solution to this would be to replace the actuator with stepper motors. This removes the need for a mechanical feedback system, and there are no brushes that will be insulated by the oil. However, since the stepper motors are not able to hold the wings in place by themselves, it will be important to either get stepper motors that has a gear system built in, or design a screw gear system.

Higher voltage

Since the yellow utility cable is under-dimensioned and cannot supply the ROV with enough current, a solution would be to raise the supply voltage. As mentioned in section (3.11.1) the ROV is capable of handling up to 75V. Because of this, we could remove one or two of the bat-

teries we have and use a transformer that transforms the voltage up to 70V. By doing this, the voltage drop would not affect the ROV (2.3), and the battery pack would be lighter.

Laptop charging capabilities

When operating the ROV out in the field, the operating computer will need charging long before the ROV does. A solution to this would be to get a 12V -> 230V transformer and connect it to the ROV's battery pack.

5.6 Possible areas of utility

Areas for utility for a finished version of a "Towed ROV" might be

- Explore new areas of the sea floor.
- Surveying of offshore installations, such as pipelines.
- Maritime search and rescue.
- Scientific research.

5.7 Prototype to finished product

What separates the prototype from a finished product are presented in this section.

Camera

The RPi Camera we used is not the most suited camera for capturing areas with low lighting. A camera that is better in darker areas would be better. Since the camera housing is not filled with oil like the other boxes, it is easy to open it up to replace the camera. The motors for the gimbal we designed are supposed to hold cameras up to approximately 150g, which should be enough for a decent camera like a GoPro.

Power supply

A better solution for charging the batteries, or a power supply directly from the surface vessel should be implemented. Instead of docking to recharge the batteries one by one, it should be considered to charge the batteries with for example a diesel generator. A solution for bigger boats could be to use the power directly from the boat. This would remove the need to recharge any batteries.

Automatic pitch trim adjustment

The spoiler on the ROV is static and has to be manually adjusted. This spoiler is supposed to help with the pitch trim of the ROV when being towed. The towing bracket that is connected to the utility cable is in the front of the ROV, meaning that the ROV will tilt a bit forward when it is being towed. To adjust the pitch of the ROV, we would then adjust the spoiler, test, and then adjust it again if necessary. To remove this problem, it should be implemented a motor that is adjusting the spoiler while it is being towed.

5.8 Challenges with a towed-ROV

There are several challenges with a towed ROV. One of the challenges is objects that have not been detected by the echo sounders, which might end up with the ROV colliding or the utility cable snapping. Another challenge is when the boat is turning too hard, the cable between the boat and the ROV will become slack, and the ROV will start to rise to the surface. These challenges can be greatly reduced with the correct measures. Challenges like this do not prevent the concept of a towed ROV from being useful, but should be accounted for.

5.9 Experiences from the project

Work distribution

The group has similar backgrounds but with different interests. These different interests have been used to distribute the different tasks during the project. This has saved us a considerable amount of time, and it has allowed us to execute such a diverse project.

Project plan

During the start of the pre-project phase, we created a list of demands for the ROV, see table (1.1). On the must-have we have completed the user-friendly GUI, the control system works properly, we have a live video stream, the gimbal works with manual control and calibration of the PID in GUI.

On the should-have list we have implemented a sensor and signalling rod on the ship, manual control of the ROV, an alarm system that detect leaks moisture pressure and temperature, and a plug and play system to ease the use of the ROV.

The could-have list has only one item completed, which is the heater to prevent condensation within the camera housing.

Gannt diagram

A gannt diagram has been used to plan the necessary tasks during the project. The group tried to estimate how much time each task would take and created the diagram early in the project. Many of the tasks were completed within the deadline, and some were delayed. This was compensated with tasks completed early and extra work hours if necessary.

Group size

There was a lot of work for only three persons. In such a diverse project as this, it would be better with one or two more persons. Primarily students from Product and System Design (POD). This way, we could focus on hardware, electronics, software and logic, while the POD students could focus on design, 3D modeling, hydrodynamics, and buoyancy.

Chapter 6

Conclusions

In the preliminary report, we created a list of specifications we meant were necessary to build a functioning prototype of a towed ROV. Almost all of these were completed, and the group are satisfied with the prototype, which fulfills the concept of a functional towed ROV. This prototype works as a multipurpose platform, that should be able to control its depth automatically. With this ROV, we are also able to stream live video, and control the ROV through a lucidly graphical user interface. It was also a demand that the prototype should be based on a modular design, which the group believe they have achieved; it shows that it is possible to develop a towed ROV that uses affordable materials.

The project has given the group valuable experience in planning and executing projects as big and diverse as this. In this project, we linked many of the different subjects during our three years of engineering studies, to an entirety. During the project phase, the group has developed new experiences in subjects that are not related to the automation study, this because of the projects multidisciplinary.

The topic of this thesis was to study the concept of a towed ROV, and develop a functioning prototype with software, which includes a control system to further study the concept. Based on the results and research in this thesis, the group experiences the project as a success, and therefore concludes that the concept of a towed ROV is a valid project for further development.

6.1 Further development

Based on the work done, we see that there are several areas for further improvement.

- A better suited camera. Because of the bad light conditions under water, and since the Raspberry Pi camera is not well suited in these kind of environments, it would then be wise to replace this camera with a camera that is better suited for dark areas.
- Trying out different materials for housings, preferably in aluminum, because of its strength-to-weight ratio.
- Simplify assembly.
- Easier and faster solution for waterproofing. The Plexiglas boxes took a long time to finish; welded boxes would be preferred for waterproofing.
- Automatic adjustment of the rear spoiler. A motor could adjust the spoiler on the ROV to adjust the pitch while towing the ROV.
- Implementing a collision prevention system.

Bibliography

[1] Finn Haugen (2010). Ziegler-nichols' closed-loop method. techteach. http://techteach.no/publications/articles/zn_closed_loop_method/zn_closed_loop_method.pdf.

[2] Atlassian. Types of version control. https://confluence.atlassian.com/get-started-with-bitbucket/types-of-version-control-856845192.html.

[3] BlueRobotics. Bar30 high-resolution 300m depth/pressure sensor. https://www.bluerobotics.com/store/electronics/bar30-sensor-r1/.

[4] Halvor Bothner-By. Tcm – nettverksprotokoll. https://snl.no/TCM_-_nettverksprotokoll.

[5] Robert D Christ and Robert L Wernli Sr. *The ROV manual: a user guide for remotely operated vehicles*. Butterworth-Heinemann, 2013.

[6] EE|Times. What is power line communication. https://www.eetimes.com/document.asp?doc_id=127901

[7] Starting Electronics. Measuring dc voltage using arduino. https://startingelectronics.org/articles/arduino/measuring-voltage-with-arduino/.

[8] Raspberry Pi Foundation. Camera module v2. https://www.raspberrypi.org/products/camera-module-v2/.

[9] Raspberry Pi Foundation. Raspberr pi model 3b+. https://www.raspberrypi.org/products/raspberry-pi-3-model-b-plus/.

[10] Garmin. Virb edit for windows software version 5.4.3. https://www8.garmin.com/support/download_details.jsp?id=6591.

[11] Gearbeast. Gladius underwater drone rov. https://www.gearbest.com/r-c-boats/pp_009926631474.html.

[12] Rodolfo Giometti. *BeagleBone Essentials*. Packt Publishing, Birmingham, UK, 2015.

[13] Joseph Greene. Pla and pha biodegradiation in the marine environment. *Report Topic*, 1(1):114, 05.03.2012.

[14] Mark A. Haidekker. *Linear FeedBack Controls - The Essensials*. Elsevier, 32 Jamestown Road, London NW1 7BY, UK, 2013.

[15] Daniel Rasmus Reite Morten Lerstad Solli Kristian Salvesen Homdrom, Marius Nonsvik. Towed-rov. *Bachelor thesis*, 478(10):114, 31.05.2018.

[16] Richard B. Langley. Nmea 0183: A gps reciever interface standard. http://gauss.gge.unb.ca/papers.pdf/gpsworld.july95.pdf.

[17] Oracle. Executor. https://docs.oracle.com/javase/8/docs/api/java/util/concurrent/Executor.html.

[18] Oracle. Networking basics. https://docs.oracle.com/javase/tutorial/networking/overview/networking.ht

[19] Kendall Roberg. Underwater color loss with gopro 0 to 155 feet depth - fishing lure deep test. https://www.youtube.com/watch?v=AAJjdA6b4Ts.

[20] Gimson Robotics. Electric linear actuator. https://gimsonrobotics.co.uk/categories/linear-actuators/products/gla750-p-12v-dc-linear-actuator-with-position-feedback.

[21] SparkFun. Humidity and temperature sensor hih6130. https://www.sparkfun.com/products/11295.

[22] SSH. Ssh (secure shell). https://www.ssh.com/ssh/.

[23] Technopedia. Actuator. https://www.techopedia.com/definition/17043/actuator.

[24] Technopedia. What is a network port? https://www.techopedia.com/definition/24717/network-port.

[25] Wikipedia. Coreldraw. https://en.wikipedia.org/wiki/CorelDRAW.

[26] Wikipedia. Google earth. https://en.wikipedia.org/wiki/Google_Earth.

Appendices

A Git

Git - Towed ROV.

B Preliminary Report

PRELIMINARY REPORT

FOR BACHELOR THESIS

NTNU
Kunnskap for en bedre verden

TITLE:
Towed ROV

CANDIDATE NUMBERS:			
Robin Stamnes Thorholm – 10051 **Håkon Inge Longva Haram – 10032** **Bjørnar Magnus Tennfjord - 10055**			
DATE: **17.01-2018**	COURCE CODE: **IE303612**	COURSE: **Bachelor thesis**	DOKUMENT ACCESS: - Open
STUDIUM: **AUTOMATION**		PAGES/ATTACHEMENT: 21/3	BIBL. NR: - Not in use -

ADVISORS:
Ottar L. Osen Paul Steffen Kleppe

OBJECTIVE/SUMMARY:

This is the pre-project report of the bachelor thesis *Towed ROV* which is provided by the Norwegian University of Science and Technology (NTNU). The thesis is a continuation of last year's thesis on the same subject.

The purpose of this thesis is to improve and further develop the underwater ROV that was made last year. The ROV should be remotely controllable while being towed behind a support vessel. The purpose of the underwater ROV is to be a multipurpose platform where sensors and functionality can easily be added/changed depending on the use.

The main objective of this thesis will be to rebuild the ROV for taking photos of the seabed and converting it to a 3D map, building a mobile base station for controlling and operating the ROV, along with using/improving what has already been built by the last year's thesis. We will improve the design of the ROV and fix the problems that occurred in the testing phase last year (leakage, condensation, camera). The graphical user interface will also be made more user friendly. We will also automate as much of the system as possible and try to make a "plug-n-play" system.

Denne oppgaven er en eksamensbesvarelse utført av student(er) ved NTNU i Ålesund.

Postadresse
Høgskolen i Ålesund
N-6025 Ålesund
Norway

Besøksadresse
Larsgårdsvegen 2
Internett
www.hials.no

Telefon
70 16 12 00
Epostadresse
postmottak@hials.no

Telefax
70 16 13 00

Bankkonto
7694 05 00636
Foretaksregisteret
NO 971 572 140

INNHOLD

1 INTRODUCTION

The maritime segment is entering a new era, with more and more of maritime operations performed underwater rather than on water. This is an area that is open for improvement and new ideas that the world has yet to see. Therefore, our goal is to be able to plan, design, engineer and build an underwater operated vehicle, that can handle underwater exploration with as little human involvement as possible.

With this bachelor thesis the group wants to use their knowledge from previous work experience and combine it with the knowledge they have gathered throughout their time at NTNU. Although some of the group members already have experience from subsea operation, it will be a completely new experience for everyone to build a ROV. This will result in a lot of research to get a better understanding of the subject.

NTNU is the provider of the thesis, and it is a continuation of last year's bachelor thesis. Because of this we already have the body of the ROV, and we know some of the fall pits. Our focus will be to improve the design of the ROV, its software and hardware. We will also look for new ideas we can implement to make the ROV even better.

2 CONCEPTS

ROV	Remote operated vehicle
UI	User interface
GUI	Graphical User Interface
Asana	Development tool for project organization, assigns tasks to members etc.

3 PROJECTORGANIZATION

3.1 *Project group*

Studentnummer(e)
Robin Stamnes Thorholm - 997507 Håkon Inge Longva Haram - 997501 Bjørnar Magnus Tennfjord - 997521

3.1.1 Assignments for the project group - organization

We will be using Instagantt together with Asana in order to keep track of all the assignments for the bachelor thesis. Each task will be assigned a designated person which is responsible for making sure that the task is done within the deadline. A second member will also be assigned as support to each task.

Name	**Role**
Robin Stamnes Thorholm	Project leader
Håkon Inge Longva Haram	Project secretary
Bjørnar Magnus Tennfjord	Project engineer

3.1.2 Assignments for project leader

- Main contact person
- Delegate tasks when necessary
- Responsible for progress

3.1.3 Assignments for secretary

- Secondary contact person
- Responsible for meetings (invitations, report writing)

3.1.4 Assignments for all members

- Each member is responsible for completing every task he has been assigned
- Each member is obligated to work on the thesis Mon-Fri from 9-15. See section 4.2 for more details regarding this
- Each member must log every work hour at the end of each week

3.2 *Management group*

Project advisors:

- Ottar L. Osen
- Paul Steffen Kleppe

4 AGREEMENTS

4.1 *Workspace and resources*

- Access to a place to work on the bachelor thesis
- Access to laboratories for automation, POD and mechanical work
- Access to 3D printers, welding equipment, and tools needed for the thesis

4.2 *Group agreements – work hours – attitude*

POSITIVE MENTAL ATTITUDE

Core time is between 9-15 Mon-Fri
Ideal work time is 8-16 Mon-Fri
Easter holiday from 17. April – 21. April.

- If a group member cannot be present during core time, the rest of the group shall be informed of this beforehand, and the group member must work in lost hours.
- Work hours shall be logged on a weekly basis
- Industry 4.0 is prioritized as the lectures is mandatory, on these days the work on the bachelor thesis expires.
- Progress meetings with advisors every other Thursday.
- Internal progress meeting every Monday 09.00 unless in conflict with Industry 4.0, then the meeting will be held the following day.
- Weekly progress reports every Friday.
- It is important to reach the set deadlines. All members must be willing to work overtime and the person responsible for the task is in charge for reaching the deadline.
- Although every task has two designated persons, the group should strive to have three individual operations going at all time. This is to maximize workhours and reduce the amount of "sitting and watching others work".
- If someone is stuck on a task, all group members should be informed of this, so the task can progress as soon as possible.

As automation engineers it is important to strive after new and innovative ideas. Because of how fast the technology develops today it is important to always research and adapt to new technologies to secure further development. We should always ask ourselves if this is the best way to do it, is there room for improvements? We should focus on improving safety, efficiency and value as well as human and environment life quality.

5 PROJECT DESCRIPTION

5.1 *Thesis problem*

The purpose of this thesis is to improve and further develop the underwater ROV that was made last year. The ROV should be remotely controllable while being towed behind a support vessel. The purpose of the underwater ROV is to be a multipurpose platform where sensors and functionality can easily be added/changed depending on the use.

The main objective of this thesis will be to rebuild the ROV for taking photos of the seabed and converting it to a 3D map, building a mobile base station for controlling and operating the ROV, along with using/improving what has already been built by the last year's thesis. We will improve the design of the ROV and fix the problems that occurred in the testing phase last year (leakage, condensation, camera). The graphical user interface will also be made more user friendly. We will also automate as much of the system as possible and try to make a "plug-n-play" system.

5.2 *Project specification*

As the ROV is already built we want to have our main focus on added functionality, but we will also focus on repairing and improving the ROV itself.

At the end of this bachelor thesis our goal is to have a functional ROV that can be properly tested underwater without leaks. We also want to continue and finish the control system of the ROV. Where last year's students focused on a control system that kept a set depth, in addition to this, we also wish to develop a collision detection system that can maneuver over steep sandbanks/hills automatically. The real time video feed will be upgraded to be on a gimbal and placed in a 180 degrees translucent dome. One of our goals is to make the picture quality good enough to create 3D maps of the ocean floor after the ROV has gathered data. We also want to design a graphical user interface which is more user friendly and visually pleasing than what was designed earlier.

Must have:	Should have:	Could have:
Working and user-friendly GUI	Anti-collision system that can detect sandbanks/hills	FPV-goggles to watch live feed
A control system that can keep the ROV at level, dive and rise	Sensor and signaling rod on ship	Heater to prevent moisture and condensation in camera enclosure
Live feed	Manual control of ROV	Auto adjust external lights based on video feed
Gimbal on camera with manual controls	Calculate approximate GPS position of ROV	
An electrical housing that is not filled with oil	Alarm system (detect leak, moisture, etc.)	
Live PID calibration in GUI	A plug and play system on all cables to ease the use of the ROV	

5.2.1 Economical requirements / budget

Last year's group used approximately 24 130 NOK. Their budget did not include consumables parts and was therefore a roughly estimate. Some of the equipment they used can be reused for our project. Because of this we estimate that our budget will be lower. One unknown variable is the waterproof boxes, this is not standard equipment and therefore it will be expensive. An alternative is to find a local mechanical workshop that can make this for us. Because of this it is hard to get an accurate estimate for total price. As of now, our ordering list has reached 15 000 NOK without the boxes but including all consumables we need. Another cost reduction factor is if the NTNU already has some of the parts we need already. Our roughly estimate of the total budget is 20 000 NOK. We believe that this will be enough to make a ROV with higher quality and easier use than last year.

5.3 *Method for development*

As mentioned, we will use Instagantt and Asana to keep track of our assignments and deadlines for each task. This will also help with keeping track of which person is responsible for making sure these tasks are done within the deadline.

How an object behaves in water is not always possible to predict, and we will therefore use the principle of trial and error.

The ROV has been made by other students in an earlier bachelor thesis. But it had some issues with a leakage, therefore the project will begin with getting an overview of what components needs to be repaired or changed, and find the leak and repair this.

Lean Product Development (LPD) is used for the process of utilizing less of everything. Examples of this are less development time, less production time and less cost to produce something. Either a physical product, knowledge product or service product. Lean Product Development is based on the idea of lean thinking and principles in lean manufacturing.

The most common concepts of LPD are:

- Creation of re-usable knowledge; knowledge created and maintained so it will be available for similar projects.
- Set-based concurrent engineering; multiple tasks will be ongoing simultaneously to decrease development time.
- Teams of responsible experts; organizing members associated on the project in areas they are experts on.
- Cadence and pull; engineers plan their own work and work their own plan.
- Visual management; visualizing the project progress, datagrams.
- Entrepreneurial system designer; LPD has one person responsible for the engineering and design of the product.

Since we have limited time and resources to make our prototype, will we use the LPD to decrease development and production time. This will be a viable development tool for our project.

5.4 *Information gathering*

Information gathered:

- Read the thesis that was done last year.
- Information given by the advisor at the start-up meeting.
- Inspection of the ROV

Information to be gathered:

- Inspect and dismantle most of the ROV to see what we can do to make it better.
- Research technologies revolving ROVs
- What we can do to improve the design
- Find a better solution for waterproofing
- Camera
- Video stream
- Possibilities of mapping the ocean floor

Frequency of insident	Severity of Consequences				
	Very Low Severity	Low Severity	Medium Severity	High Severity	Very High Severity
Very High Frequency					
High Frequency	• Electrical shock	• Crush hazard			• Ship collision
Medium Frequency	• Batteries not charged	• Work overload / stress	• Exposure to chemicals • ROV power loss at sea • Short circuit	• ROV Collision • ROV implosion • Sensor loss or controll loss	• Injuries when working with power tools • Man over board • Electrical Fire
Low Frequency				• Damage to equipment during transport	• Water leakage ROV
Very low Frequency				• Loss of ROV at sea	

5.5 *Risk analyzes*

Special risks for this project	Description / risk mitigation
Ship collision	When towing the ROV a lot of focus will be on the ROV, one person should be the designated driver of the ship to prevent collision.
Injuries when working with power tools	Necessary safety equipment will be used. The group has experience with power tools from before.
Man over board	When focusing on the ROV the group might not be focused on their environment. Every person onboard shall wear a lifejacket.
Electrical fire	Electrical fire might occur on the ROV or the base station. The group shall pay attention to the location of fire exits and fire countermeasures.
ROV Collision	In the test phase there is a chance for the ROV to hit underwater object. This can lead to a hull breach, loss of ROV or permanent damage to the ROV. It is important to drive carefully and take precautions.
ROV Implosion	As the ROV dives to its max depth there is a chance for the parts won't handle the pressure and implode. To minimize the risk of losing all the electronics we will perform a pressure test at 20 meters depth to ensure that it will not break.
Sensor loss or control loss	We will implement redundancy systems where needed. If anything is abnormal, we will start the emergency surface plan that we will design later.
Water leakage ROV	If the seals or gaskets are not properly sealed there is a chance for water to get into the ROV. We will perform tests without electronics to check if there is any leakage.
Damage to equipment during transport	Some of the equipment is heavy and fragile. The group is aware of this and will secure all components before transport and handle it with care.
Exposure to chemicals	As the ROV contains Oil and we will be using different kinds of chemical, there will always be a risk of exposure. The group will not be using any heavy chemicals so exposure should not have any health impact. Protective clothing will be used when needed.
ROV power loss at sea	If the ROV loses power the procedure will be to stop the ship and wait for the ROV to surface. As the ROV should have positive buoyancy.
Short circuit	Short circuit can damage components. The group will design all electrical circuits with fuses so the damage will be minimal to none.
Crush hazard	When working with the ROV there is a chance to get fingers crushed. The equipment we are using is not heavy enough to cause any serious damage.
Loss of ROV at sea	If the umbilical cord should be cut, the ROV will rise to the surface du to positive buoyancy. If not the group as access to diving equipment.
Electrical shock	The equipment is operating with 36V DC, therefore there is no danger for injuries due to electricity.
Batteries not charged	If the batteries to the ROV is not charged then it will cause long delays in the until we can continue with testing.

Risk for not be finished in time	Description / risk mitigation
Redesign main ROV structure	If the original design of the ROV is not enough hydrodynamical, we will have to redesign it from scratch. From the beginning of the project we will look into ways to improve the old design without major changes.
Failed plans	There is a possibility that our plans work in theory but not in practice. It is important to have a backup plan, so we don't loose to much time with redesigns.
Sickness	Sickness is hard to avoid in our workplace since many other people work in the same area. Good personal hygiene is a key component to stay heathy.
Lack of resources / testing equipment	Testing of equipment will take a lot of time preparing, if the necessary equipment for testing isn't ready within time. Make sure everything is ready before testing.
Unpredicted issues	There is a chance that some unpredicted issues might be time consuming and delaying other tasks. We can counter this by studying last years thesis do extensive research.
Delay in delivery	Parts ordered might not be delivered within expected time. To reduce this risk, we will try order part in good time if possible. And try to avoid ordering from countries with long delivery time.
Work overload / stress	The group is known for working hard from earlier projects. Due to the size of this project it is important to plan ahead and plan the work hours to reduce stress.
Injury while working	The work we will be performing does not involve any risk for long time injuries. We will follow NTNU's laboratory policies and the "Laboratory and workshop Handbook": http://www.goo.gl/xhfX6x

The group believes that it is possible to finish the project within the given timeframe. Many of the software aspect like communications and GUI can be reused from earlier projects we have done.

One of our main concerns is how much of last year's work that has to be redone. We have to spend time cleaning and draining the ROV for oil. And we are concerned that the box design of the ROV will not be as hydrodynamic as the last years student believed, making it hard to control underwater. As they did not perform any lengthy sea trail, we are going blind into the seaworthiness of the ROV until we can test for our self. This might have a huge time impact if we have to do any major redesign.

The deadlines in the Gantt diagram is important to follow. If any of the tasks exceeds its deadline the whole project will be delayed. It is therefore important to hold these deadlines. There is of course no need to wait until the start date of a task, if the person is done with his current tasks.

If the group finds out they will not reach their main goals, we will start to exclude tasks from the "could have"-list. Worst case scenario, we will exclude tasks from the "should have"-list.

5.6 *Main work activities*

RST – Robin Stamnes Thorholm

HILH – Håkon Inge Longva Haram

BMT – Bjørnar Magnus Tennfjord

Nr	Main activity	Responsibility	Time/Scope
A1	**Writing and Research**	**RST, HILH, BMT**	**5 Months**
A11	Pre-report	BMT, HILH, RST	1 Week
A12	Review	HILH, RST, BMT	1 Day
A13	Main Thesis	HILH, RST, BMT	5 Months
A14	Review 1	BMT	1 Day
A15	Review 2	RST	1 Day
A16	Review 3	HILH	1 Day
A17	Delivery	RST	1 Day
A2	**Materials**	**RST, BMT**	**2 Weeks**
A21	3D printing	RST, HILH	2 Weeks
A22	Metal work	RST, BMT	2 Weeks
A23	Order parts	BMT, RST	2 Weeks
A24	Find Suppliers	BMT, RST	2 Weeks
A3	**Software**	**RST, HILH, BMT**	**2 Months**
A31	Main program ROV	RST, HILH	2 Months
A32	Communication ROV	RST, BMT	3 Weeks
A33	Communication vessel	BMT, RST	5 Weeks
A34	Communication	RST, BMT	2 Weeks
A35	GUI	HILH, RST	7 Weeks
A36	Mapping ocean floor	RST, HILH	1.5 Weeks
A4	**Regulating System**	**HILH, BMT**	**2 Months**
A41	Simulation	HILH, BMT	1 Month
A42	PID	HILH, BMT	2 Weeks
A43	Fuzzy Logic	BMT, HILH	1 Month
A44	Calibration	BMT, HILH	2 Weeks
A5	**Hardware**	**RST, BMT**	**2 Months**
A51	Cabling and power solutions	RST, BMT	1.5 Weeks
A52	Replace components	RST, BMT	2 Weeks
A53	Camera system	RST, HILH	2 Weeks
A54	Ship base station	BMT, HILH	4 Weeks
A55	Installation	BMT, HILH	2 Weeks
A6	**Testing**	**RST, BMT, HILH**	**4 Weeks**
A61	Test of hardware	RST, BMT, HILH	2 Weeks
A62	Test at sea	RST, BMT, HILH	2 Weeks

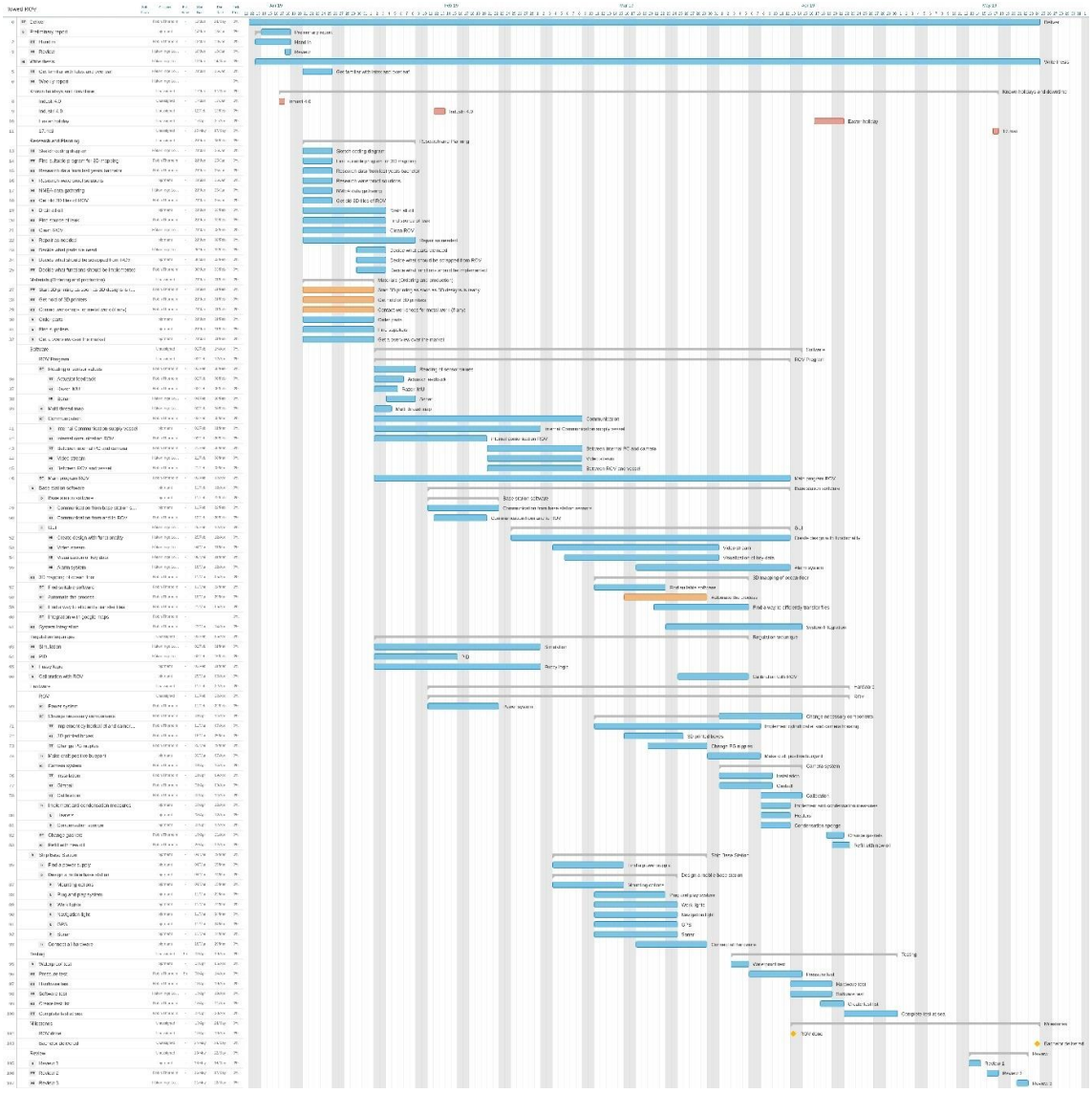
5.7 *Schedule / Timetable*

5.7.1 Main project plan

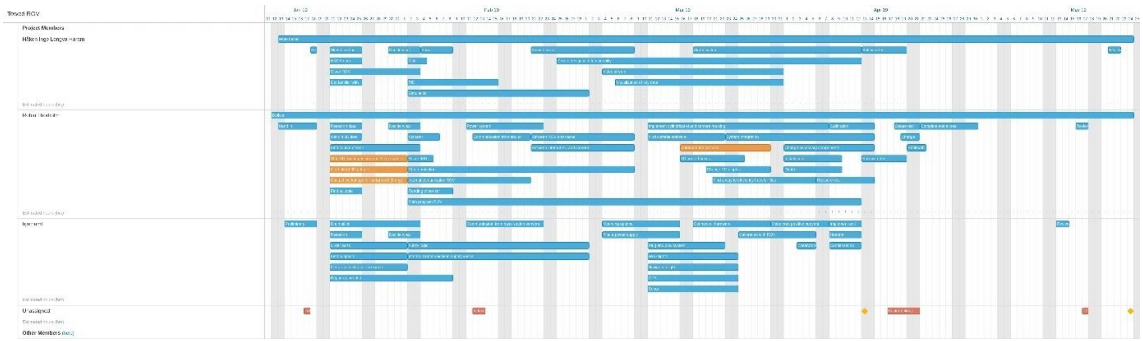
It is important to follow the work plan as closely as possible to reach the deadlines. All members will always be working on different tasks, resulting in concurrent engineering.

All of the tasks in the main project plan has a designated person responsible. This does not mean that it is that person who will do it, he is only responsible for making sure that it is done within the deadline. There will always be at least one person assigned as support for each task.

The diagrams below (Figure 1 and 2) is set up using Instagantt together with Asana which displays the main project plan and a workload plan showing the task distribution between each member. When distributing the tasks, we made sure each member always has 2-3 tasks to work on at all times, which gives good efficiency and reduces work downtime.



Figur 1: Gantt diagram of all tasks. For larger image, see appendix.



Figur 2: Workload diagram of all tasks. For larger image, see appendix.

5.7.2 Project control assets

To have a system of the projects schedule, keeping track of hours and deadlines, will we be using Asana. This is a development kit to help developers, and include features such as:

- Gantt diagrams, a graphical representation of work needed to be done over a period of time.
- Assign task to members.
- Applying priorities to tasks
- Mark tasks as done.
- Synchronization with Github, possible to mark tasks as completed when source code is uploaded.

5.7.3 Development assets

For development and creation of a good quality product, it is necessary to use effective, and up to date development assets. This includes developing environments for creating the required software, schematics, design drawings etc.

- Netbeans IDE, to create the java application.
- Autodesk ReCap, for 3D mapping of ocean floor
- Matlab/Simulink, simulation of PID.
- Sourcetree, used so multiple people can work simultaneously at the project. Used to create multiple working copies of the source code and sync it to the main repository.
- Github, remote repository for code development
- Cura, preparing 3D models for 3D printing
- Siemens NX, tool for designing 3D models.

5.7.4 Internal evaluating control

Internal control is covered by many of the earlier topics. One part of the internal control is the meeting with the advisors. These meeting will be held every other Thursday. In these meeting we will receive feedback on the work that has been done and how we could improve.

We will also have internal meetings every Monday to discuss present and future challenges. We will also discuss how each other's task will affect one another. This is

important, so we prepare for integration of each other's work and avoid stepping on anyone's toes.

The criteria for a task to be set complete is that we don't have to go back and spend time redoing a task. The task must be ready to be implemented to the ROV when set completed, in other words, testing is finished.

Although each task should be working as intended when marked complete, we must remember that there is a difference between perfect and complete. We do not have enough time to make everything perfect, especially on a large project like this with only three group members.

The ROV should be as environmentally clean as possible. As we have learned from last year students it is hard to make the ROV completely sealed and there is a guaranty that some oil will leak. It is therefore important to use oil and materials that will not hurt the environment in any way. It is also important that during the test phase we only operate in shallow water, no deeper than 20m. This way we can recover the ROV or lost parts with standard scuba diving equipment with low to non-risk for the diver.

We also want to implement "stop the job" policy. Every group member has the right to stop the job in progress if a risk is observed that could lead to injury, damage to equipment or the environment.

5.8 *Decision making process*

Every decision regarding the project will be decided by the group. We will also discuss this with the advisors in the next available group meeting. These decisions can be limitations to a task or cancelling a task altogether. If we find ourselves in a position where we have to limit our expectations due to time, we will first and foremost cancel objectives that are on our "should have" list, see 5.2. We will after this update our Gantt diagram to how much workload we have removed and update our weekly report.

6 DOCUMENTATION

6.1 *Reports and technical documents*

In documentation every source found or used will be noted and why we used it or why not. This applies to much of the equipment that will be used. Why we used it and why the equipment was chosen over other solutions. The datasheets of the used equipment will be added as appendixes in the main report, this way it is easy to access the attributes and limitations of the equipment. Sources used in the project is to be referred, when and where it was found.

Documentation of test will be done after every test is completed, the details of the test will be added in the report both when the test goes well or if there is a failure while testing. Documentation of what caused the failure and improvements done after testing. The same applies for the test if it was successful.

Project materials will be stored in the universities workshops in assigned project areas. The materials will be secured behind password protected doors where explicit access is needed to get in. This means there are only engineering students and co-workers allowed access to this equipment.

Maintenance will be done routinely and before and after testing to prevent equipment damage. The group will go over the construction before the test, to control there is no fault in equipment. The same follows for after tests of the ROV. We will also develop a

maintenance manual so later students will have good documentation on how to do proper maintenance.

A simple user manual will also be made so that later users can make use of the ROV even though they do not have the necessary technical skills.

7 PLANNED MEETINGS AND REPORTS

7.1 Meetings

7.1.1 Meetings with control group

After the first meeting with the supervisors, it was decided to have the meeting every other Thursday from 10:30 to 11:00. During these meetings we will discuss issues encountered and explanation of solutions. We will also discuss if there are any changes we should consider, or if there is something we can or could do differently.
The meeting will be led by the project leader, and the secretary will write the meeting report.

7.1.2 Internal meetings

The group will have internal meetings Mondays at 09:00. We have decided to start the week with a meeting, were we will go through the previous weeks progress, what we will focus on further and update our goals in Instagantt. We will discuss problems encountered and figure out possible solutions for the problems. The meeting will be led by the project leader, and the secretary will write the meeting report.

7.2 *Progress report*

A report will be written every week to keep track of the progress. This report will be a short summary of progress made and include the focus for the next weeks progress. This report will also be used to document our deadlines for tasks to be completed or if the task has gone past the deadline.

8 PLANNED DEVIATION MANAGEMENT

In the case of delayed work, the group will adjust their schedules with additional work hours to get the tasks complete within the deadline. If the task is becoming too time consuming or challenging, the group must try to find an additional solution that does not affect the end-result.

If there are changes in the schedule or the deadline of a task has been changed, the group must discuss this in the weekly group meetings and make changes on task priority and edit the amount of work to the students that are responsible for this task.

Each student will be assigned different tasks with a responsibility for completing this within the deadline date. To every task there will be assigned one other student to assist

with the responsibility for completing the task. In the event of tasks seems to be too time consuming or other reasons, the student must take this up with the group. Where the group then will make a decision regarding this issue.

9 EQUIPMENT REQUIREMENTS FOR PROJECT EXECUTION

Equipment used on ROV from last year that we need:

What:	Why:
Sonar	View of seabed, regulation control
Depth sensor	Get actual depth, regulation control
Camera	Footage of seabed
Gyro	Stabilization factor, regulation control
Fibre rope/wiew	Used for towing ROV
Power supply	Powering the ROV
3D-printer	3D printing the wings and possibly other components
Mini pc/Odroid/raspberry	Operating processor onboard ROV, runs main program
Waterproof actuators	Control of ROV wings
Accessories	Seals, wiring bearing
Waterproof floodlight	Illuminate the seafloor
Construction Materials	Constructing the ROV

Equipment we need for our project execution:

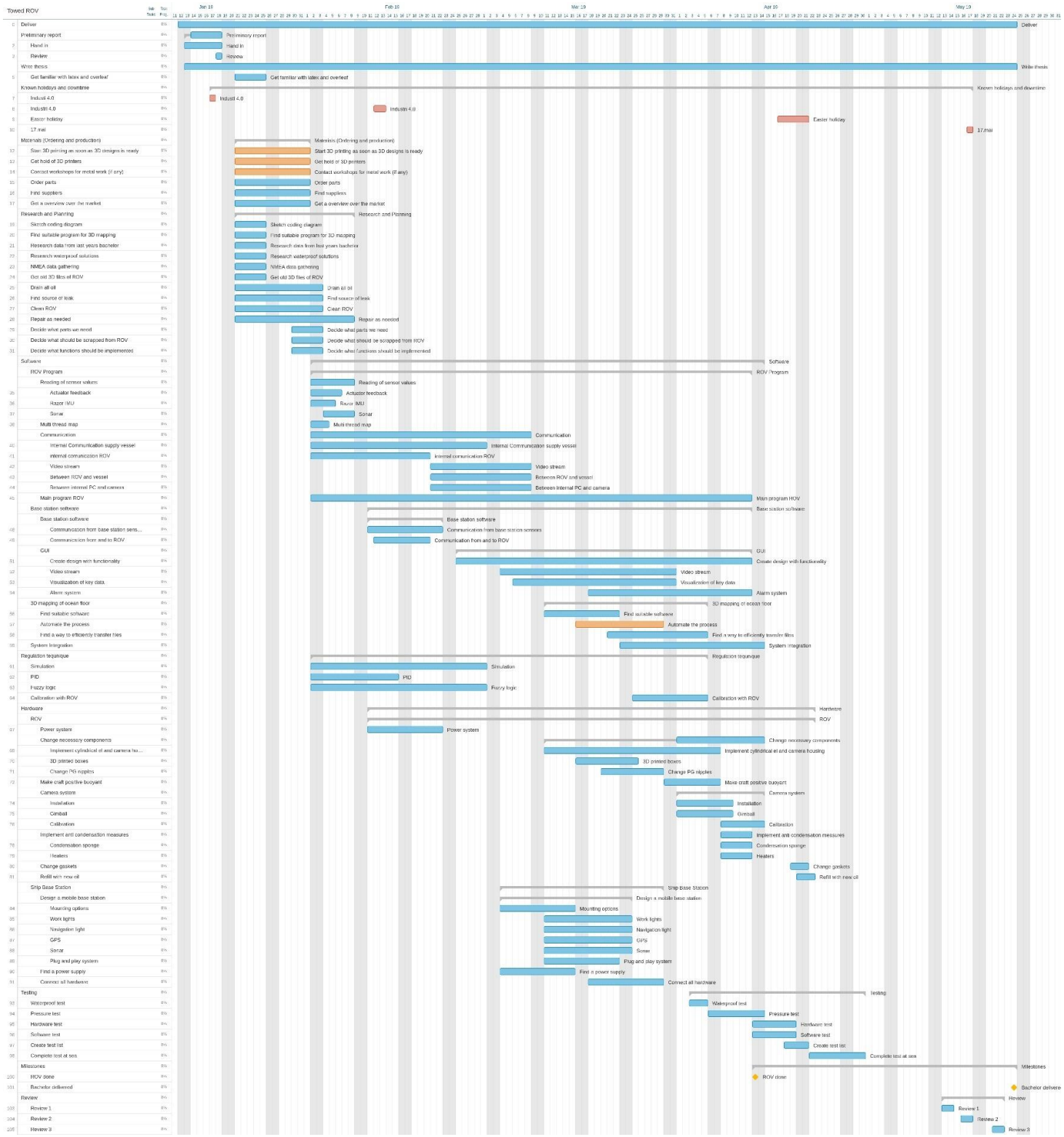
What:	Why:
Extra Sonar	Sonar mounted on ship, so we can detect dangers ahead of ROV
Gimbal	Control and stabilize camera on ROV
Waterproof enclosure	Changing some of the boxes on the ROV from 3D printed boxes to pressure proof enclosures
Razor IMU	Sensor for measuring yaw, pitch, roll and heading on the ROV. (This has inbuilt magnetic deviation compensation)
FPV goggles	Used for watching the live feed from the ROV, utilizing head tracking for controlling gimbal
Low light high resolution camera	We need the best quality to be able to 3D map the ocean floor
Underwater lights	We suspect that the lights on the ROV is leaking.

10 REFERENCES

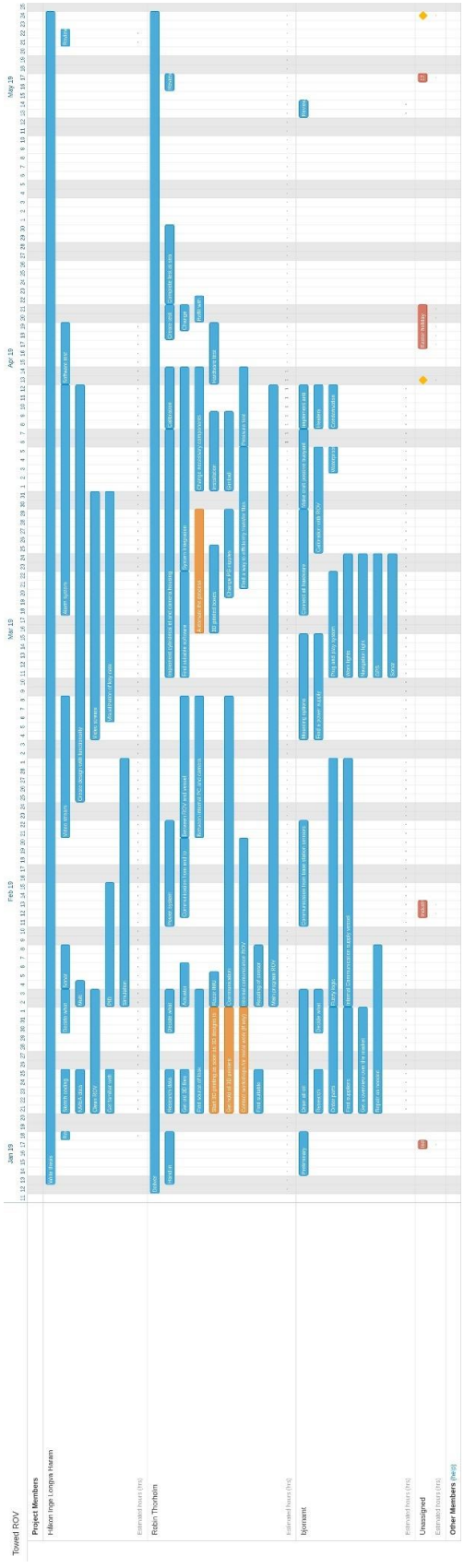
APENDIXES

Appendix A	Instagantt diagram
Appendix B	Workload diagram
Appendix C	Work plan in text

10.1 *Appendix A – Gantt diagram*



10.2 *Appendix B – Workload diagram*

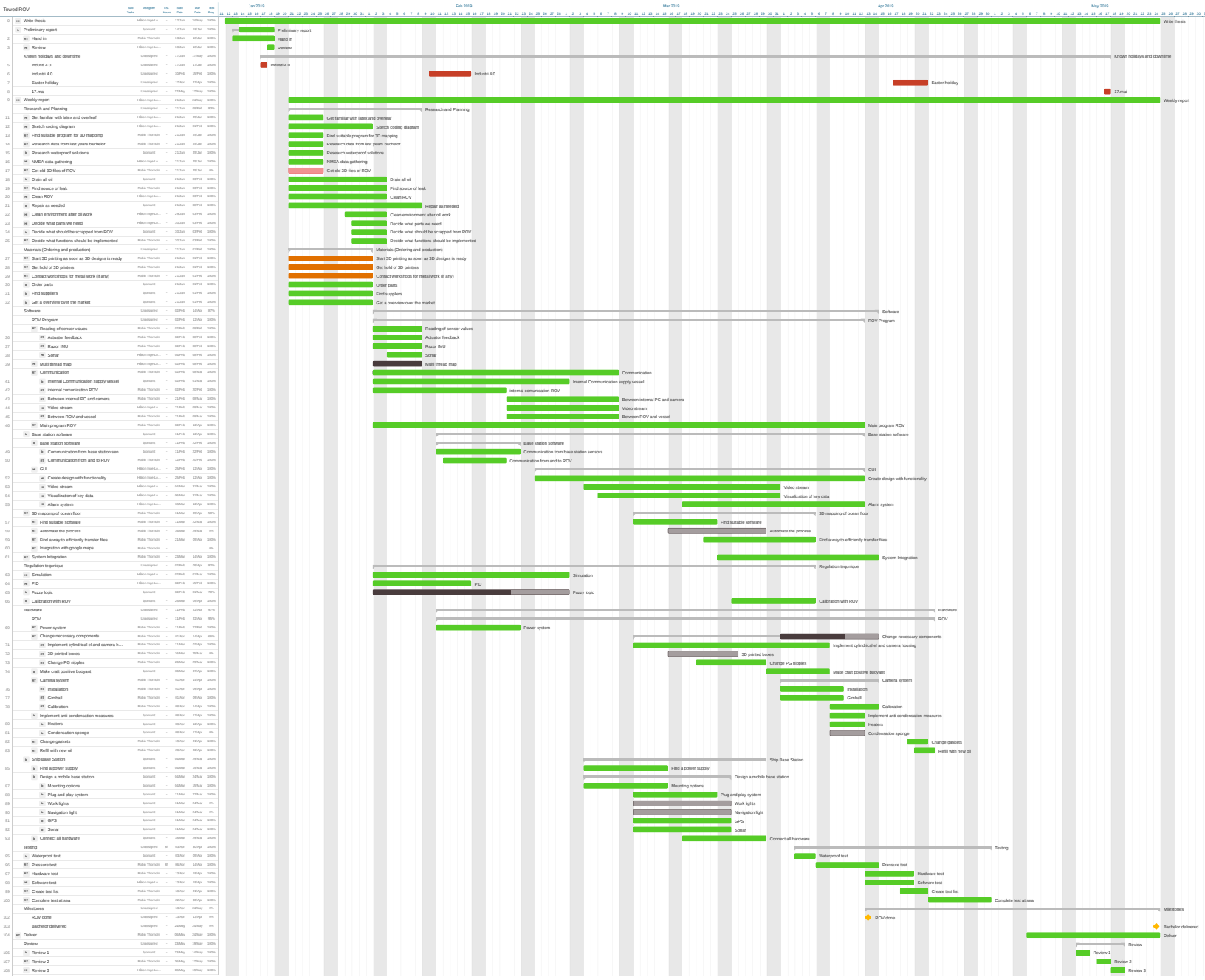


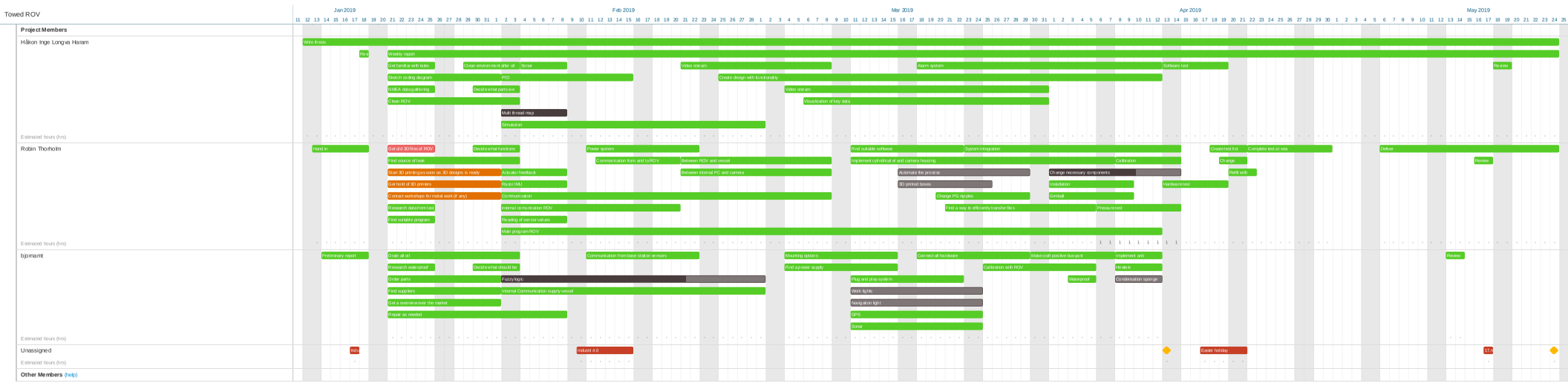
10.3 *Appendix C - Work plan in text*

Level	Name	Progress	Start	Due	User	Assistant
1	Deliver		12/01/2019	24/05/2019	Robin Thorholm	All
1	Preliminary report	0%	14/01/2019	18/01/2019	Bjørnar Magnus Tennfjord	All
2	Hand in		13/01/2019	18/01/2019	Robin Thorholm	All
2	Review		18/01/2019	18/01/2019	Håkon Inge Longva Haram	All
1	Write thesis	0%	13/01/2019	24/05/2019	Håkon Inge Longva Haram	All
2	Get familiar with latex and overleaf		21/01/2019	25/01/2019	Håkon Inge Longva Haram	All
2	Weekly report				Håkon Inge Longva Haram	All
1	Materials (Ordering and	0%				
2	Start 3D printing as soon as 3D designs is ready		21/01/2019	01/02/2019	Robin Thorholm	Håkon Inge Longva Haram
2	Get hold of 3D printers		21/01/2019	01/02/2019	Robin Thorholm	Håkon Inge Longva Haram
2	Contact workshops for metal work (if any)	0%	21/01/2019	01/02/2019	Robin Thorholm	Bjørnar Magnus Tennfjord
2	Order parts		21/01/2019	01/02/2019	Bjørnar Magnus Tennfjord	Robin Thorholm
2	Find suppliers		21/01/2019	01/02/2019	Bjørnar Magnus Tennfjord	Robin Thorholm
2	Get a overview over the market		21/01/2019	01/02/2019	Bjørnar Magnus Tennfjord	Robin Thorholm
1	Research and Planning	0%				
2	Sketch coding diagram		21/01/2019	25/01/2019	Håkon Inge Longva Haram	Robin Thorholm
2	Find suitable program for 3D mapping		21/01/2019	25/01/2019	Robin Thorholm	Håkon Inge Longva Haram
2	Research data from last years bachelor		21/01/2019	25/01/2019	Robin Thorholm	All
2	Research waterproof solutions		21/01/2019	25/01/2019	Bjørnar Magnus Tennfjord	Robin Thorholm
2	NMEA data gathering		21/01/2019	25/01/2019	Håkon Inge Longva Haram	Robin Thorholm
2	Get old 3D files of ROV		21/01/2019	25/01/2019	Robin Thorholm	Bjørnar Magnus Tennfjord
2	Drain all oil		21/01/2019	03/02/2019	Bjørnar Magnus Tennfjord	Håkon Inge Longva Haram
2	Find source of leak		21/01/2019	03/02/2019	Robin Thorholm	Bjørnar Magnus Tennfjord
2	Clean ROV		21/01/2019	03/02/2019	Håkon Inge Longva Haram	Robin Thorholm
2	Repair as needed		21/01/2019	08/02/2019	Bjørnar Magnus Tennfjord	Robin Thorholm
2	Decide what parts we need		30/01/2019	03/02/2019	Håkon Inge Longva Haram	Robin Thorholm
2	Decide what should be scrapped from ROV		30/01/2019	03/02/2019	Bjørnar Magnus Tennfjord	Håkon Inge Longva Haram
2	Decide what functions should be implemented		30/01/2019	03/02/2019	Robin Thorholm	Bjørnar Magnus Tennfjord
1	Software	0%				
2	ROV Program	0%				
3	Reading of sensor values	0%	02/02/2019	08/02/2019	Robin Thorholm	
4	Actuator feedback		02/02/2019	06/02/2019	Robin Thorholm	Bjørnar Magnus Tennfjord
4	Razor IMU		02/02/2019	05/02/2019	Robin Thorholm	Bjørnar Magnus Tennfjord
4	Sonar		04/02/2019	08/02/2019	Håkon Inge Longva Haram	Robin Thorholm
3	Multi thread map		02/02/2019	04/02/2019	Håkon Inge Longva Haram	Robin Thorholm
3	Communication	0%	02/02/2019	08/03/2019	Robin Thorholm	
4	Internal Communication supply vessel	0%	02/02/2019	01/03/2019	Bjørnar Magnus Tennfjord	Robin Thorholm
4	Internal comunication ROV		02/02/2019	20/02/2019	Robin Thorholm	Bjørnar Magnus Tennfjord
4	Video stream		21/02/2019	08/03/2019	Håkon Inge Longva Haram	Robin Thorholm
4	Between ROV and vessel		21/02/2019	08/03/2019	Robin Thorholm	Bjørnar Magnus Tennfjord
4	Between internal PC and camera		21.02.2019	08.03.2019	Robin Thorholm	Bjørnar Magnus Tennfjord
3	Main program ROV		02/02/2019	12/04/2019	Robin Thorholm	Håkon Inge Longva Haram
2	Base station software	0%			Bjørnar Magnus Tennfjord	
3	Base station software	0%			Bjørnar Magnus Tennfjord	
4	Communication from base station sensors		11/02/2019	22/02/2019	Bjørnar Magnus Tennfjord	Robin Thorholm
4	Communication from and to ROV		12/02/2019	20/02/2019	Robin Thorholm	Bjørnar Magnus Tennfjord
3	GUI	0%			Håkon Inge Longva Haram	
4	Create design with functionality		25/02/2019	12/04/2019	Håkon Inge Longva Haram	Robin Thorholm
4	Video stream		04/03/2019	31/03/2019	Håkon Inge Longva Haram	Robin Thorholm
4	Visualization of key data		06/03/2019	31/03/2019	Håkon Inge Longva Haram	Robin Thorholm
4	Alarm system		18/03/2019	12/04/2019	Håkon Inge Longva Haram	Robin Thorholm
2	3D mapping of ocean floor	0%			Robin Thorholm	
3	Find suitable software		11/03/2019	22/03/2019	Robin Thorholm	Håkon Inge Longva Haram
3	Automate the process		16/03/2019	29/03/2019	Robin Thorholm	Håkon Inge Longva Haram
3	Find a way to efficiently transfer files		21/03/2019	05/04/2019	Robin Thorholm	Håkon Inge Longva Haram
3	Integration with google maps			Invalid date	Robin Thorholm	Håkon Inge Longva Haram
2	System integration		23/03/2019	14/04/2019	Robin Thorholm	All
1	Regulation tequnique	0%				
2	Simulation		02/02/2019	01/03/2019	Håkon Inge Longva Haram	Bjørnar Magnus Tennfjord
2	PID		02/02/2019	15/02/2019	Håkon Inge Longva Haram	Bjørnar Magnus Tennfjord
2	Fuzzy logic		02/02/2019	01/03/2019	Bjørnar Magnus Tennfjord	Håkon Inge Longva Haram
2	Calibration with ROV		25/03/2019	05/04/2019	Bjørnar Magnus Tennfjord	Håkon Inge Longva Haram

1	Hardware	0 %				
2	ROV	0 %				
3	Power system		11.02.2019	22.02.2019	Robin Thorholm	Bjørnar Magnus Tennfjord
3	Change necessary components	0 %	01.04.2019	14.04.2019	Robin Thorholm	Bjørnar Magnus Tennfjord
4	Implement cylindrical el and camera housing		11.03.2019	07.04.2019	Robin Thorholm	Bjørnar Magnus Tennfjord
4	3D printed boxes		16.03.2019	25.03.2019	Robin Thorholm	Bjørnar Magnus Tennfjord
4	Change PG nipples		20.03.2019	29.03.2019	Robin Thorholm	Bjørnar Magnus Tennfjord
3	Make craft positive buoyant		30.03.2019	07.04.2019	Bjørnar Magnus Tennfjord	Robin Thorholm
3	Camera system	0 %			Robin Thorholm	
4	Installation		01.04.2019	09.04.2019	Robin Thorholm	Håkon Inge Longva Haram
4	Gimball		01.04.2019	09.04.2019	Robin Thorholm	Håkon Inge Longva Haram
4	Calibration		08.04.2019	14.04.2019	Robin Thorholm	Håkon Inge Longva Haram
3	Implement anti condensation measures	0 %	08.04.2019	12.04.2019	Bjørnar Magnus Tennfjord	
4	Condensation sponge		08.04.2019	12.04.2019	Bjørnar Magnus Tennfjord	Robin Thorholm
4	Heaters		08.04.2019	12.04.2019	Bjørnar Magnus Tennfjord	Robin Thorholm
3	Change gaskets		19.04.2019	21.04.2019	Robin Thorholm	Bjørnar Magnus Tennfjord
3	Refill with new oil		20.04.2019	22.04.2019	Robin Thorholm	Bjørnar Magnus Tennfjord
2	Ship Base Station	0 %			Bjørnar Magnus Tennfjord	
3	Design a mobile base station	0 %			Bjørnar Magnus Tennfjord	
4	Mounting options		04.03.2019	15.03.2019	Bjørnar Magnus Tennfjord	Håkon Inge Longva Haram
4	Work lights		11.03.2019	24.03.2019	Bjørnar Magnus Tennfjord	Håkon Inge Longva Haram
4	Navigation light		11.03.2019	24.03.2019	Bjørnar Magnus Tennfjord	Håkon Inge Longva Haram
4	GPS		11.03.2019	24.03.2019	Bjørnar Magnus Tennfjord	Håkon Inge Longva Haram
4	Sonar		11.03.2019	24.03.2019	Bjørnar Magnus Tennfjord	Håkon Inge Longva Haram
4	Plug and play system		11.03.2019	22.03.2019	Bjørnar Magnus Tennfjord	Håkon Inge Longva Haram
3	Find a power supply		04.03.2019	15.03.2019	Bjørnar Magnus Tennfjord	Håkon Inge Longva Haram
3	Connect all hardware		18.03.2019	29.03.2019	Bjørnar Magnus Tennfjord	Håkon Inge Longva Haram
1	Testing	0 %				
2	Waterproof test		03.04.2019	05.04.2019	Bjørnar Magnus Tennfjord	All
2	Pressure test		06.04.2019	14.04.2019	Robin Thorholm	All
2	Hardware test		13.04.2019	19.04.2019	Robin Thorholm	All
2	Software test		13.04.2019	19.04.2019	Håkon Inge Longva Haram	All
2	Create test list		18.04.2019	21.04.2019	Robin Thorholm	All
2	Complete test at sea		22.04.2019	30.04.2019	Robin Thorholm	All
1	Milestones	0 %				
2	ROV done		13.04.2019	13.04.2019		
1	Review	0 %				
2	Review 1		13.05.2019	14.05.2019	Bjørnar Magnus Tennfjord	All
2	Review 2		16.05.2019	17.05.2019	Robin Thorholm	All
2	Review 3		21.05.2019	22.05.2019	Håkon Inge Longva Haram	All

C Gantt diagram


Towed ROV
Jan 2019
Feb 2019
Mar 2019
Apr 2019
May 2019
Write thesis
Preliminary report
Hand in
Review
Known holidays and downtime
Industri 4.0
Easter holiday
17.mai
Weekly report
Research and Planning
Get familiar with latex and overleaf
Sketch coding diagram
Find suitable program for 3D mapping
Research data from last years bachelor
Research waterproof solutions
NMEA data gathering
Get old 3D files of ROV
Drain all oil
Find source of leak
Clean ROV
Repair as needed
Clean environment after oil work
Decide what parts we need
Decide what should be scrapped from ROV
Decide what functions should be implemented
Materials (Ordering and production)
Start 3D printing as soon as 3D designs is ready
Get hold of 3D printers
Contact workshops for metal work (if any)
Order parts
Find suppliers
Get a overview over the market
Software
ROV Program
Reading of sensor values
Actuator feedback
Razor IMU
Sonar
Multi thread map
Communication
Internal Communication supply vessel
internal comunication ROV
Between internal PC and camera
Video stream
Between ROV and vessel
Main program ROV
Base station software
Communication from base station sensors
Communication from and to ROV
GUI
Create design with functionality
Video stream
Visualization of key data
Alarm system
3D mapping of ocean floor
Find suitable software
Automate the process
Find a way to efficiently transfer files
System Integration
Regulation tequnique
Simulation
PID
Fuzzy logic
Calibration with ROV
Hardware
ROV
Power system
Change necessary components
Implement cylindrical el and camera housing
3D printed boxes
Change PG nipples
Make craft positive buoyant
Camera system
Installation
Gimball
Calibration
Implement anti condensation measures
Heaters
Condensation sponge
Change gaskets
Refill with new oil
Ship Base Station
Find a power supply
Design a mobile base station
Mounting options
Plug and play system
Work lights
Navigation light
GPS
Sonar
Connect all hardware
Testing
Waterproof test
Pressure test
Hardware test
Software test
Create test list
Complete test at sea
Milestones
ROV done
Bachelor delivered
Deliver
Review
Review 1
Review 2
Review 3


Towed ROV
Jan 2019
Feb 2019
Mar 2019
Apr 2019
May 2019
Project Members
Håkon Inge Longva Hasrum
Write thesis
Plan
Weekly report
Get familiar with latex
Clean environment after oil
Sonar
Video stream
Alarm system
Software test
Review
Sketch coding diagram
PID
Create design with functionality
NMEA data gathering
Decide what parts we
Video stream
Clean ROV
Visualization of key data
Multi thread map
Simulation
Estimated hours (hrs)
Robin Thorholm
Hand in
Get old 3DFiles of ROV
Decide what functions
Power system
Find suitable software
System integration
Create test list
Complete test at sea
Deliver
Find source of leak
Communication from and to ROV
Between ROV and vessel
Implement cylindrical oil and camera housing
Calibration
Change
Review
Start 3D printing as soon as 3D designs is ready
Actuator feedback
Between internal PC and camera
Automate the process
Change necessary components
Refit with
Get hold of 3D printers
Rasmi MU
3D printed bases
Institution
Hardware test
Contact workshops for metal work (if any)
Communication
Change PID replies
Gimbal
Research data from last
Internal communication ROV
Find a way to efficiently transfer files
Pressure test
Find suitable programs
Reading of sensor values
Main program ROV
Estimated hours (hrs)
bjornamt
Preliminary report
Drown all oil
Communication from base station sensors
Mounting options
Connect all hardware
Make craft positive buoyant
Implement unit
Review
Research waterproof
Decide what should be
Find a power supply
Calibration with ROV
Penetrators
Order parts
Fuzzy logic
Plug and play system
Weterproof
Condemnation sponge
Find suppliers
Internal Communication supply vessel
Work lights
Get a overview over the market
Navigation light
Repair as needed
GPS
Sonar
Estimated hours (hrs)
Unassigned
Industri 4.0
Easter holiday
17.
Estimated hours (hrs)
Other Members (help)

D Work hours

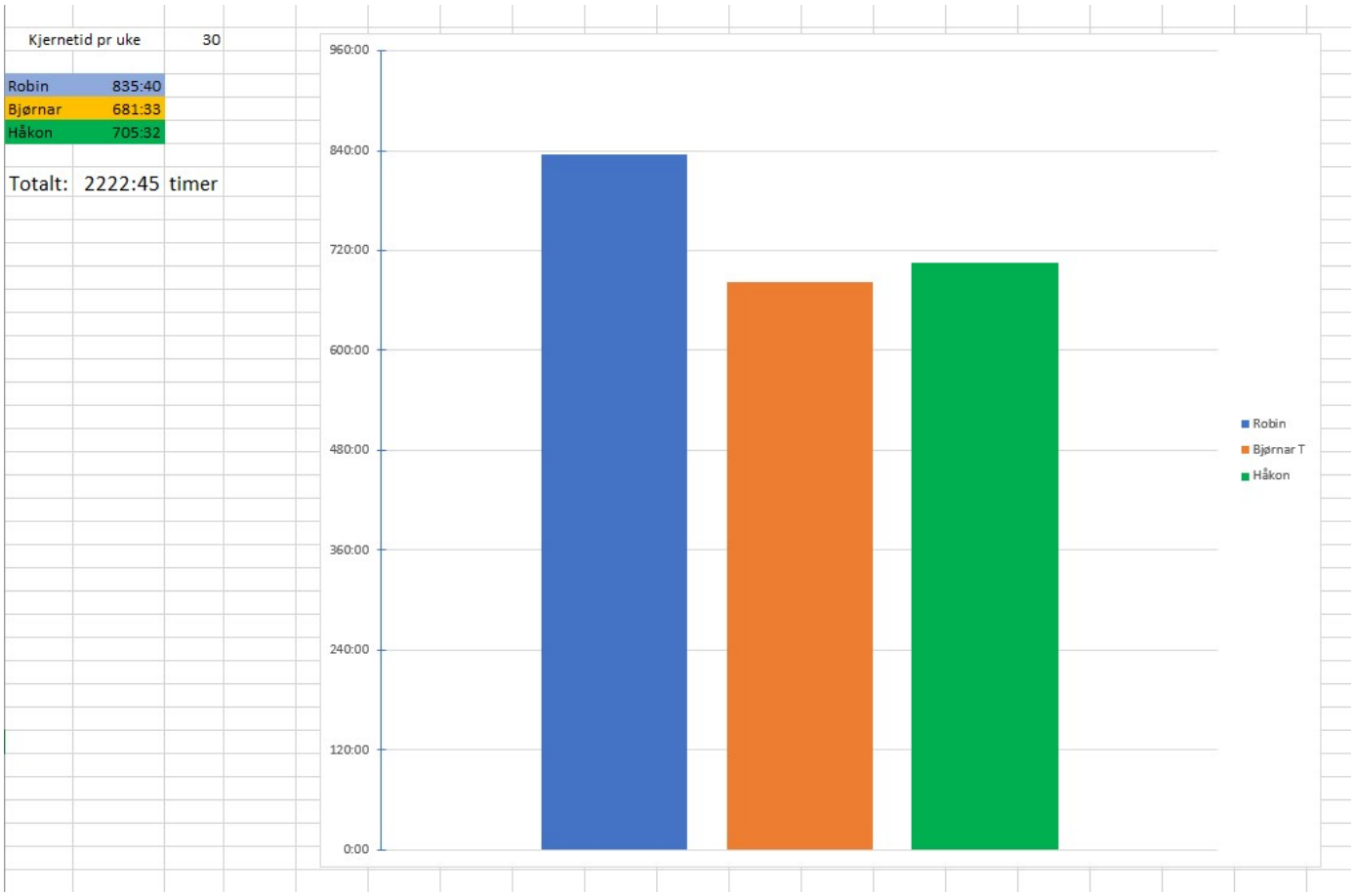


Figure 1: Work hours statistics

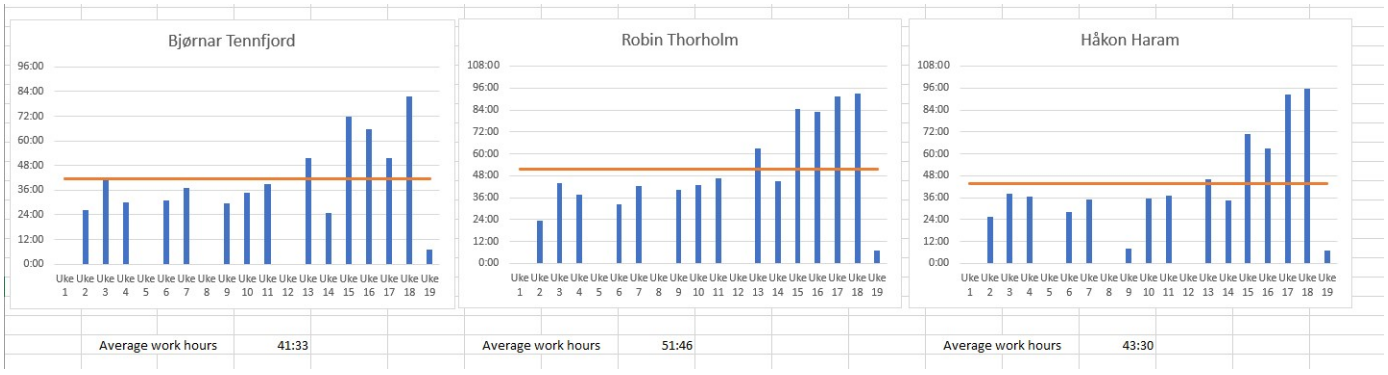


Figure 2: Average work hours statistics

E Budget

Type	vare.nr	link	antall	pris	total	total NOK
Trykk-sensor	9026.20.4000	erobotics.com/store/sensors-sonars-cameras/sensors/	1			
SOS Probes 6" Single tip probe	5309.29.30	uerobotics.com/store/sensors-sonars-cameras/leak-ser	8	$2,00	$16,00	
Vent plug	7616.10	bluerobotics.com/store/cables-connectors/penetrators	6	$8,00	$48,00	
Vacuum Plug	7616,1	tics.com/store/watertight-enclosures/enclosure-tools-	1	$8,00	$8,00	
Watertight Enclosure 4"		https://www.bluerobotics.com/store/watertight-enclo	1	$171,00	$171,00	
				Total	$243,00	NOK 2 057,97

Figure 3: Budget Blue Robotics

Vare	vare.nr	link	Antall	Pris	total	
9DoF Razor IMU	SEN-14001	https://www.sparkfun.com/products/14001	2	$35,95	$71,90	
Humidity & Temp Sensor Breakout - HIH6130	SEN-11295	https://www.sparkfun.com/products/11295	2	$29,95	$59,90	
				Total	$131,80	kr 1 116,21

Figure 4: Sparkfun budget

Vare	Vare.nr	link	antall	Pris	total	
Jrk 21v3 motor controller	3146	https://www.pololu.com/product/3146	2	$49,95	$99,90	
					$99,90	kr 846,05

Figure 5: Pololu budget

Vare	Vare.nr	link	antall	pris	total
koffert clas ohlson	31-4142	https://www.clasohlson.com/no/Koffert/Pr4084	1	kr 599,00	kr 599,00
Marifix blokk med svivel	31-8707	https://www.clasohlson.com/no/Marifix-blokk-n	1	kr 169,00	kr 169,00
Markeringsbøye	31-4737	https://www.clasohlson.com/no/Danfender-mar	1	kr 89,90	kr 89,90
				Total	kr 857,90

Figure 6: Clas Ohlson budget

5 PCB prints main card	$38,07	
5 pcb print opto card	$5,00	
Shipping	$32,14	
Shipping discount	-$20,00	
Total	$55,21	kr 467,57

Figure 7: JLCPCB budget

Name	Price	Link
Storm32	NOK 200,00	https://www.ebay.com/itm/Storm32-BGC-32Bit-3-Axis-Brushless-Gimba
Current sensor	NOK 16,00	https://www.ebay.com/itm/5A-20A-30A-Range-Sensor-ACS712-GY712-2
Total	NOK 216,00	

Figure 8: Ebay budget

Item	RS.no	link	pieces	price	total
TE Connectivity SPDT Non-Latching Relay	680-3593	https://no.rs-online.com/web/p/non-latching-relays/1355111/?releva	6	NOK 19,87	NOK 119,20
DC-DC Converter 12vdc	173-7108	https://no.rs-online.com/web/p/isolated-dc-dc-converters/1736890/	2	NOK 493,52	NOK 987,05
DC-DC Converter 5vdc	706-5906	https://no.rs-online.com/web/p/isolated-dc-dc-converters/1616385/	1	NOK 346,28	NOK 346,28
Sikring 20A FF	188-6849	https://no.rs-online.com/web/p/cartridge-fuses/1886849/	10	NOK 31,11	NOK 311,09
Sikringsholder 20A	360-7294	https://no.rs-online.com/web/p/inline-fuse-holders/3607294/	5	NOK 27,00	NOK 135,00
PG nippler M20 10 Bar	839-5340	https://no.rs-online.com/web/p/cable-glands/8395340/	10	NOK 57,33	NOK 573,30
PG nippler M16 10 Bar	839-5331	https://no.rs-online.com/web/p/cable-glands/8395331/	10	NOK 48,74	NOK 487,40
PG nippler M20	822-9760	https://no.rs-online.com/web/p/cable-glands/8229760/	10	NOK 9,51	NOK 95,12
PG nippler M16 bag of 5	669-4667	https://no.rs-online.com/web/p/cable-glands/6694667/	2	NOK 47,52	NOK 95,04
Strips 200mm	810-3879	https://no.rs-online.com/web/p/cable-ties/8103879/	1	NOK 72,48	NOK 72,48
Strips 100mm	686-8638	https://no.rs-online.com/web/p/cable-ties/6868638/	1	NOK 37,58	NOK 37,58
Til mobilt ekkolodd og GPS på båt:					
Koblingsboks	773-9584	https://no.rs-online.com/web/p/general-purpose-enclosures/773958	1		
Aluminium-profil 1m 40x40mm, 8mm Groove	761-3319	https://no.rs-online.com/web/p/tubing-struts/7613319/	1	NOK 168,38	NOK 168,38
Aluminium-profil 2m 40x80mm, 8mm Groove	761-3325	https://no.rs-online.com/web/p/tubing-struts/7613296/	1	NOK 644,71	NOK 644,71
Aluminiums-profil Pivot Joint 180 grader 8mm Groove	767-5737	https://no.rs-online.com/web/p/connecting-components/7675579/	2	NOK 244,45	NOK 488,90
Aluminiums-profil Joint 90 grader 8mm Groove	767-5695	https://no.rs-online.com/web/p/connecting-components/7675695/	4	NOK 66,00	NOK 264,00
T-Slot Nut, Groove Size 8mm - bag of 10	767-5654	https://no.rs-online.com/web/p/connecting-components/7675654/	2	NOK 60,51	NOK 121,02
M6 12mm bolt - box of 50	232-8271	https://no.rs-online.com/web/p/socket-screws/2328271/	1	NOK 62,40	NOK 62,40
Hovedkort					
Internal fan 25x25x8	482-106	https://no.rs-online.com/web/p/products/0482106/	2		
Cartridge fuse 5A 5x20	610-9721	https://no.rs-online.com/web/p/products/6109721/	10	NOK 2,68	NOK 26,83
Cartridge fuse 10A 5x20	563-643	https://no.rs-online.com/web/p/products/0563643/	10	NOK 1,79	NOK 17,88
Cartridge fuse holder 5x20	176-9047	https://no.rs-online.com/web/p/products/1769047/	10	NOK 4,32	NOK 43,16
PCB terminal connector big	112-695	https://no.rs-online.com/web/p/products/0112695/	5	NOK 6,07	NOK 30,34
Gigabit switch	760-4019	https://no.rs-online.com/web/p/products/7604019/	1		
Car fuse holder	410-7672	https://no.rs-online.com/web/p/products/4107672/	1	NOK 33,44	NOK 33,44
Capasitator 10000 micro Farad	711-1217	https://no.rs-online.com/web/p/products/7111217/	4	NOK 22,23	NOK 88,90
Capasitator 4700 micro Farad	727-0562	https://no.rs-online.com/web/p/products/7270562/	2	NOK 17,05	NOK 34,10
Resistor 2W 100 ohm	707-8827	https://no.rs-online.com/web/p/products/7078827/	10	NOK 0,34	NOK 3,39
PCB terminal connector 2 way small	790-1165	https://no.rs-online.com/web/p/products/7901165/	30	NOK 4,93	NOK 147,96
Enclosure heating element	299-5764	https://no.rs-online.com/web/p/products/2995764/	1	NOK 178,31	NOK 178,31
RS Pro hex standoff 20mm	806-6613	https://no.rs-online.com/web/p/standoffs/8066613/	50	NOK 1,89	NOK 94,30
Machine Screw, M3, 8mm pack of 100	797-6193	https://no.rs-online.com/web/p/machine-screws/7976193/	1	NOK 27,73	NOK 27,73
Vulk teip 25mm	494-433	https://no.rs-online.com/web/p/self-amalgamating-tapes/0494433/	3	NOK 84,59	NOK 253,77
PCB Terminal Block 3 way small	790-1073	https://no.rs-online.com/web/p/pcb-terminal-blocks/7901073/	15	NOK 6,94	NOK 104,04
Quad Optocoupler	691-2233	https://no.rs-online.com/web/p/optocouplers/6912233/	4	NOK 6,68	NOK 26,72
Kasse for batteri					
Hjul	694-8765	https://no.rs-online.com/web/p/castor-wheels/6948765/	2	NOK 197,09	NOK 394,18
Kasse for elektronikk i båt					
RJ45 Plug	916-0255	https://no.rs-online.com/web/p/rj-adapters-couplers-extensions/916	1		
USB plug	916-0221	https://no.rs-online.com/web/p/type-a-usb-connectors/9160221/	1		
Connector 4 way female power	131-6372	https://no.rs-online.com/web/p/industrial-automation-circular-conne	1		
Connector 4 way male power	131-6384	https://no.rs-online.com/web/p/industrial-automation-circular-conne	1		
Connector 4 way 5A female power	124-6654	https://no.rs-online.com/web/p/industrial-automation-circular-conne	1		
Connector 4 way 5A male power	124-6663	https://no.rs-online.com/web/p/industrial-automation-circular-conne	1		
Rekkeklemmer	826-3044	https://no.rs-online.com/web/p/non-fused-din-rail-terminals/826304	10	NOK 5,80	NOK 58,00
Din skinne	467-406	https://no.rs-online.com/web/p/din-rails/0467406/	1		
Endemodul for rekkeklemmer	295-9140	https://no.rs-online.com/web/p/din-rail-terminal-accessories/295914	5	NOK 5,90	NOK 29,50
USB kabel A til A	121-6565	https://no.rs-online.com/web/p/usb-cables/1216565/	1		
Connector 4 way 20A female	178-6537	https://no.rs-online.com/web/p/industrial-automation-circular-conne	1		
Connector 4 way 20A male	178-6512	https://no.rs-online.com/web/p/industrial-automation-circular-conne	1		
M20 Locknut	839-5381	https://no.rs-online.com/web/p/products/8395381/	2	NOK 33,21	NOK 66,42
M16 Locknut	839-5387	https://no.rs-online.com/web/p/products/8395387/	2	NOK 25,65	NOK 51,30
I2C extender card and screws					
M2.5 x 12mm Hex Socket	483-8130	https://no.rs-online.com/web/p/socket-screws/4838130/	1	NOK 216,34	NOK 216,34
PCB switch	204-7865	https://no.rs-online.com/web/p/dip-sip-switches/2047865/	10	NOK 10,91	NOK 109,06
BAT45 Schottky diode	544-4758	https://no.rs-online.com/web/p/rectifier-diodes-schottky-diodes/544	20	NOK 4,37	NOK 87,32
Ceramic Capacitor 100pF	736-8842	https://no.rs-online.com/web/p/ceramic-single-layer-capacitors/736	20	NOK 2,06	NOK 41,28
Ceramic Capacitor 220pF	736-8845	https://no.rs-online.com/web/p/ceramic-single-layer-capacitors/736	20	NOK 1,14	NOK 22,74
PCB terminal connector 2 way small	790-1165	https://no.rs-online.com/web/p/products/7901165/	40	NOK 4,93	NOK 197,28
P82B96P	812-2662	https://no.rs-online.com/web/p/bus-buffers/8122662/	10	NOK 36,89	NOK 368,90
Verktøy					
Thread tap M16	152-205	https://no.rs-online.com/web/p/threading-taps/0152205/	0	NOK 52,50	
Thread tap M20	152-255	https://no.rs-online.com/web/p/threading-taps/0152255/	0	NOK 82,83	
Thread tap M6	152-542	https://no.rs-online.com/web/p/threading-taps/0152542/	0	NOK 16,94	
Blanking Plug M16	406-9390	https://no.rs-online.com/web/p/cable-gland-plugs/4069390/	10	NOK 15,52	NOK 155,22
O-Ring	664-4612	https://no.rs-online.com/web/p/products/6644612/	25	NOK 0,61	NOK 15,33
Syringe	514-717	https://no.rs-online.com/web/p/syringes/0514717/	1	NOK 76,51	NOK 76,51
Kanyle	400-8272	https://no.rs-online.com/web/p/adhesive-dispenser-needles/400827	1	NOK 143,00	NOK 143,00
M6 Threaded rod	175-7012	https://no.rs-online.com/web/p/threaded-rod/1757012/	1	NOK 63,06	NOK 63,06
Nylon Insert Lock Nut	122-4372	https://no.rs-online.com/web/p/locking-nuts/1224372/	200	NOK 0,71	NOK 142,40
M6 Washer	525-947	https://no.rs-online.com/web/p/plain-washers/0525947/	500	NOK 0,19	NOK 96,50
Rubber Gasket	840-5548	https://no.rs-online.com/web/p/natural-rubber-sheets/8405548/	1	NOK 179,05	NOK 179,05
				Total	NOK 8 633,21

Figure 9: Rs Online budget

			Color defenitions:
Today's currency exchange USD/NOK	8,469		Ordered
			Recieved
Personal expenses	kr 1 965,30		Already in stock
			Backorder
			Not recieved
Total:	kr 16 160,21		Not ordered

Figure 10: Total budget

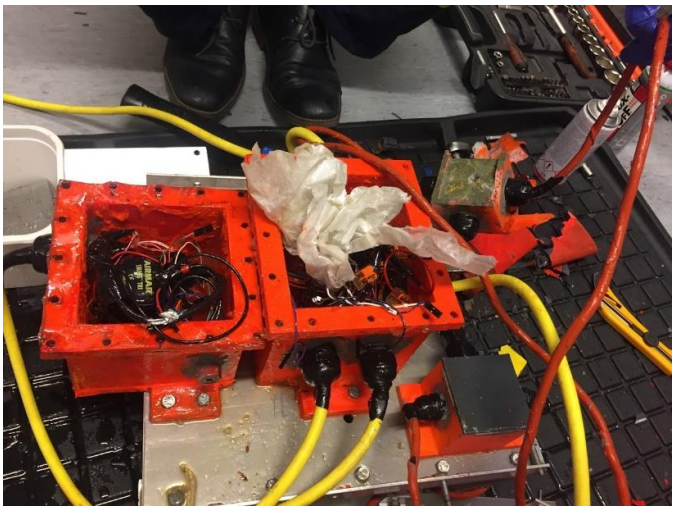
F Weekly reports

Weekly report – Week 4

Date: Thursday 25.01.2019

Work done this week:

- Wrote 95% of the preliminary report.
- Got familiar with LaTeX and Overleaf.
- Completely dismantled and washed all the ROV parts. It was not easy because of how everything was glued together. Everything except the frame of the ROV had to be broken off. So, we dumped everything except the frame. We also cleaned all the electrical equipment that is to be used.



- Found suitable program for 3D mapping of images, Autodesk Recap.
- Researched data from last year's thesis.
- Sketched electrical drawing to get an overview of which parts we need to order.

Issues

- Find a way to build/weld the boxes for the actuators. Maybe ask André Tranvåg.
- Get all 3D models and source code from last year. Still waiting for these…

Plan forward:

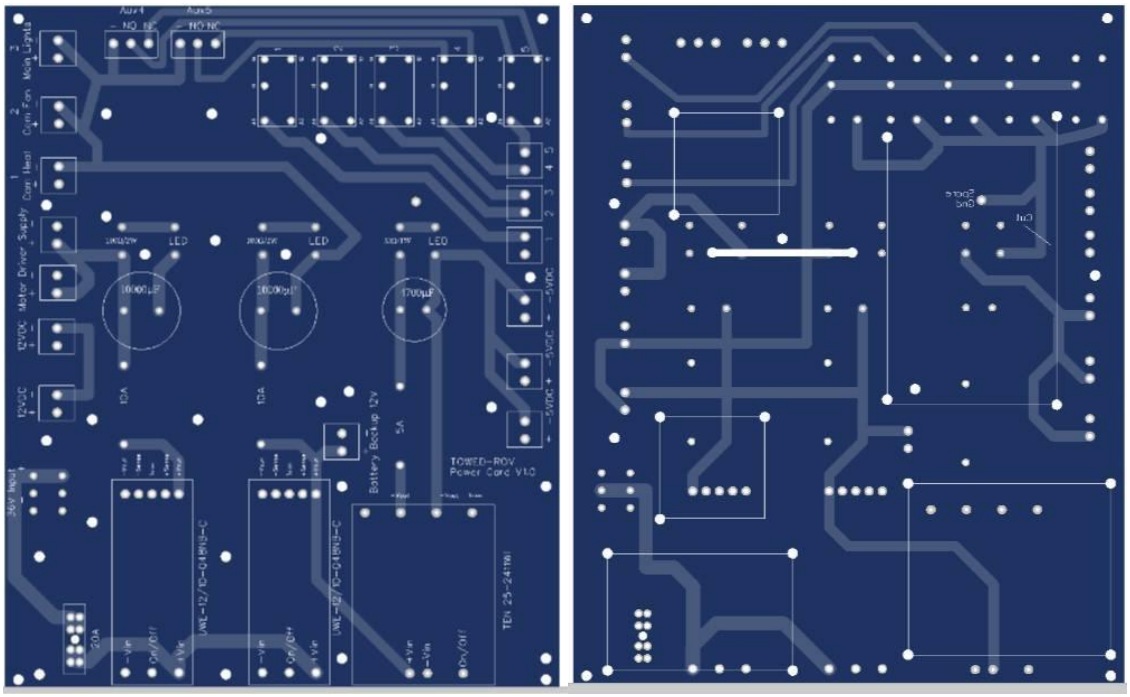
- Finish preliminary report next week. Write risk mitigation plan.
- Maybe find pre-made boxes for actuators?
- Consider placing a box for sensor equipment behind the cylindrical el. box instead of having it together with everything else. This to make it easier to implement new sensors/make changes.
- Continue with planned tasks.

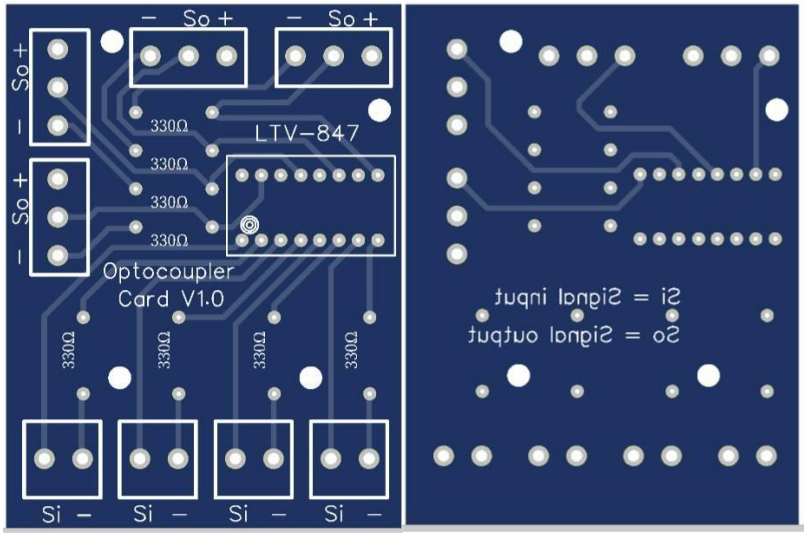
Weekly report – Week 5

Date: Thursday 01.02.2019

Work done this week:

- Finished the preliminary report.
- Sketched electrical drawings and did a lot of planning and sketching to get an overview of which parts we need to order, and where it is supposed to be mounted on the ROV.
- Tested the echo sounder through an Arduino. Code working as it should.
- Tested and got the PWM control of the Blue Robotics lights working.
- Used EasyEDA to design a main circuit board for most of the electronics that is supposed to be in the ROV. Also designed a circuit board for controlling the echo sounders and lights through opto-isolators.





Issues

- [In progress] Find a way to build/weld watertight boxes.

Plan forward:

- Start to look at the PID/fuzzy logic, with simulation.
- Reading of sensor values.
- Communication
- Continue with planned tasks.

Weekly report – Week 6

Date: Friday 08.02.2019

Work done this week:

- Started working on the code. Communication, sensor readings etc.
- Set up the main raspberry pi.
- Ordered parts, received from RS and Blue Robotics.
- Researching PID/Fuzzy.
- Built an early concept of PID simulation in Simulink.
- Weekly meeting with supervisors.
- Got double/triple of everything from RS, so we needed to go through every single item and send it back.

Issues

- [In progress] Find a way to build/weld watertight boxes.

Plan forward:

- Communication.
- Work on the Java code.
- Continue with planned tasks.

Weekly report – Week 8

Date: Friday 22.02.2019

Work done this week:

- 3D modelling of all boxes, including making mechanical drawings of them.
- Designing I2C bus buffer and PCB for this.
- I2C research and testing code/bus buffer.

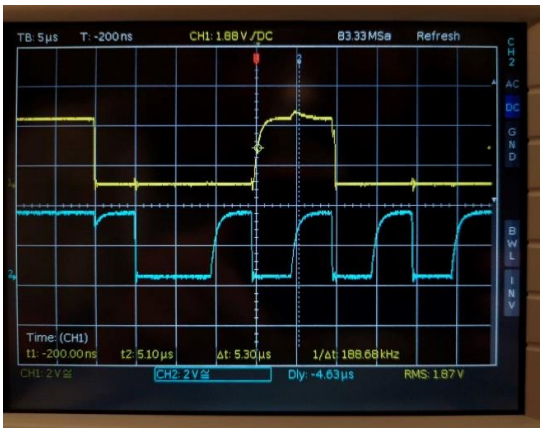


Figure 1 - I2C over 3m cable WITHOUT bus buffer.

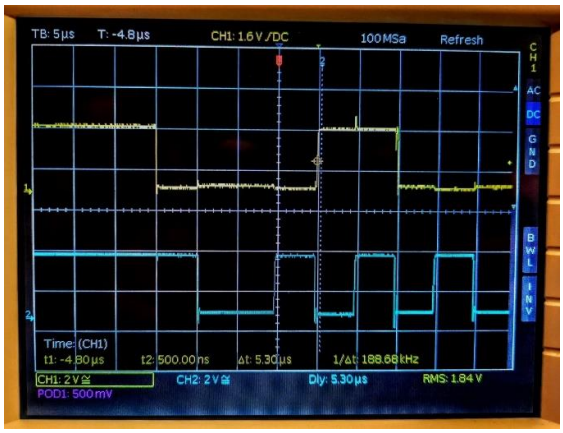


Figure 2 - I2C over 3m cable WITH bus buffer

- Communication between Arduino and Java, optimizing the code.
- Fault finding and optimizing GPS signal code.
- Implementing the code from the echo sounder.
- A lot of research regarding different methods to do the video stream, with UDP communication to Java. Setting up the raspberry pi and testing some of them. Still working on this.

Plan forward:

- Communication.
- Work on the Java code.
- Test I2C noise.
- Video Stream.
- Sensor input.
- Continue with planned tasks.

Weekly report – Week 9 and 10

Date: Friday 08.03.2019

Work done these two weeks:

- Researched how we can laser cut the boxes instead of metal. We have made a test box which is ready to be cut, that will be tested by lowering it 50m in the ocean. This is to test condensation.
- To reduce condensation our plan was to place a small circulation fan and a 15W heating element in the camera housing. We built a test rig of the camera housing, an Arduino with a temperature sensor, and power to the heating element and fan. We tested this outside in -2 degrees Celsius (air temp) which resulted in the temperature in the camera housing dropping too fast/too low. A bigger heating element might be needed.
- Had a lot of issues with the Arduino sensor reading code. There was a problem when processing the data from the Arduino. This took a lot of time but is now working fine.
- Overhauled the actuators because of the oil have washed away the gear grease.
- Tested the I2C bus buffer together with the actuators. We had an issue with the actuator, where one of the gears got loose because of the overhaul, so this is not quite finished yet.
- The process of installing OpenCV on the raspberry took longer than expected. This was due to the installation process was constantly crashing. We had taken precautions by placing a heat sink on the processor along with a cooling fan which might have helped a little bit, at least against overheating. Eventually we managed to finish the installation, and now the video stream works from the RPi to Java over UDP, with 20fps and an average 0.05 sec delay between each frame which gives around 1 second delay to the eye. We tried to speed this up by duplicating the SD card on to a 9x faster SD card, but we did not notice any significant changes.
- Week 10 (wed, thu, fri) was focused on Industry 4.0.

Issues

- We have waited 2 weeks for access to the FAB-lab. We need to start the process of making the boxes as soon as possible. As soon as the boxes are finished, we are ready to start the assembly.
- An issue we are worried about is the food oil becoming rancid when exposed to light. The plexi glass boxes might speed this up. Do we need to look at alternative oil types than food oil?
- While testing the actuators we have an issue regarding the brush motors being surrounded by oil.

Plan forward:

- Finish the boxes and start assembling the ROV.
- Work on the Java code.
- Start rebuilding the GUI and integrate the new video stream.
- Continue with planned tasks.

Weekly Report – Week 12

Date 15.03.2019

Work done this week:

- Built rollable case for batteries
- Started Wiring plug and play system for base station
- Started coding alarm system
- 3D models of different parts
- Finished the drawings for the Plexiglass boxes
- Video stream integration with GUI done and working.
- Working on refurbishing the GUI.

Plan forward:

- Finish plug and play system
- Finish java code for base station
- Full test of signals from base station
- Laser cut plexiglass and glue together
- Pressure test plexiglass boxes
- Finish camera housing
- GUI
- Continue with planned tasks

Weekly Report – Week 14

Date 04.04.2019

Work done this week:

- Finished the base station case.
- Finished Wiring plug-and-play system for base station
- 90% finished with the GUI.
- Added functionality for Photo Mode/Video Mode, where in Photo Mode the ROV takes still photos in 3280x2464 and saves it on the SD card which is shared through an FTP server (free space for 1200 photos). By only one click, the photos are retrieved from the FTP server to the PC (GUI).
- Started to integrate all the different software systems.
- 3D modelling of various parts.
- Laser cut the Echo sounder box. Looks good.
- Started on building the main electrical system for the ROV.
- Heading value from the Razor IMU has shown to be an issue. Other groups also have problem with this. The IMU is very sensitive to magnetic noise, which makes the calibration process challenging.
- Tested the TCP communication over the 60m utility cable. This gave approximately 300 messages per second, with 2ms delay. The transfer capacity of the system is approximately 80 Mbit.

Issues

- Unexpected long waiting time for the cable and the acrylic plates. Also, the first order of the acrylic plates we got was 6mm instead of 8mm... We need this to start building the ROV…
- The gimbal motors we got from Anders was not working. New ones had to be ordered. Because of this the camera house is not yet assembled.

Plan forward:

- Finish plug and play system
- Finish java code for base station
- Full test of signals from base station
- Laser cut plexiglass and glue together
- Pressure test plexiglass boxes
- Finish camera housing
- Build ROV
- Use Robins boat for testing
- Continue with planned tasks

Weekly Report – Week 15 and 16

Date 19.04.2019

Work done these weeks:

- Finished the plug and play system.
- Laser cut all the boxes and started to glue them together. It showed that the diameter of the laser was a bit too wide, which made the gap between each element of the box a little bit too big. This made it hard to be 100% sure the glue would make it waterproof. But this was as expected, and we solved this using transparent silicone. We tested them by filling them with water, and after some small leakages were found, the sealing was good.
- Continued building the electrical system of the ROV.
- Tested controlling i2c and GPIO devices on the raspberry. Working well.
- Almost finished all the python code on the raspberry pi in the camera housing. Also implemented a multithreaded program for reading multiple sensors data and a TCP server.
- We also took a couple days off in week 16 for Easter holiday.

Issues

- Building the boxes took a bit longer than expected.

Plan forward:

- Finish java code for base station
- Finish camera housing
- Build ROV
- Use Robins boat for testing
- Continue with planned tasks

Weekly Report – Week 17

Date 26.04.2019

Work done this week:

- Finish Java code for base station sensors.
- Finished the adjustable echo sounder mounting system.
- Assembled most of the ROV body.
- We did some buoyancy calculations after assembling the ROV and bought some pipes for additional buoyancy. We mounted some of these at the bottom of the ROV, and some of them at the top for stability.
- Expanded the GUI with a lot of new features for controlling the camera pitch, lights, manual wing control, live graph of the rov and sea depth, and for managing the images on the camera RPi, as well as the communication between the camera RPi, GUI and the ROV RPi. Also added an alarm sound if an emergency alarm goes off. All these features were also tested and are working well.
- Programmed all the communication of the feedbacks and IO through the system.
- We had a problem with not receiving all the data from the echo sounder Arduino. This was probably because of clock stretching, which makes things difficult for us. We decided to move the Arduino for IO to the main electronics box and connect it by USB to the raspberry pi. This makes it easy to reprogram through the RPi.
- Various 3D-modelling and printing.

Plan forward:

- Finish camera housing
- Build ROV
- Use Robins boat for testing
- Continue with planned tasks

Weekly Report – Week 18

Date 03.05.2019

Work done this week:

- We had a lot of issued with reading the actuator feedback from the pololu boards directly into the raspberry pi. We did not think this was because of noise on the bus, since every command we sent was 100% received. We researched this and found out that the version of Arduino Nano we had did not cooperate well with the pololu board on i2c. We replaced this with an Arduino Uno which solved our problems. We then decided to move the IO to this Arduino as well, and let the Nano only handle the echo sounder data.
-

Issues

- The actuators are not suitable for this kind of environment, especially since they have brushes. The oil from last year has made them slow, inaccurate, and has eaten up the gaskets. We have spent a lot of time overhauling these

Plan forward:

- Finish java code for base station
- Finish camera housing
- Build ROV
- Use Robins boat for testing
- Continue with planned tasks

Weekly Report – Week 19

Date 03.05.2019

Work done this week:

- The week started with finishing the code and making the ROV ready for sea trail.
- On Tuesday we built the ROV and filled the boxes with rapeseed oil.
- On Wednesday morning, we went over the boxes, and could not find any leaks. Afterwards we packed everything in our cars and travelled to Store Kalvoy by boat. We spent a lot of time setting up and testing everything before we took Robins boat out and performed a complete test. At the end of the day, right before we were supposed to do a complete test at sea, the starboard actuator went bad. The speed of it dropped, so much that it was unusable. We decided lock the actuators in dive position and regulate the depth by the speed of the boat. This worked really well, and the stability of the ROV was really good.
- On Thursday we showcased the ROV at the Ocean Week event at NMK. We got a lot of positive feedback, and people were really impressed that we had done this much work in such a short time.
- Because of the faulty actuator, a second test day is not possible. The rest of the bachelor period we will be focusing on finish writing the report.

Issues

- The actuators are not suitable for this kind of environment, especially since they have brushes. The oil from last year has made them slow, inaccurate, and has eaten up the gaskets. We have spent a lot of time overhauling these.

Plan forward:

- Finish writing the thesis report

G Meeting reports

Report meeting with advisors

Date: Thursday 24.01.19
Place: Rundskue 702_D318, Main building, NTNU Aalesund

Present: Ottar, Håkon, Bjørnar, Robin
Time: 10:30 – 11:00

Issues

- Find a way to build/weld the boxes for the actuators. Maybe ask André Tranvåg.
- Get all 3D models and source code from last year.

Plan forward:

- Finish preliminary report next week. Write risk mitigation plan.
- Maybe use plastic boxes for actuators?
- Consider placing a box for sensor equipment behind the cylindrical el. box instead of having it together with everything else. This to make it easier to implement new sensors/make changes.
- Continue with planned tasks.

Meeting was adjourned at 10:59 by Robin. The next meeting will be held at 10:30 - 11:00 on February 7, 2018, at NTNU Aalesund.

Approved by:
Håkon Longva Haram

Report meeting with advisors

Date: Thursday 07.02.19
Place: Rundskue 702_D318, Main building, NTNU Aalesund

Present: Ottar, Håkon, Bjørnar, Robin
Time: 10:30 – 11:00

Topics

- Find a way to build/weld the boxes for the actuators. Maybe ask André Tranvåg.
- Ottar recommended to use oil in the boxes, instead of trying to do it without. Plastic boxes, or thin metal.
- Make the ROV design longer.
- Maybe add floaters to get positive buoyancy.
- Consider placing an extra battery on the ROV to counter voltage drops from the base station.

Plan forward:

- Continue with planned tasks.

Meeting was adjourned at 11:04 by Robin. The next meeting will be held at 10:30 - 11:00 on February 21, 2018, at NTNU Aalesund.

Approved by:
Håkon Longva Haram

Report meeting with advisors

Date: Thursday 21.02.19
Place: Rundskue 702_D318, Main building, NTNU Aalesund

Present: Ottar, Håkon, Bjørnar, Robin
Time: 10:30 – 11:00

Topics

- Revise the 3D modelled boxes together with Ottar.
- Talk to Paul Steffen about making the design longer. A new laser cutter will be installed in the basement in the Lab building. Wait for this, and use it to laser cut the boxes and walls of the ROV.
- I2C communication tested over long cable WORKING. **Test with actuators running**
- Oppdrift i vingane? Vakuumtrekke? Frese vingane i skom? Lage vingane med hull i? **Lage flate vinga i akryl?**
-

Plan forward:

- Continue with planned tasks.

Meeting was adjourned at 10:59 by Robin. The next meeting will be held at 10:30 - 11:00 on March 7, 2018, at NTNU Aalesund.

Approved by:
Håkon Longva Haram

Report meeting with advisors

Date: Thursday 21.03.19
Place: Rundskue 702_D318, Main building, NTNU Aalesund

Present: Ottar, Paul Steffen, Håkon, Bjørnar, Robin
Time: 10:30 – 11:00

Topics

- Ask the guy whos coming to install the laser cutter about Autocad/Fusion integration. We used Corell Draw.
- If time, maybe simulate the hydrodynamics of the ROV model.
- Maybe calculate how long we can operate the ROV from the load. Discharging curve for the batteries and find a way to estimate the energy in the batteries based on the voltage. Find battery %, and the point where the voltage drops fast.

Plan forward:

- Continue with planned tasks.

Meeting was adjourned at 11:00 by Robin. The next meeting will be held at 10:30 - 11:00 on April 4th, 2018, at NTNU Aalesund.

Approved by:
Håkon Longva Haram

Report meeting with advisors

Date: Thursday 25.04.19
Place: L160, Lab building, NTNU Aalesund

Present: Ottar, Paul Steffe, Håkon, Bjørnar, Robin
Time: 13:00 – 13:30

Topics

- Add buoyancy elements to make it float. Calculate the need and go to MegaFlis and buy.
- Important to log as much data as possible. Implement this.
- Discussed the type of oil. The advisors agreed that rapeseed oil is good.
- Went over the overall build of the ROV. Everything looks good.
- Discussed the actuators. These are not well suited for being coved in oil for a long period of time. This has made the actuators faulty, which we have used a lot of time to overhaul and fix.

Plan forward:

- Finish building the ROV.
- Finish writing the thesis.

Meeting was adjourned at 13:35 by Robin. The next meeting will be held at 13:00 - 13:30 on May 9th, 2018, at NTNU Aalesund.

Approved by:
Håkon Longva Haram

Report meeting with advisors

Date: Thursday 09.05.19
Place: L160, Lab building, NTNU Aalesund

Present: Ottar, Paul Steffe, Håkon, Bjørnar, Robin
Time: 13:00 – 13:30

Topics

- Showed the video from the test day, along with all the data we gathered. The results were very positive.
- Not possible to do a second test because of the bad actuator. These actuators have shown to be a problem. A possible solution would be to replace these actuators with stepper motors.

Plan forward:

- Finish writing the thesis.

Meeting was adjourned at 13:35 by Robin. This was the last meeting for this bachelor thesis.

Approved by:
Håkon Longva Haram

H Electrical drawings

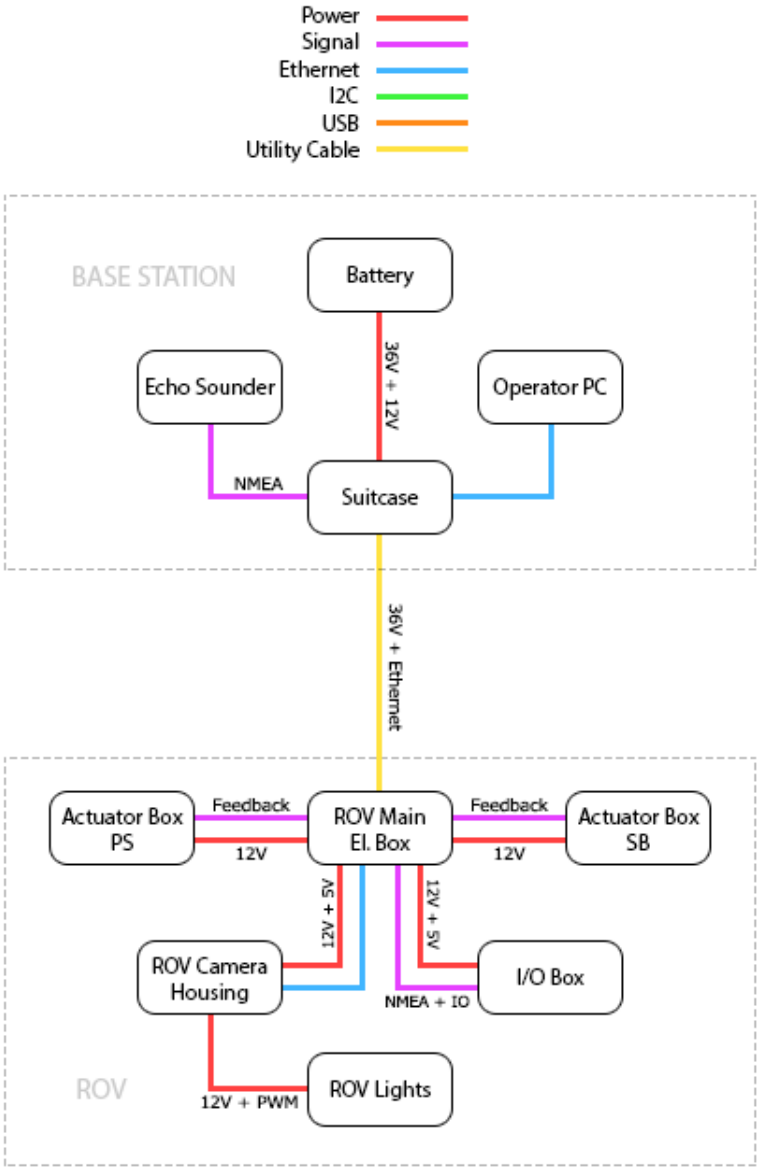


Figure 11: One Line Diagram - MAIN.

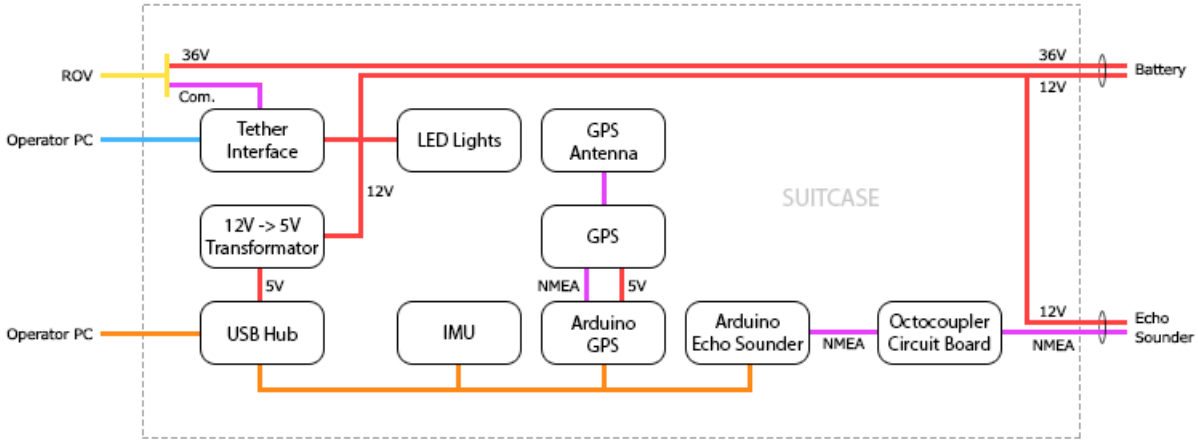


Figure 12: One Line Diagram - SUITCASE.

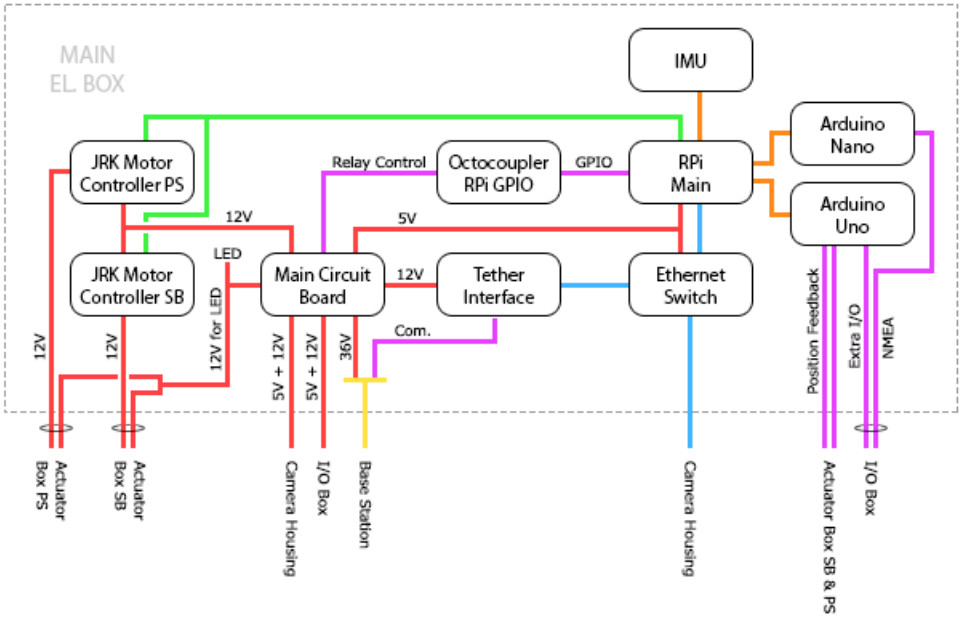


Figure 13: One Line Diagram - MAIN EL BOX.

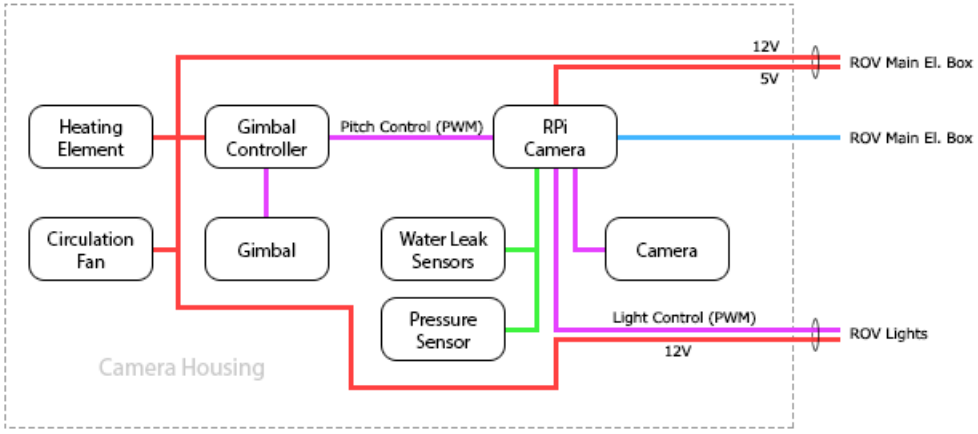


Figure 14: One Line Diagram - CAMERA HOUSING.

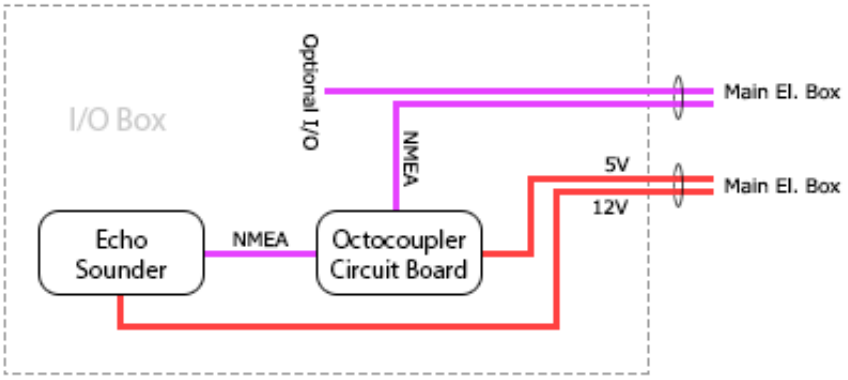
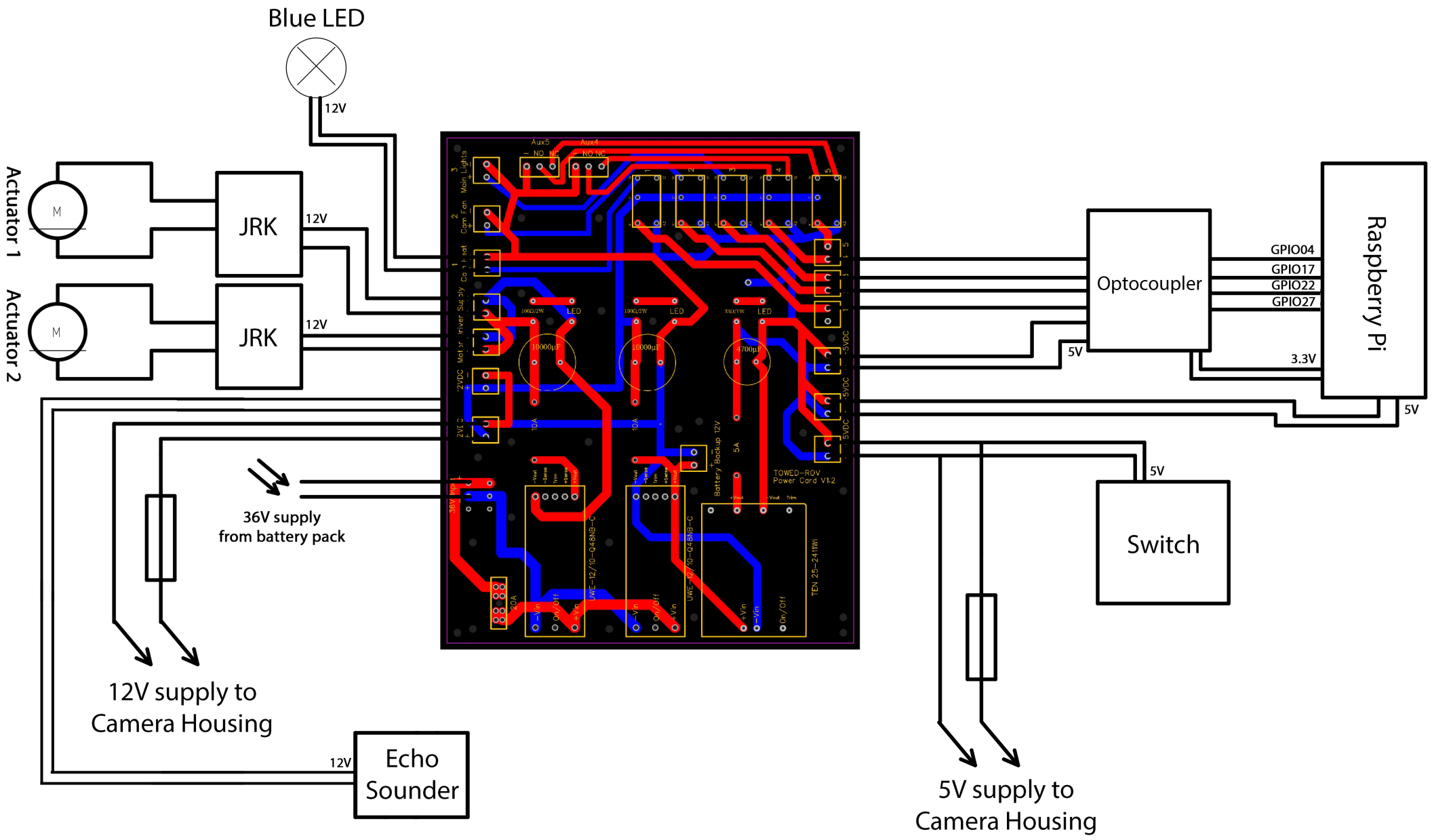


Figure 15: One Line Diagram - I/O BOX.


Blue LED
12V
Actuator 1
Actuator 2
M
M
JRK
JRK
12V
12V
Aux5
Aux4
NO NC
NO NC
Main Lights
Cam Fan
Cam 1+ 5VDC
Cam 2+ 5VDC
Motor Driver Supply
12VDC
12VDC
LED
LED
LED
10000µF
10000µF
4700µF
10A
10A
5A
Battery Backup 12V
TOWED-ROV
Power Card V1.2
36V supply
from battery pack
12V supply to
Camera Housing
12V
Echo
Sounder
Optocoupler
5V
3.3V
GPIO04
GPIO17
GPIO22
GPIO27
Raspberry Pi
5V
5V
Switch
5V supply to
Camera Housing

I Source code

J GUI Java code

D:\Dokumenter\Skule\04 - NTNU\Bachelor\Github\TowedROV_GUI\src\ntnusubsea\gui\NTNUSubseaGUI.java

```
1  /*
2   * This code is for the bachelor thesis named "Towed-ROV".
3   * The purpose is to build a ROV which will be towed behind a surface vessel
4   * and act as a multi-sensor platform, were it shall be easy to place new
5   * sensors. There will also be a video stream from the ROV.
6   *
7   * The system consists of two Raspberry Pis in the ROV that is connected to
8   * several Arduino micro controllers. These micro controllers are connected to
9   * feedback from the actuators, the echo sounder and extra optional sensors.
10  * The external computer which is on the surface vessel is connected to a GPS,
11  * echo sounder over USB, and the ROV over ethernet. It will present and
12  * log data in addition to handle user commands for controlling the ROV.
13  */
14 package ntnusubsea.gui;
15
16 import basestation_rov.LogFileHandler;
17 import basestation_rov.ReadSerialData;
18 import java.util.Map.Entry;
19 import java.util.concurrent.Executors;
20 import java.util.concurrent.ScheduledExecutorService;
21 import java.util.concurrent.TimeUnit;
22 import javax.swing.SwingUtilities;
23 import basestation_rov.SerialDataHandler;
24 import InputController.InputController;
25
26 /**
27  * Main class that launches the application and schedules the different threads
28  */
29 public class NTNUSubseaGUI {
30 //
31 //    private static Thread readSerialData;
32 //    private static Thread imuThread;
33 //    private static Thread gpsThread;
34 //    private static Thread echoSounderThread;
35 //    private static Thread ROVDummyThread;
36
37    private static Thread InputControllerThread;
38    private static Thread comPortFinderThread;
39
40    protected static String ipAddress = "localHost";
41    protected static int sendPort = 5057;
42    protected static String IP_ROV = "192.168.0.101";
43    protected static String IP_camera = "192.168.0.102";
44    protected static int Port_ROV = 8080;
45    protected static int Port_cameraStream = 8083;
46    protected static int Port_cameraCom = 9006;
47    protected static ROVFrame frame;
48
49    /**
50     * Main class that launches the application and schedules the different
51     * threads
52     *
53     * @param args the command line arguments
54     */
55    public static void main(String[] args) {
56
57        Data data = new Data();
58        Sounder sounder = new Sounder();
59        SerialDataHandler sdh = new SerialDataHandler(data);
60        EchoSounderFrame sonar = new EchoSounderFrame(data);
61        LogFileHandler lgh = new LogFileHandler(data);
```

```
62      TCPpinger client_Pinger = new TCPpinger(IP_ROV, Port_ROV, data);
63      TCPClient client_ROV = new TCPClient(IP_ROV, Port_ROV, data);
64      TCPClient client_Camera = new TCPClient(IP_camera, Port_cameraCom, data);
65      UDPServer stream = new UDPServer(Port_cameraStream, data);
66      IOControlFrame io = new IOControlFrame(data, client_ROV);
67      frame = new ROVFrame(sonar, data, io, client_Pinger, client_ROV, client_Camera, stream, sounder, lgh);
68      DataUpdater dataUpdater = new DataUpdater(client_ROV, client_Camera, data);
69
70      ScheduledExecutorService executor
71              = Executors.newScheduledThreadPool(8);
72      SwingUtilities.invokeLater(frame);
73      SwingUtilities.invokeLater(io);
74      sonar.setVisible(false);
75      data.addObserver(sonar);
76      data.addObserver(frame);
77      data.addObserver(io);
78      executor.scheduleAtFixedRate(lgh,
79              0, 100, TimeUnit.MILLISECONDS);
80      executor.scheduleAtFixedRate(sonar,
81              0, 100, TimeUnit.MILLISECONDS);
82      executor.scheduleAtFixedRate(dataUpdater,
83              1000, 100, TimeUnit.MILLISECONDS);
84
85      InputControllerThread = new Thread(new InputController(data, client_ROV));
86      InputControllerThread.start();
87      InputControllerThread.setName("InputController");
88
89      comPortFinderThread = new Thread(new ComPortFinder(sdh, data));
90      comPortFinderThread.start();
91      comPortFinderThread.setName("ComPortFinder");
92
93  //      // Start searching for com ports:
94  //      long timeDifference = 0;
95  //      long lastTime = 0;
96  //      long timeDelay = 5000;
97  //      boolean connected = false;
98  //      boolean foundComPort = false;
99  //      boolean listedCom = false;
100 //
101 //       while (true)
102 //       {
103 //
104 //          if (!foundComPort)
105 //
106 //          {
107 //              System.out.println("Searching for com ports...");
108 //              sdh.findComPorts();
109 //              foundComPort = true;
110 //          }
111 //
112 //          if (!listedCom)
113 //          {
114 //              System.out.println("Com ports found:");
115 //
116 //              if (data.comPortList.isEmpty())
117 //              {
118 //                  System.out.println("None");
119 //              } else
120 //              {
121 //                  for (Entry e : data.comPortList.entrySet())
122 //                  {
123 //                      String comPortKey = (String) e.getKey();
124 //                      String comPortValue = (String) e.getValue();
125 //                      System.out.println(comPortKey + " : " + comPortValue);
```

```
126 //
127 //                }
128 //            }
129 //            System.out.println("--End of com list--");
130 //            listedCom = true;
131 //
132 //            for (Entry e : data.comPortList.entrySet())
133 //            {
134 //                String comPortKey = (String) e.getKey();
135 //                String comPortValue = (String) e.getValue();
136 //                if (comPortValue.contains("IMU"))
137 //                {
138 //                    imuThread = new Thread(new ReadSerialData(data, comPortKey, 115200, comPortValue));
139 //                    imuThread.start();
140 //                    imuThread.setName(comPortValue);
141 //
142 //                }
143 //
144 //                if (comPortValue.contains("GPS"))
145 //                {
146 //                    gpsThread = new Thread(new ReadSerialData(data, comPortKey, 115200, comPortValue));
147 //                    gpsThread.start();
148 //                    gpsThread.setName(comPortValue);
149 //
150 //                }
151 //
152 //                if (comPortValue.contains("EchoSounder"))
153 //                {
154 //                    echoSounderThread = new Thread(new ReadSerialData(data, comPortKey, 4800, comPortValue));
155 //                    echoSounderThread.start();
156 //                    echoSounderThread.setName(comPortValue);
157 //                }
158 //
159 //                if (comPortValue.contains("ROVDummy"))
160 //                {
161 //                    ROVDummyThread = new Thread(new ReadSerialData(data, comPortKey, 115200, comPortValue));
162 //                    ROVDummyThread.start();
163 //                    ROVDummyThread.setName(comPortValue);
164 //                }
165 //            }
166 //        }
167 //    }
168     }
169 }
170
```

D:\Dokumenter\Skule\04 - NTNU\Bachelor\Github\TowedROV_GUI\src\basestation_rov\calibrationClasses\ActuatorCalibration.java

```
1  /*
2   * This code is for the bachelor thesis named "Towed-ROV".
3   * The purpose is to build a ROV which will be towed behind a surface vessel
4   * and act as a multi-sensor platform, were it shall be easy to place new
5   * sensors. There will also be a video stream from the ROV.
6   *
7   * The system consists of two Raspberry Pis in the ROV that is connected to
8   * several Arduino micro controllers. These micro controllers are connected to
9   * feedback from the actuators, the echo sounder and extra optional sensors.
10  * The external computer which is on the surface vessel is connected to a GPS,
11  * echo sounder over USB, and the ROV over ethernet. It will present and
12  * log data in addition to handle user commands for controlling the ROV.
13  */
14 package basestation_rov.calibrationClasses;
15
16 import ntnusubsea.gui.*;
17
18 /**
19  * Responsible for calibrating the actuators. Not currently in use/finished.
20  */
21 public class ActuatorCalibration implements Runnable {
22
23     private final static String actuatorEndPos = "254";
24     private final static String actuatorMiddlePos = "127";
25     private final static String actuatorStartPos = "1";
26     private final static int accuracy = 4;
27     private int posChangeTime = 8000;  // millisecond
28     private int lastActuatorPSPos = 0;
29     private int lastActuatorSBPos = 0;
30     private boolean findingMinPS = true;
31     private boolean findingMinSB = true;
32     private boolean findingMaxPS = true;
33     private boolean findingMaxSB = true;
34     private long currentTime = 0;
35     private long lastTimePS = 0;
36     private long lastTimeSB = 0;
37     private long PSActuatorMaxToMinTime = 0;
38     private long SBActuatorMaxToMinTime = 0;
39
40     //Error List
41     //PS actuator
42     private boolean Error_PSActuatorNotInMinPos = false;
43     private boolean Error_PSActuatorNotInMaxPos = false;
44     private boolean Error_PSActuatorTooSlow = false;
45
46     //SB actuator
47     private boolean Error_SBActuatorNotInMinPos = false;
48     private boolean Error_SBActuatorNotInMaxPos = false;
49     private boolean Error_SBActuatorTooSlow = false;
50
51     private Data data;
52     private TCPClient client_ROV;
53
54     /**
```

```
55   * The constructor of the ActuatorCalibration class.
56   *
57   * @param data the shared resource Data class
58   * @param client_ROV The ROV TCP client
59   */
60  public ActuatorCalibration(Data data, TCPClient client_ROV) {
61      this.data = data;
62      this.client_ROV = client_ROV;
63  }
64
65  /**
66   * Runs the ActuatorCalibration thread.
67   */
68  @Override
69  public void run() {
70      calibrateMinPos();
71      speedFromMinToMax();
72  }
73
74  /**
75   * Calibrates the minimum position of the actuator.
76   */
77  private void calibrateMinPos() {
78      try {
79          client_ROV.sendCommand("<cmd_actuatorPS:" + actuatorStartPos + ">");
80          client_ROV.sendCommand("<cmd_actuatorSB:" + actuatorStartPos + ">");
81          boolean waitingForPSPosChange = true;
82          boolean waitingForSBPosChange = true;
83          lastTimePS = System.currentTimeMillis();
84          lastTimeSB = System.currentTimeMillis();
85          while (waitingForPSPosChange || waitingForSBPosChange) {
86              if (System.currentTimeMillis() - lastTimePS >= posChangeTime || !waitingForPSPosChange) {
87                  client_ROV.sendCommand("<cmd_actuatorPS:" + data.getFb_actuatorPSPos() + ">");
88                  System.out.println("Error: PS_Actuator did not reach min pos in time");
89                  waitingForPSPosChange = false;
90                  Error_PSActuatorNotInMinPos = true;
91              }
92              if (data.getFb_actuatorPSPos() < 3) {
93                  System.out.println("PS actuator in position");
94                  waitingForPSPosChange = false;
95              }
96
97              if (System.currentTimeMillis() - lastTimeSB >= posChangeTime || !waitingForPSPosChange) {
98                  client_ROV.sendCommand("<cmd_actuatorSB:" + data.getFb_actuatorSBPos() + ">");
99                  System.out.println("Error: SB_Actuator did not reach min pos in time");
100                 waitingForSBPosChange = false;
101                 Error_PSActuatorNotInMinPos = true;
102             }
103             if (data.getFb_actuatorSBPos() < 3) {
104                 System.out.println("SB actuator in position");
105                 waitingForSBPosChange = false;
106             }
107         }
108
109     } catch (Exception e) {
110     }
111 }
112
```

```
113   /**
114    * Finds the speed from minimum to maximum position.
115    */
116   private void speedFromMinToMax() {
117       try {
118
119           client_ROV.sendCommand("<cmd_actuatorPS:" + actuatorEndPos + ">");
120           client_ROV.sendCommand("<cmd_actuatorSB:" + actuatorEndPos + ">");
121
122           boolean waitingForPSPosChange = true;
123           boolean waitingForSBPosChange = true;
124           lastTimePS = System.currentTimeMillis();
125           lastTimeSB = System.currentTimeMillis();
126           while (waitingForPSPosChange || waitingForSBPosChange) {
127               if (System.currentTimeMillis() - lastTimePS >= posChangeTime || !waitingForPSPosChange) {
128                   client_ROV.sendCommand("<cmd_actuatorPS:" + data.getFb_actuatorPSPos() + ">");
129                   System.out.println("Error: PS_Actuator did not reach max pos in time");
130                   waitingForPSPosChange = false;
131                   Error_PSActuatorNotInMaxPos = true;
132               }
133               if (data.getFb_actuatorPSPos() > 250) {
134                   data.setPSActuatorMaxToMinTime(System.currentTimeMillis() - lastTimePS);
135                   System.out.println("PS actuator in position");
136                   waitingForPSPosChange = false;
137               }
138
139               if (System.currentTimeMillis() - lastTimeSB >= posChangeTime || !waitingForSBPosChange) {
140                   client_ROV.sendCommand("<cmd_actuatorSB:" + data.getFb_actuatorSBPos() + ">");
141                   System.out.println("Error: SB_Actuator did not reach max pos in time");
142                   waitingForSBPosChange = false;
143                   Error_SBActuatorNotInMaxPos = true;
144               }
145               if (data.getFb_actuatorSBPos() > 250) {
146                   data.setSBActuatorMaxToMinTime(System.currentTimeMillis() - lastTimeSB);
147                   System.out.println("SB actuator in position");
148                   waitingForPSPosChange = false;
149               }
150           }
151       } catch (Exception e) {
152       }
153   }
154 }
155
```

D:\Dokumenter\Skule\04 - NTNU\Bachelor\Github\TowedROV_GUI\src\ntnusubsea\gui\ComPortFinder.java

```
1 /*
2  * This code is for the bachelor thesis named "Towed-ROV".
3  * The purpose is to build a ROV which will be towed behind a surface vessel
4  * and act as a multi-sensor platform, were it shall be easy to place new
5  * sensors. There will also be a video stream from the ROV.
6  *
7  * The system consists of two Raspberry Pis in the ROV that is connected to
8  * several Arduino micro controllers. These micro controllers are connected to
9  * feedback from the actuators, the echo sounder and extra optional sensors.
10  * The external computer which is on the surface vessel is connected to a GPS,
11  * echo sounder over USB, and the ROV over ethernet. It will present and
12  * log data in addition to handle user commands for controlling the ROV.
13  */
14 package ntnusubsea.gui;
15
16 import basestation_rov.ReadSerialData;
17 import basestation_rov.SerialDataHandler;
18 import java.util.Map;
19
20 /**
21  * The constructor of the ComPortFinder class.
22  */
23 public class ComPortFinder implements Runnable {
24
25     private static Thread imuThread;
26     private static Thread gpsThread;
27     private static Thread echoSounderThread;
28     private static Thread ROVDummyThread;
29     private SerialDataHandler sdh;
30     private Data data;
31
32     /**
33      * The constructor of the ComPortFinder class.
34      *
35      * @param sdh the given SerialDataHandler
36      * @param data the shared resource class Data
37      */
38     public ComPortFinder(SerialDataHandler sdh, Data data) {
39         this.sdh = sdh;
40         this.data = data;
41     }
42
43     /**
44      * Runs the ComPortFinder thread. Starts the necessary threads based on the
45      * com ports connected.
46      */
47     @Override
48     public void run() {
49
50         // Start searching for com ports:
51         long timeDifference = 0;
52         long lastTime = 0;
53         long timeDelay = 5000;
54         boolean connected = false;
55         boolean foundComPort = false;
56         boolean listedCom = false;
57
58         //while (true) {
59         if (!foundComPort) {
```

```
60          System.out.println("Searching for com ports...");
61          sdh.findComPorts();
62          foundComPort = true;
63      }
64
65      if (!listedCom) {
66          System.out.println("Com ports found:");
67
68          if (data.comPortList.isEmpty()) {
69              System.out.println("None");
70          } else {
71              for (Map.Entry e : data.comPortList.entrySet()) {
72                  String comPortKey = (String) e.getKey();
73                  String comPortValue = (String) e.getValue();
74                  System.out.println(comPortKey + " : " + comPortValue);
75              }
76          }
77          System.out.println("--End of com list--");
78          listedCom = true;
79
80          for (Map.Entry e : data.comPortList.entrySet()) {
81              String comPortKey = (String) e.getKey();
82              String comPortValue = (String) e.getValue();
83              if (comPortValue.contains("IMU")) {
84                  imuThread = new Thread(new ReadSerialData(data, comPortKey, 115200, comPortValue));
85                  imuThread.start();
86                  imuThread.setName(comPortValue);
87              }
88              if (comPortValue.contains("GPS")) {
89                  gpsThread = new Thread(new ReadSerialData(data, comPortKey, 115200, comPortValue));
90                  gpsThread.start();
91                  gpsThread.setName(comPortValue);
92              }
93              if (comPortValue.contains("EchoSounder")) {
94                  echoSounderThread = new Thread(new ReadSerialData(data, comPortKey, 4800, comPortValue));
95                  echoSounderThread.start();
96                  echoSounderThread.setName(comPortValue);
97              }
98              if (comPortValue.contains("ROVDummy")) {
99                  ROVDummyThread = new Thread(new ReadSerialData(data, comPortKey, 115200, comPortValue));
100                 ROVDummyThread.start();
101                 ROVDummyThread.setName(comPortValue);
102             }
103         }
104     }
105     //}
106 }
107 }
108
```

D:\Dokumenter\Skule\04 - NTNU\Bachelor\Github\TowedROV_GUI\src\ntnusubsea\gui\Data.java

```
/*
 * This code is for the bachelor thesis named "Towed-ROV".
 * The purpose is to build a ROV which will be towed behind a surface vessel
 * and act as a multi-sensor platform, were it shall be easy to place new
 * sensors. There will also be a video stream from the ROV.
 *
 * The system consists of two Raspberry Pis in the ROV that is connected to
 * several Arduino micro controllers. These micro controllers are connected to
 * feedback from the actuators, the echo sounder and extra optional sensors.
 * The external computer which is on the surface vessel is connected to a GPS,
 * echo sounder over USB, and the ROV over ethernet. It will present and
 * log data in addition to handle user commands for controlling the ROV.
 */
package ntnusubsea.gui;

import java.awt.image.BufferedImage;
import java.io.BufferedReader;
import java.io.File;
import java.io.FileReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.net.URI;
import java.nio.file.Path;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;
import java.util.Observable;
import java.util.concurrent.ConcurrentHashMap;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.imageio.ImageIO;
import javax.swing.JOptionPane;

/**
 * The data class is a storage box that let's the different threads change and
 * retrieve various data. The data class is a subclass of the java class
 * Observable, which makes it possible for observers to subscribe and update
 * their values whenever they change.
 *
 */
public final class Data extends Observable {

    public HashMap<String, String> comPortList = new HashMap<>();
    public ConcurrentHashMap<String, Boolean> completeAlarmListDh = new ConcurrentHashMap<>();

    //------------------------
    //Do not change the times, this is measured movement time without oil
    private static final long actuatorPSInitialMovementTime = 8000;
    private static final long actuatorSBInitialMovementTime = 8000;
    //------------------------
    private static final int actuatorTolerableSpeedLoss = 10; //Percent
    private long PSActuatorMaxToMinTime;
    private long SBActuatorMaxToMinTime;

    private int arduinoBaudRate = 115200;
    private byte[] dataFromArduino = new byte[11];
    private boolean dataFromArduinoAvailable = false;
    private byte requestCodeFromArduino;
    private boolean threadStatus = true;
    private boolean dataUpdated = false;
    private boolean controllerEnabled = false;

    //Dummy signals
    public double TestDepth = 0;
```

```
66
67  // Feedback from GPS
68  public int satellites = 0;
69  public float altitude = 0;
70  public double gpsAngle = 0;
71  public float speed = 0;
72  public float latitude = (float) 0;
73  public float longitude = (float) 0;
74  public float depth = (float) 0.01;
75  public float temperature = (float) 0.01;
76  public double voltage = 0.01;
77
78  // Feedback from IMU
79  public double roll = 0.00;
80  public double pitch = 0.00;
81  public float heading = 100;
82
83  // Feedback from the Camera RPi
84  private boolean leakStatus = false;
85  private double rovDepth = -0.00;
86  private double pressure = 0.00;
87  private double outsideTemp = 0.00;
88  private double insideTemp = 0.00;
89  private double humidity = 0.00;
90
91  // Feedback from ROV
92  private Double rovPing = 999.99;
93  private boolean rovReady = false;
94  private boolean i2cError;
95  private int fb_actuatorPSPos;
96  private int fb_actuatorSBPos;
97  private int fb_actuatorPScmd;
98  private int fb_actuatorSBcmd;
99
100  private int fb_actuatorPSMinPos;
101  private int fb_actuatorSBMinPos;
102  private int fb_actuatorPSMaxPos;
103  private int fb_actuatorSBMaxPos;
104  private double fb_tempElBoxFront;
105  private double fb_tempElBoxRear;
106
107  // Feedback from GUI
108  public boolean startLogging = true;
109  public ConcurrentHashMap<String, String> data = new ConcurrentHashMap<>();
110  public List<String> rovDepthDataList = new ArrayList<>();
111  public List<String> depthBeneathBoatDataList = new ArrayList<>();
112
113  private double timeBetweenBoatAndRov = 4.0;
114  private double depthBeneathRov = 0;
115  private double depthBeneathBoat = 0;
116  private double pitchAngle = 0;
117  private float wingAngle = 0;
118  private double rollAngle = 0;
119  private float channel1 = 0;
120  private float channel2 = 0;
121  private float channel3 = 0;
122  private float channel4 = 0;
123  private float channel5 = 0;
124  private float channel6 = 0;
125  private float channel7 = 0;
126  private float channel8 = 0;
127  private float channel9 = 0;
128  private byte actuatorStatus = 0;
129  private ArrayList<String> labels = new ArrayList();
130  private float[] channelValues = new float[9];
131  private String IP_Rov = "";
132  private String IP_Camera = "";
133  private BufferedImage videoImage;
```

```
134   private String Kp = "1";
135   private String Ki = "2";
136   private String Kd = "3";
137   private String offsetDepthBeneathROV = "0.00";
138   private String offsetROVdepth = "0.00";
139   private long timer = System.currentTimeMillis();
140   private boolean photoMode = false;
141   private double photoModeDelay = 1.00;
142   private double photoModeDelay_FB = 1.00;
143   private int imageNumber = 0;
144   private boolean imagesCleared = false;
145   private int cameraPitchValue = 0;
146   private boolean doRovCalibration = false;
147   private boolean emergencyMode = false;
148   private boolean streaming = false;
149   private boolean manualMode = false;
150
151   /**
152    * Creates an object of the class Data.
153    */
154   public Data() {
155      try {
156         BufferedReader br = null;
157         try {
158            br = new BufferedReader(new FileReader(new File("ROV Options.txt")));
159            //this.updateRovDepthDataList();
160            IP_Rov = br.readLine();
161            IP_Camera = br.readLine();
162            labels.add(0, br.readLine());
163            labels.add(1, br.readLine());
164            labels.add(2, br.readLine());
165            labels.add(3, br.readLine());
166            labels.add(4, br.readLine());
167            labels.add(5, br.readLine());
168            labels.add(6, br.readLine());
169            labels.add(7, br.readLine());
170            this.setKp(br.readLine());
171            this.setKi(br.readLine());
172            this.setKd(br.readLine());
173            this.setOffsetDepthBeneathROV(br.readLine());
174            this.setOffsetROVdepth(br.readLine());
175         } catch (Exception e) {
176            System.out.println("Error getting the ROV Options.txt file.");
177            IP_Rov = "0";
178            IP_Camera = "0";
179            labels.add(0, "1");
180            labels.add(1, "2");
181            labels.add(2, "3");
182            labels.add(3, "4");
183            labels.add(4, "5");
184            labels.add(5, "6");
185            labels.add(6, "7");
186            labels.add(7, "8");
187            this.setKp("0");
188            this.setKi("0");
189            this.setKd("0");
190            this.setOffsetDepthBeneathROV("0");
191            this.setOffsetROVdepth("0");
192         }
193
194         channelValues[0] = channel1;
195         channelValues[1] = channel2;
196         channelValues[2] = channel3;
197         channelValues[3] = channel4;
198         channelValues[4] = channel5;
199         channelValues[5] = channel6;
200         channelValues[6] = channel7;
201         channelValues[7] = channel8;
```

```
202             channelValues[8] = channel9;
203             videoImage = ImageIO.read(getClass().getResource("/ntnusubsea/gui/Images/TowedROV.jpg"));
204         } catch (IOException ex) {
205             Logger.getLogger(Data.class.getName()).log(Level.SEVERE, null, ex);
206         }
207     }
208 
209     /**
210      * Sets the default ROV IP
211      *
212      * @param ip The default ROV IP
213      */
214     public synchronized void setIP_Rov(String ip) {
215         this.IP_Rov = ip;
216         setChanged();
217         notifyObservers();
218     }
219 
220     /**
221      * Returns the default ROV IP
222      *
223      * @return The default ROV IP
224      */
225     public synchronized String getIP_Rov() {
226         return IP_Rov;
227     }
228 
229     /**
230      * Sets the default Camera IP
231      *
232      * @param ip The default Camera IP
233      */
234     public synchronized void setIP_Camera(String ip) {
235         this.IP_Camera = ip;
236         setChanged();
237         notifyObservers();
238     }
239 
240     /**
241      * Returns the default Camera IP
242      *
243      * @return The default Camera IP
244      */
245     public synchronized String getIP_Camera() {
246         return IP_Camera;
247     }
248 
249     /**
250      * Sets the Kp parameter of the PID
251      *
252      * @param value the Kp parameter of the PID
253      */
254     public void setKp(String value) {
255         this.Kp = value;
256     }
257 
258     /**
259      * Returns the Kp parameter of the PID
260      *
261      * @return the Kp parameter of the PID
262      */
263     public synchronized String getKp() {
264         return Kp;
265     }
266 
267     /**
268      * Sets the Ki parameter of the PID
269      *
```

```
270     * @param value the Ki parameter of the PID
271     */
272    public void setKi(String value) {
273        this.Ki = value;
274    }
275
276    /**
277     * Returns the Ki parameter of the PID
278     *
279     * @return the Ki parameter of the PID
280     */
281    public synchronized String getKi() {
282        return Ki;
283    }
284
285    /**
286     * Sets the Kd parameter of the PID
287     *
288     * @param value the Kd parameter of the PID
289     */
290    public void setKd(String value) {
291        this.Kd = value;
292    }
293
294    /**
295     * Returns the Kd parameter of the PID
296     *
297     * @return the Kd parameter of the PID
298     */
299    public synchronized String getKd() {
300        return Kd;
301    }
302
303    /**
304     * Returns the depth beneath the ROV offset
305     *
306     * @return the depth beneath the ROV offset
307     */
308    public String getOffsetDepthBeneathROV() {
309        return offsetDepthBeneathROV;
310    }
311
312    /**
313     * Sets the depth beneath the ROV offset
314     *
315     * @param offsetDepthBeneathROV the depth beneath the ROV offset
316     */
317    public void setOffsetDepthBeneathROV(String offsetDepthBeneathROV) {
318        this.offsetDepthBeneathROV = offsetDepthBeneathROV;
319    }
320
321    /**
322     * Returns the ROV depth offset
323     *
324     * @return the ROV depth offset
325     */
326    public String getOffsetROVdepth() {
327        return offsetROVdepth;
328    }
329
330    /**
331     * Sets the ROV depth offset
332     *
333     * @param offsetROVdepth the ROV offset
334     */
335    public void setOffsetROVdepth(String offsetROVdepth) {
336        this.offsetROVdepth = offsetROVdepth;
337    }
```

```
338
339     /**
340      * Sets the label of all the different I/O channels and notifies observers
341      *
342      * @param c1 Channel 1 label
343      * @param c2 Channel 2 label
344      * @param c3 Channel 3 label
345      * @param c4 Channel 4 label
346      * @param c5 Channel 5 label
347      * @param c6 Channel 6 label
348      * @param c7 Channel 7 label
349      * @param c8 Channel 8 label
350      */
351     public synchronized void setIOLabels(String c1, String c2, String c3, String c4, String c5, String c6, String c7, String c8) {
352         labels.set(0, c1);
353         labels.set(1, c2);
354         labels.set(2, c3);
355         labels.set(3, c4);
356         labels.set(4, c5);
357         labels.set(5, c6);
358         labels.set(6, c7);
359         labels.set(7, c8);
360         setChanged();
361         notifyObservers();
362     }
363
364     /**
365      * Returns a string containing the label of the channel. If the channel is
366      * an input, the string also contains its measure value. (Index 1-8)
367      *
368      * @param channel Index of channel to return
369      * @return String containing label and value
370      */
371     public synchronized String getChannel(int channel) {
372         if (channel > 0 && channel < 9) {
373             if (channel < 5) {
374                 String channelString = labels.get(channel - 1) + ": ";
375                 channelString += channelValues[channel - 1];
376                 return channelString;
377             } else {
378                 return labels.get(channel - 1);
379             }
380         } else {
381             return null;
382         }
383     }
384
385     /**
386      * Returns the value of the channel as a float. (Index 1-4)
387      *
388      * @param channel Index of channel
389      * @return Value of the channel as float
390      */
391     public synchronized float getChannelValue(int channel) {
392         if (channel < 0 && channel > 5) {
393             return channelValues[channel - 1];
394         } else {
395             return (float) 0.001;
396         }
397     }
398
399     /**
400      * Sets the value of one of the inputs and notifies observers (Index 1-4).
401      *
402      * @param value Value of the channel
403      * @param channel Index of the channel
404      */
405     public synchronized void setChannel(float value, int channel) {
```

```
406         if (channel < 0 && channel > 5) {
407             channelValues[channel - 1] = value;
408         }
409         setChanged();
410         notifyObservers();
411     }
412 
413     /**
414      * Updates the current pitch angle of the ROV and notifies observers
415      *
416      * @param angle Current pitch angle of the ROV
417      */
418     public synchronized void setPitchAngle(double angle) {
419         pitchAngle = angle;
420         setChanged();
421         notifyObservers();
422     }
423 
424     /**
425      * Retrieves the current pitch angle
426      *
427      * @return Current pitch angle of the ROV
428      */
429     public synchronized double getPitchAngle() {
430         return pitchAngle;
431     }
432 
433     /**
434      * Updates the current roll angle of the ROV
435      *
436      * @param angle Current roll angle of the ROV
437      */
438     public synchronized void setRollAngle(double angle) {
439         rollAngle = angle;
440         setChanged();
441         notifyObservers();
442     }
443 
444     /**
445      * Retrieves the current roll angle
446      *
447      * @return Current roll angle of the ROV
448      */
449     public synchronized double getRollAngle() {
450         return rollAngle;
451     }
452 
453     /**
454      * Updates the current wing angle of the ROV and notifies observers
455      *
456      * @param angle Current wing angle of the ROV
457      */
458     public synchronized void setWingAngle(float angle) {
459         wingAngle = angle;
460         setChanged();
461         notifyObservers();
462     }
463 
464     /**
465      * Retrieves the current pitch angle
466      *
467      * @return Current wing angle of the ROV
468      */
469     public synchronized float getWingAngle() {
470         return wingAngle;
471     }
472 
473     /**
```

```
474     * Updates the current heading of the ROV and notifies observers
475     *
476     * @param heading Current heading of the ROV
477     */
478    public synchronized void setHeading(float heading) {
479        this.heading = heading;
480        setChanged();
481        notifyObservers();
482    }
483
484    /**
485     * Retrieves the current heading
486     *
487     * @return Current heading of the ROV
488     */
489    public synchronized float getHeading() {
490        return heading;
491    }
492
493    /**
494     * Updates the current latitude of the ROV and notifies observers
495     *
496     * @param latitude Current latitude of the ROV
497     */
498    public synchronized void setLatitude(float latitude) {
499        this.latitude = latitude;
500        setChanged();
501        notifyObservers();
502    }
503
504    /**
505     * Retrieves the current latitude
506     *
507     * @return Current latitude of the ROV
508     */
509    public synchronized float getLatitude() {
510        return latitude;
511    }
512
513    /**
514     * Updates the current longitude of the ROV and notifies observers
515     *
516     * @param longitude Current longitude of the ROV
517     */
518    public synchronized void setLongitude(float longitude) {
519        this.longitude = longitude;
520        setChanged();
521        notifyObservers();
522    }
523
524    /**
525     * Retrieves the current longitude
526     *
527     * @return Current longitude of the ROV
528     */
529    public synchronized float getLongitude() {
530        return longitude;
531    }
532
533    /**
534     * Updates the current depth of the ROV and notifies observers
535     *
536     * @param depth Current depth of the ROV
537     */
538    public synchronized void setDepth(float depth) {
539        this.depth = depth;
540        setChanged();
541        notifyObservers();
```

```
542     }
543
544     /**
545      * Retrieves the current depth
546      *
547      * @return Current depth of the ROV
548      */
549     public synchronized float getDepth() {
550         return depth;
551     }
552
553     /**
554      * Returns the time between the boat and the ROV
555      *
556      * @return the time between the boat and the ROV
557      */
558     public double getTimeBetweenBoatAndRov() {
559         return timeBetweenBoatAndRov;
560     }
561
562     /**
563      * Sets the time between the boat and the ROV
564      *
565      * @param timeBetweenBoatAndRov the time between the boat and the ROV
566      */
567     public void setTimeBetweenBoatAndRov(double timeBetweenBoatAndRov) {
568         this.timeBetweenBoatAndRov = timeBetweenBoatAndRov;
569     }
570
571     /**
572      * Updates the current depth beneath the ROV and notifies observers
573      *
574      * @param depth Depth beneath the ROV
575      */
576     public synchronized void setDepthBeneathRov(double depth) {
577         depthBeneathRov = depth;
578         setChanged();
579         notifyObservers();
580     }
581
582     /**
583      * Returns the depth beneath the ROV
584      *
585      * @return Depth beneath the ROV
586      */
587     public synchronized double getDepthBeneathRov() {
588         return depthBeneathRov;
589     }
590
591     /**
592      * Updates the current depth beneath the vessel and notifies observers
593      *
594      * @param depth Depth beneath the vessel
595      */
596     public synchronized void setDepthBeneathBoat(double depth) {
597         depthBeneathBoat = depth;
598         setChanged();
599         notifyObservers();
600     }
601
602     /**
603      * Returns the depth beneath the vessel
604      *
605      * @return Depth beneath the vessel
606      */
607     public synchronized double getDepthBeneathBoat() {
608         return depthBeneathBoat;
609     }
```

```
610
611     /**
612      * Updates the image of the video stream and notifies observers
613      *
614      * @param image New image in the video stream
615      */
616     public synchronized void setVideoImage(BufferedImage image) {
617         videoImage = null;
618         videoImage = image;
619         setChanged();
620         notifyObservers();
621     }
622
623     /**
624      * Updates the status of the actuators. 1 if they are currently running and
625      * 0 if they are currently idle.
626      *
627      * @param status Current status of the actuators
628      */
629     public synchronized void setActuatorStatus(byte status) {
630         this.actuatorStatus = status;
631         setChanged();
632         notifyObservers();
633     }
634
635     /**
636      * Returns the status of the actuators. true if they are currently running
637      * and false if they are currently idle.
638      *
639      * @return Current status of the actuators
640      */
641     public synchronized boolean getActuatorStatus() {
642         if (actuatorStatus == 1) {
643             return true;
644         } else {
645             return false;
646         }
647     }
648
649     /**
650      * Updates the leak status in the ROV. 1 if a leak is detected, 0 if no leak
651      * is detected.
652      *
653      * @param leak Current leak status of the ROV
654      */
655     public synchronized void setLeakStatus(boolean leak) {
656         leakStatus = leak;
657         if (!leak) {
658             setEmergencyMode(false);
659         }
660         setChanged();
661         notifyObservers();
662     }
663
664     /**
665      * Returns the leak status of the ROV. Returns true if a leak is detected,
666      * false if no leak is detected
667      *
668      * @return Current leak status of the ROV
669      */
670     public synchronized boolean getLeakStatus() {
671         return leakStatus;
672     }
673
674     /**
675      * Updates the temperature of the water and notifies observers
676      *
677      * @param temp Temperature of the water
```

```
678     */
679    public synchronized void setTemperature(float temp) {
680        temperature = temp;
681        setChanged();
682        notifyObservers();
683    }
684
685    /**
686     * Returns the current temperature of the water
687     *
688     * @return Temperature of the water
689     */
690    public synchronized float getTemperature() {
691        return temperature;
692    }
693
694    /**
695     * Updates the pressure surrounding the ROV and notifies observers
696     *
697     * @param pres Pressure surrounding the ROV
698     */
699    public synchronized void setPressure(double pres) {
700        pressure = pres;
701        setChanged();
702        notifyObservers();
703    }
704
705    /**
706     * Returns the current pressure around the ROV
707     *
708     * @return Current pressure around the ROV
709     */
710    public synchronized double getPressure() {
711        return pressure;
712    }
713
714    /**
715     * Returns the temperature in the sea
716     *
717     * @return the temperature in the sea
718     */
719    public double getOutsideTemp() {
720        return outsideTemp;
721    }
722
723    /**
724     * Sets the temperature in the sea
725     *
726     * @param outsideTemp the temperature in the sea
727     */
728    public void setOutsideTemp(double outsideTemp) {
729        this.outsideTemp = outsideTemp;
730    }
731
732    /**
733     * Returns the temperature inside the camera housing
734     *
735     * @return the temperature inside the camera housing
736     */
737    public double getInsideTemp() {
738        return insideTemp;
739    }
740
741    /**
742     * Sets the temperature inside the camera housing
743     *
744     * @param insideTemp the temperature inside the camera housing
745     */
```

```
746   public void setInsideTemp(double insideTemp) {
747       this.insideTemp = insideTemp;
748   }
749
750   /**
751    * Returns the humidity in the camera housing
752    *
753    * @return the humidity in the camera housing
754    */
755   public double getHumidity() {
756       return humidity;
757   }
758
759   /**
760    * Sets the humidity in the camera housing
761    *
762    * @param humidity the humidity in the camera housing
763    */
764   public void setHumidity(double humidity) {
765       this.humidity = humidity;
766   }
767
768   /**
769    * Sets the current speed of the vessel and notifies observers
770    *
771    * @param speed Current speed of the vessel
772    */
773   public synchronized void setSpeed(float speed) {
774       this.speed = speed;
775       setChanged();
776       notifyObservers();
777   }
778
779   /**
780    * Returns the current speed of the vessel
781    *
782    * @return Current speed of the vessel
783    */
784   public synchronized float getSpeed() {
785       return speed;
786   }
787
788   /**
789    * Returns the current image in the video stream
790    *
791    * @return Current image in the video stream
792    */
793   public synchronized BufferedImage getVideoImage() {
794       return videoImage;
795   }
796
797   /**
798    * Returns the state of the photoMode variable
799    *
800    * @return the state of the photoMode variable, true or false
801    */
802   public boolean isPhotoMode() {
803       return photoMode;
804   }
805
806   /**
807    * Sets the state of the photoMode variable
808    *
809    * @param photoMode the state of the photoMode variable, true or false
810    */
811   public void setPhotoMode(boolean photoMode) {
812       this.photoMode = photoMode;
813   }
```

```
814
815     /**
816      * Returns the photo mode delay
817      *
818      * @return the photo mode delay
819      */
820     public double getPhotoModeDelay() {
821         return photoModeDelay;
822     }
823
824     /**
825      * Sets the photo mode delay and notifies observers
826      *
827      * @param photoModeDelay
828      */
829     public void setPhotoModeDelay(double photoModeDelay) {
830         this.photoModeDelay = photoModeDelay;
831         setChanged();
832         notifyObservers();
833     }
834
835     /**
836      * Returns the photo mode delay feedback
837      *
838      * @return the photo mode delay feedback
839      */
840     public double getPhotoModeDelay_FB() {
841         return photoModeDelay_FB;
842     }
843
844     /**
845      * Sets the photo mode delay feedback and notifies observers
846      *
847      * @param photoModeDelay_FB
848      */
849     public void setPhotoModeDelay_FB(double photoModeDelay_FB) {
850         this.photoModeDelay_FB = photoModeDelay_FB;
851         setChanged();
852         notifyObservers();
853     }
854
855     /**
856      * Returns the camera pitch value
857      *
858      * @return the camera pitch value
859      */
860     public int getCameraPitchValue() {
861         return cameraPitchValue;
862     }
863
864     /**
865      * Sets the camera pitch value
866      *
867      * @param cameraPitchValue
868      */
869     public void setCameraPitchValue(int cameraPitchValue) {
870         this.cameraPitchValue = cameraPitchValue;
871     }
872
873     /**
874      * Returns the image number value
875      *
876      * @return the image number value
877      */
878     public int getImageNumber() {
879         return imageNumber;
880     }
881
```

```
882  /**
883   * Sets the image number value and notifies observers
884   *
885   * @param imageNumber the image number value
886   */
887  public void setImageNumber(int imageNumber) {
888      this.imageNumber = imageNumber;
889      setChanged();
890      notifyObservers();
891  }
892
893  /**
894   * Returns the images cleared status
895   *
896   * @return the images cleared status
897   */
898  public boolean isImagesCleared() {
899      return imagesCleared;
900  }
901
902  /**
903   * Sets the images cleared status
904   *
905   * @param imagesCleared the images cleared status
906   */
907  public void setImagesCleared(boolean imagesCleared) {
908      this.imagesCleared = imagesCleared;
909  }
910
911  /**
912   * Increases the image number by one and notifies observers
913   */
914  public void increaseImageNumberByOne() {
915      this.imageNumber++;
916      setChanged();
917      notifyObservers();
918  }
919
920  // CODE BELOW ADDED FROM THE BASESTATION PROJECT
921  /**
922   * Checks status of thread
923   *
924   * @return thread status
925   */
926  public boolean shouldThreadRun() {
927      return threadStatus;
928  }
929
930  /**
931   * Sets the thread status
932   *
933   * @param threadStatus
934   */
935  public void setThreadStatus(boolean threadStatus) {
936      this.threadStatus = threadStatus;
937  }
938
939  /**
940   * Returns the data from the arduino
941   *
942   * @return the data from the arduino
943   */
944  public byte[] getDataFromArduino() {
945      return dataFromArduino;
946  }
947
948  /**
949   * Returns true if there is new data available, false if not
```

```
950     *
951     * @return true if new data available, false if not
952     */
953    public synchronized boolean isDataFromArduinoAvailable() {
954        return this.dataFromArduinoAvailable;
955    }
956
957    /**
958     * Sets the emergency mode status
959     *
960     * @param status the emergency mode status
961     */
962    public void setEmergencyMode(boolean status) {
963        this.emergencyMode = status;
964 //         setChanged();
965 //         notifyObservers();
966    }
967
968    /**
969     * Returns the emergency mode status
970     *
971     * @return the emergency mode status
972     */
973    public boolean isEmergencyMode() {
974        return this.emergencyMode;
975    }
976
977    /**
978     * Returns the streaming status
979     *
980     * @return the streaming status
981     */
982    public boolean isStreaming() {
983        return streaming;
984    }
985
986    /**
987     * Sets the streaming status
988     *
989     * @param streaming the streaming status
990     */
991    public void setStreaming(boolean streaming) {
992        this.streaming = streaming;
993    }
994
995    /**
996     * Returns the manual mode status
997     *
998     * @return the manual mode status
999     */
1000    public boolean isManualMode() {
1001        return manualMode;
1002    }
1003
1004    /**
1005     * Sets the manual mode status
1006     *
1007     * @param manualMode the manual mode status
1008     */
1009    public void setManualMode(boolean manualMode) {
1010        this.manualMode = manualMode;
1011    }
1012
1013    /**
1014     * Returns the data updated status
1015     *
1016     * @return the data updated status
1017     */
```

```
1018  public synchronized boolean isDataUpdated() {
1019      return dataUpdated;
1020  }
1021
1022  /**
1023   * Sets the data updated status
1024   *
1025   * @param dataUpdated the data updated status
1026   */
1027  public synchronized void setDataUpdated(boolean dataUpdated) {
1028      this.dataUpdated = dataUpdated;
1029  }
1030
1031  /**
1032   * Returns the controller enabled status
1033   *
1034   * @return the controller enabled status
1035   */
1036  public boolean isControllerEnabled() {
1037      return controllerEnabled;
1038  }
1039
1040  /**
1041   * Sets the controller enabled status
1042   *
1043   * @param controllerEnabled the controller enabled status
1044   */
1045  public void setControllerEnabled(boolean controllerEnabled) {
1046      this.controllerEnabled = controllerEnabled;
1047  }
1048
1049  /**
1050   * Returns the number of satellites
1051   *
1052   * @return the number of satellites
1053   */
1054  public synchronized int getSatellites() {
1055      return satellites;
1056  }
1057
1058  /**
1059   * Sets the number of satellites
1060   *
1061   * @param satellites the number of satellites
1062   */
1063  public synchronized void setSatellites(int satellites) {
1064      this.satellites = satellites;
1065      setChanged();
1066      notifyObservers();
1067  }
1068
1069  /**
1070   * Returns the altitude
1071   *
1072   * @returnthe altitude
1073   */
1074  public synchronized float getAltitude() {
1075      return altitude;
1076  }
1077
1078  /**
1079   * Sets the altitude
1080   *
1081   * @param altitude the altitude
1082   */
1083  public synchronized void setAltitude(float altitude) {
1084      this.altitude = altitude;
1085      setChanged();
```

```
1086        notifyObservers();
1087    }
1088
1089    /**
1090     * Returns the GPS angle
1091     *
1092     * @return the GPS angle
1093     */
1094    public synchronized double getGPSAngle() {
1095        return gpsAngle;
1096    }
1097
1098    /**
1099     * Sets the GPS angle
1100     *
1101     * @param angle the GPS angle
1102     */
1103    public synchronized void setGPSAngle(double angle) {
1104        this.gpsAngle = angle;
1105        setChanged();
1106        notifyObservers();
1107    }
1108
1109    /**
1110     * Returns the roll if the ROV
1111     *
1112     * @return the roll if the ROV
1113     */
1114    public synchronized double getRoll() {
1115        return roll;
1116    }
1117
1118    /**
1119     * Sets the roll if the ROV
1120     *
1121     * @param roll the roll if the ROV
1122     */
1123    public synchronized void setRoll(double roll) {
1124        this.roll = roll;
1125        setChanged();
1126        notifyObservers();
1127    }
1128
1129    /**
1130     * Returns the pitch of the ROV
1131     *
1132     * @return the pitch of the ROV
1133     */
1134    public synchronized double getPitch() {
1135        return pitch;
1136    }
1137
1138    /**
1139     * Sets the pitch of the ROV
1140     *
1141     * @param pitch the pitch of the ROV
1142     */
1143    public synchronized void setPitch(double pitch) {
1144        this.pitch = pitch;
1145        setChanged();
1146        notifyObservers();
1147    }
1148
1149    /**
1150     * Returns the voltage supply value
1151     *
1152     * @return the voltage supply value
1153     */
```

```
1154    public synchronized double getVoltage() {
1155        return voltage;
1156    }
1157
1158    /**
1159     * Sets the voltage supply value
1160     *
1161     * @param voltage the voltage supply value
1162     */
1163    public synchronized void setVoltage(double voltage) {
1164        this.voltage = voltage;
1165        setChanged();
1166        notifyObservers();
1167    }
1168
1169    /**
1170     * Returns the start logging status
1171     *
1172     * @return the start logging status
1173     */
1174    public boolean getStartLogging() {
1175        return startLogging;
1176    }
1177
1178    /**
1179     * Sets the start logging status
1180     *
1181     * @param startLogging the start logging status
1182     */
1183    public void setStartLogging(boolean startLogging) {
1184        this.startLogging = startLogging;
1185        setChanged();
1186        notifyObservers();
1187    }
1188
1189    /**
1190     * Returns the ROV ping
1191     *
1192     * @return the ROV ping
1193     */
1194    public Double getRovPing() {
1195        return rovPing;
1196    }
1197
1198    /**
1199     * Sets the ROV ping
1200     *
1201     * @param rovPing the ROV ping
1202     */
1203    public void setRovPing(Double rovPing) {
1204        this.rovPing = rovPing;
1205        setChanged();
1206        notifyObservers();
1207    }
1208
1209    /**
1210     * Returns the ROV ready status
1211     *
1212     * @return the ROV ready status
1213     */
1214    public boolean isRovReady() {
1215        return rovReady;
1216    }
1217
1218    /**
1219     * Sets the ROV ready status
1220     *
1221     * @param rovReady the ROV ready status
```

```
1222     */
1223    public void setRovReady(boolean rovReady) {
1224        this.rovReady = rovReady;
1225    }
1226
1227    /**
1228     * Returns the i2c error status
1229     *
1230     * @return the i2c error status
1231     */
1232    public boolean isI2cError() {
1233        return i2cError;
1234    }
1235
1236    /**
1237     * Sets the i2c error status
1238     *
1239     * @param i2cError the i2c error status
1240     */
1241    public void setI2cError(boolean i2cError) {
1242        this.i2cError = i2cError;
1243    }
1244
1245    /**
1246     * Returns the ROV depth
1247     *
1248     * @return the ROV depth
1249     */
1250    public Double getRovDepth() {
1251        return rovDepth;
1252    }
1253
1254    /**
1255     * Sets the ROV depth
1256     *
1257     * @param rovDepth the ROV depth
1258     */
1259    public void setRovDepth(Double rovDepth) {
1260        this.rovDepth = rovDepth;
1261    }
1262
1263    /**
1264     * Returns the PS actuator position
1265     *
1266     * @return the PS actuator position
1267     */
1268    public int getFb_actuatorPSPos() {
1269        return fb_actuatorPSPos;
1270    }
1271
1272    /**
1273     * Sets the PS actuator position
1274     *
1275     * @param fb_actuatorPSPos the PS actuator position
1276     */
1277    public void setFb_actuatorPSPos(int fb_actuatorPSPos) {
1278        this.fb_actuatorPSPos = fb_actuatorPSPos;
1279    }
1280
1281    /**
1282     * Returns the SB actuator position
1283     *
1284     * @return the SB actuator position
1285     */
1286    public int getFb_actuatorSBPos() {
1287        return fb_actuatorSBPos;
1288    }
1289
```

```
1290    /**
1291     * Sets the SB actuator position
1292     *
1293     * @param fb_actuatorSBPos the SB actuator position
1294     */
1295    public void setFb_actuatorSBPos(int fb_actuatorSBPos) {
1296        this.fb_actuatorSBPos = fb_actuatorSBPos;
1297    }
1298
1299    /**
1300     * Returns the PS minimum actuator position
1301     *
1302     * @return the PS minimum actuator position
1303     */
1304    public int getFb_actuatorPSMinPos() {
1305        return fb_actuatorPSMinPos;
1306    }
1307
1308    /**
1309     * Sets the PS minimum actuator position
1310     *
1311     * @param fb_actuatorPSMinPos the PS minimum actuator position
1312     */
1313    public void setFb_actuatorPSMinPos(int fb_actuatorPSMinPos) {
1314        this.fb_actuatorPSMinPos = fb_actuatorPSMinPos;
1315    }
1316
1317    /**
1318     * Returns the SB minimum actuator position
1319     *
1320     * @return the SB minimum actuator position
1321     */
1322    public int getFb_actuatorSBMinPos() {
1323        return fb_actuatorSBMinPos;
1324    }
1325
1326    /**
1327     * Sets the SB minimum actuator position
1328     *
1329     * @param fb_actuatorSBMinPos the SB minimum actuator position
1330     */
1331    public void setFb_actuatorSBMinPos(int fb_actuatorSBMinPos) {
1332        this.fb_actuatorSBMinPos = fb_actuatorSBMinPos;
1333    }
1334
1335    /**
1336     * Returns the PS maximum actuator position
1337     *
1338     * @return the PS maximum actuator position
1339     */
1340    public int getFb_actuatorPSMaxPos() {
1341        return fb_actuatorPSMaxPos;
1342    }
1343
1344    /**
1345     * Sets the PS maximum actuator position
1346     *
1347     * @param fb_actuatorPSMaxPos the PS maximum actuator position
1348     */
1349    public void setFb_actuatorPSMaxPos(int fb_actuatorPSMaxPos) {
1350        this.fb_actuatorPSMaxPos = fb_actuatorPSMaxPos;
1351    }
1352
1353    /**
1354     * Returns the SB maximum actuator position
1355     *
1356     * @return the SB maximum actuator position
1357     */
```

```
1358    public int getFb_actuatorSBMaxPos() {
1359        return fb_actuatorSBMaxPos;
1360    }
1361
1362    /**
1363     * Sets the SB maximum actuator position
1364     *
1365     * @param fb_actuatorSBMaxPos the PS maximum actuator position
1366     */
1367    public void setFb_actuatorSBMaxPos(int fb_actuatorSBMaxPos) {
1368        this.fb_actuatorSBMaxPos = fb_actuatorSBMaxPos;
1369    }
1370
1371    /**
1372     * Returns the temperature in the front of the electronics box
1373     *
1374     * @return the temperature in the front of the electronics box
1375     */
1376    public double getFb_tempElBoxFront() {
1377        return fb_tempElBoxFront;
1378    }
1379
1380    /**
1381     * Sets the temperature in the front of the electronics box
1382     *
1383     * @param fb_tempElBoxFront the temperature in the front of the electronics
1384     * box
1385     */
1386    public void setFb_tempElBoxFront(double fb_tempElBoxFront) {
1387        this.fb_tempElBoxFront = fb_tempElBoxFront;
1388    }
1389
1390    /**
1391     * Returns the temperature in the rear of the electronics box
1392     *
1393     * @return the temperature in the rear of the electronics box
1394     */
1395    public double getFb_tempElBoxRear() {
1396        return fb_tempElBoxRear;
1397    }
1398
1399    /**
1400     * Sets the temperature in the rear of the electronics box
1401     *
1402     * @param fb_tempElBoxRear the temperature in the rear of the electronics
1403     * box
1404     */
1405    public void setFb_tempElBoxRear(double fb_tempElBoxRear) {
1406        this.fb_tempElBoxRear = fb_tempElBoxRear;
1407    }
1408
1409    /**
1410     * Returns PS the actuator max-to-min time
1411     *
1412     * @return the PS actuator max-to-min time
1413     */
1414    public long getPSActuatorMaxToMinTime() {
1415        return PSActuatorMaxToMinTime;
1416    }
1417
1418    /**
1419     * Sets the PS actuator max-to-min time
1420     *
1421     * @param PSActuatorMaxToMinTime
1422     */
1423    public void setPSActuatorMaxToMinTime(long PSActuatorMaxToMinTime) {
1424        this.PSActuatorMaxToMinTime = PSActuatorMaxToMinTime;
1425    }
```

```
1426
1427    /**
1428     * Returns the SB actuator max-to-min time
1429     *
1430     * @return the SB actuator max-to-min time
1431     */
1432    public long getSBActuatorMaxToMinTime() {
1433        return SBActuatorMaxToMinTime;
1434    }
1435
1436    /**
1437     * Sets the SB actuator max-to-min time
1438     *
1439     * @param SBActuatorMaxToMinTime
1440     */
1441    public void setSBActuatorMaxToMinTime(long SBActuatorMaxToMinTime) {
1442        this.SBActuatorMaxToMinTime = SBActuatorMaxToMinTime;
1443    }
1444
1445    /**
1446     * Updates the ROV depth data list
1447     *
1448     * @param time the time variable
1449     * @param value the value at that time
1450     */
1451    public void updateRovDepthDataList(String time, String value) {
1452        if (rovDepthDataList.size() >= 260) {
1453            rovDepthDataList.remove(0);
1454        }
1455        this.rovDepthDataList.add(time + ":" + value);
1456    }
1457
1458    /**
1459     * Updates the depth beneath the boat data list
1460     *
1461     * @param time the time variable
1462     * @param value the value at that time
1463     */
1464    public void updateDepthBeneathBoatDataList(String time, String value) {
1465        if (depthBeneathBoatDataList.size() >= 300) {
1466            depthBeneathBoatDataList.remove(0);
1467        }
1468        this.depthBeneathBoatDataList.add(time + ":" + value);
1469    }
1470
1471    /**
1472     * Returns the PS actuator command
1473     *
1474     * @return the PS actuator command
1475     */
1476    public int getFb_actuatorPScmd() {
1477        return fb_actuatorPScmd;
1478    }
1479
1480    /**
1481     * Sets the PS actuator command
1482     *
1483     * @param fb_actuatorPScmd the PS actuator command
1484     */
1485    public void setFb_actuatorPScmd(int fb_actuatorPScmd) {
1486        this.fb_actuatorPScmd = fb_actuatorPScmd;
1487    }
1488
1489    /**
1490     * Returns the SB actuator command
1491     *
1492     * @return the SB actuator command
1493     */
```

```
1494    public int getFb_actuatorSBcmd() {
1495        return fb_actuatorSBcmd;
1496    }
1497
1498    /**
1499     * Sets the SB actuator command
1500     *
1501     * @param fb_actuatorSBcmd the SB actuator command
1502     */
1503    public void setFb_actuatorSBcmd(int fb_actuatorSBcmd) {
1504        this.fb_actuatorSBcmd = fb_actuatorSBcmd;
1505    }
1506
1507    /**
1508     * Returns the test depth
1509     *
1510     * @return the test depth
1511     */
1512    public double getTestDepth() {
1513        return TestDepth;
1514    }
1515
1516    /**
1517     * Sets the test depth
1518     *
1519     * @param TestDepth the test depth
1520     */
1521    public void setTestDepth(double TestDepth) {
1522        this.TestDepth = TestDepth;
1523        this.setRovDepth(TestDepth);
1524    }
1525
1526 }
1527
```

D:\Dokumenter\Skule\04 - NTNU\Bachelor\Github\TowedROV_GUI\src\ntnusubsea\gui\DataUpdater.java

```
1 /*
2  * This code is for the bachelor thesis named "Towed-ROV".
3  * The purpose is to build a ROV which will be towed behind a surface vessel
4  * and act as a multi-sensor platform, were it shall be easy to place new
5  * sensors. There will also be a video stream from the ROV.
6  *
7  * The system consists of two Raspberry Pis in the ROV that is connected to
8  * several Arduino micro controllers. These micro controllers are connected to
9  * feedback from the actuators, the echo sounder and extra optional sensors.
10  * The external computer which is on the surface vessel is connected to a GPS,
11  * echo sounder over USB, and the ROV over ethernet. It will present and
12  * log data in addition to handle user commands for controlling the ROV.
13  */
14 package ntnusubsea.gui;
15
16 import java.io.IOException;
17
18 /**
19  * This class updates all of the data from the RPis in the ROV by using their
20  * respective TCP clients. It also sends the value from the echo sounder onboard
21  * the boat to the ROV.
22  */
23 public class DataUpdater implements Runnable {
24
25     private TCPClient client_Rov;
26     private TCPClient client_Camera;
27     private Data data;
28
29     /**
30      * Creates an instance of the DataUpdater class.
31      *
32      * @param client_Rov The ROV TCP client
33      * @param client_Camera The camera TCP client
34      * @param data The shared resource Data object
35      */
36     public DataUpdater(TCPClient client_Rov, TCPClient client_Camera, Data data) {
37         this.client_Rov = client_Rov;
38         this.client_Camera = client_Camera;
39         this.data = data;
40     }
41
42     /**
43      * Runs the DataUpdater thread and sends the "get data" commands to the TCP
44      * servers on the main RPi and the camera RPi. It also sends the echo
45      * sounder depth value to the main RPi.
46      */
47     @Override
48     public void run() {
49         if (client_Rov.isConnected()) {
50             try {
51                 client_Rov.sendCommand("fb_allData");
52                 if (!data.comPortList.containsKey("ROVDummy")
53                         && !data.comPortList.containsValue("ROVDummy")) {
54                     client_Rov.sendCommand("cmd_rovDepth:" + data.getRovDepth());
55                 } else {
```

```
56                client_Rov.sendCommand("cmd_rovDepth:" + data.getTestDepth());
57             }
58          } catch (IOException ex) {
59             System.out.println("Error while getting data from remote: " + ex.getMessage());
60          }
61
62       }
63       if (client_Camera.isConnected()) {
64          try {
65             client_Camera.sendCommand("getData");
66          } catch (IOException ex) {
67             System.out.println("Error while getting data from remote: " + ex.getMessage());
68          }
69
70       }
71
72    }
73
74 }
75
```

D:\Dokumenter\Skule\04 - NTNU\Bachelor\Github\TowedROV_GUI\src\ntnusubsea\gui\EchoSounderFrame.java

```
1  /*
2  * This code is for the bachelor thesis named "Towed-ROV".
3  * The purpose is to build a ROV which will be towed behind a surface vessel
4  * and act as a multi-sensor platform, were it shall be easy to place new
5  * sensors. There will also be a video stream from the ROV.
6  *
7  * The system consists of two Raspberry Pis in the ROV that is connected to
8  * several Arduino micro controllers. These micro controllers are connected to
9  * feedback from the actuators, the echo sounder and extra optional sensors.
10 * The external computer which is on the surface vessel is connected to a GPS,
11 * echo sounder over USB, and the ROV over ethernet. It will present and
12 * log data in addition to handle user commands for controlling the ROV.
13 */
14 package ntnusubsea.gui;
15
16 import java.awt.BorderLayout;
17 import java.awt.Color;
18 import java.io.BufferedReader;
19 import java.io.File;
20 import java.io.FileInputStream;
21 import java.io.InputStream;
22 import java.io.InputStreamReader;
23 import java.text.DecimalFormat;
24 import java.util.Iterator;
25 import java.util.Observable;
26 import java.util.Observer;
27 import java.util.logging.Level;
28 import java.util.logging.Logger;
29 import javax.swing.JFrame;
30 import javax.swing.JOptionPane;
31 import javax.swing.JPanel;
32 import org.jfree.chart.ChartFactory;
33 import org.jfree.chart.ChartPanel;
34 import org.jfree.chart.JFreeChart;
35 import org.jfree.chart.plot.XYPlot;
36 import org.jfree.data.xy.XYDataset;
37 import org.jfree.data.xy.XYSeries;
38 import org.jfree.data.xy.XYSeriesCollection;
39
40 /**
41  * Frame to display a graph panel
42  */
43 public class EchoSounderFrame extends javax.swing.JFrame implements Runnable, Observer {
44
45     private Data data;
46     private XYPlot plot;
47
48     /**
49      * Creates new form SonarFrame
50      *
51      * @param data Data containing shared variables
52      */
53     public EchoSounderFrame(Data data) {
54         super("XY Line Chart Example with JFreechart");
55         initComponents();
56         this.data = data;
57         JPanel chartPanel = createChartPanel();
58
59         //setSize(640, 480);
60         this.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
61         this.setLocationRelativeTo(null);
62
63         chartPanel.setSize(jPanel1.getWidth(), jPanel1.getHeight());
64         chartPanel.setVisible(true);
65         jPanel1.add(chartPanel, BorderLayout.CENTER);
66         this.add(jPanel1);
67         this.pack();
68     }
69
70     /**
71      * Creates the chart panel.
72      *
73      * @return the chart panel
74      */
75     private JPanel createChartPanel() {
76         String chartTitle = "ROV Depth Chart";
77         String xAxisLabel = "Time (s)";
78         String yAxisLabel = "Depth";
79         XYDataset dataset = createDatasetLive();
80         JFreeChart chart = ChartFactory.createXYLineChart(chartTitle,
81                 xAxisLabel, yAxisLabel, dataset);
82         plot = chart.getXYPlot();
83         plot.setBackgroundPaint(Color.DARK_GRAY);
84         plot.setRangeGridlinesVisible(true);
```

```
85      plot.setRangeGridlinePaint(Color.BLACK);
86      plot.setDomainGridlinesVisible(true);
87      plot.setDomainGridlinePaint(Color.BLACK);
88
89      return new ChartPanel(chart);
90  }
91
92  /**
93   * Creates the dataset from a dataset file.
94   *
95   * @return the dataset from a dataset file
96   */
97  private XYDataset createDatasetFromFile() {
98      XYSeriesCollection dataset = new XYSeriesCollection();
99      XYSeries series1 = new XYSeries("Seafloor");
100     XYSeries series2 = new XYSeries("ROV Depth");
101     XYSeries series3 = new XYSeries("Surface");
102
103     try {
104         InputStream depthtoseafloorList = new FileInputStream(new File("C:\\depthtoseafloor.txt"));
105         InputStream rovdepthList = new FileInputStream(new File("C:\\rovdepth.txt"));
106         BufferedReader reader1 = new BufferedReader(new InputStreamReader(depthtoseafloorList));
107         BufferedReader reader2 = new BufferedReader(new InputStreamReader(rovdepthList));
108
109         String line = "";
110         int i = 0;
111         double t = 0.0;
112         while ((line = reader1.readLine()) != null) {
113             series1.add(t, Double.parseDouble(line));
114             series3.add(t, 0.01);
115             i++;
116             t = t + 0.1;
117         }
118         i = 0;
119         t = 0.0;
120         while ((line = reader2.readLine()) != null) {
121             series2.add(t, Double.parseDouble(line) * -1);
122             i++;
123             t = t + 0.1;
124         }
125
126     } catch (Exception e) {
127         System.out.println("error");
128     }
129
130     dataset.addSeries(series1);
131     dataset.addSeries(series2);
132     dataset.addSeries(series3);
133
134     return dataset;
135 }
136
137 /**
138  * Creates the dataset from the depth data.
139  *
140  * @return the dataset from the depth data.
141  */
142 private XYDataset createDatasetLive() {
143     XYSeriesCollection dataset = new XYSeriesCollection();
144     XYSeries series1 = new XYSeries("ROV Depth");
145     XYSeries series2 = new XYSeries("Seafloor");
146     XYSeries series3 = new XYSeries("Surface");
147
148     Iterator it = data.rovDepthDataList.iterator();
149     while (it.hasNext()) {
150         String s = it.next().toString();
151         String[] data = s.split(":");
152         series1.add(Double.parseDouble(data[0]), Double.parseDouble(data[1]));
153     }
154
155     Iterator it2 = data.depthBeneathBoatDataList.iterator();
156     while (it2.hasNext()) {
157         String s = it2.next().toString();
158         String[] data = s.split(":");
159         series2.add(Double.parseDouble(data[0]), Double.parseDouble(data[1]));
160         series3.add(Double.parseDouble(data[0]), 0.01);
161     }
162
163     dataset.addSeries(series1);
164     dataset.addSeries(series2);
165     dataset.addSeries(series3);
166
167     return dataset;
168 }
169
170 /**
171  * This method is called from within the constructor to initialize the form.
```

```
172     * WARNING: Do NOT modify this code. The content of this method is always
173     * regenerated by the Form Editor.
174     */
175    @SuppressWarnings("unchecked")
176    // <editor-fold defaultstate="collapsed" desc="Generated Code">
177    private void initComponents()
178    {
179
180        jPanel1 = new javax.swing.JPanel();
181        jMenuBar1 = new javax.swing.JMenuBar();
182        jMenu1 = new javax.swing.JMenu();
183        jMenuItem1 = new javax.swing.JMenuItem();
184        jMenu2 = new javax.swing.JMenu();
185        jMenuItem2 = new javax.swing.JMenuItem();
186
187        setDefaultCloseOperation(javax.swing.WindowConstants.DISPOSE_ON_CLOSE);
188        setTitle("Echo sounder");
189        setBackground(new java.awt.Color(39, 44, 50));
190        setMinimumSize(new java.awt.Dimension(1200, 600));
191        setPreferredSize(new java.awt.Dimension(1200, 600));
192        setSize(new java.awt.Dimension(1200, 600));
193
194        jPanel1.setBackground(new java.awt.Color(39, 44, 50));
195        jPanel1.setForeground(new java.awt.Color(39, 44, 50));
196        jPanel1.setMinimumSize(new java.awt.Dimension(1200, 600));
197        jPanel1.setPreferredSize(new java.awt.Dimension(1200, 600));
198        jPanel1.setLayout(new java.awt.BorderLayout());
199        getContentPane().add(jPanel1, java.awt.BorderLayout.CENTER);
200
201        jMenu1.setText("File");
202
203        jMenuItem1.setText("Exit");
204        jMenuItem1.addActionListener(new java.awt.event.ActionListener()
205        {
206            public void actionPerformed(java.awt.event.ActionEvent evt)
207            {
208                jMenuItem1ActionPerformed(evt);
209            }
210        });
211        jMenu1.add(jMenuItem1);
212
213        jMenuBar1.add(jMenu1);
214
215        jMenu2.setText("Tools");
216
217        jMenuItem2.setText("Calibrate");
218        jMenuItem2.addActionListener(new java.awt.event.ActionListener()
219        {
220            public void actionPerformed(java.awt.event.ActionEvent evt)
221            {
222                jMenuItem2ActionPerformed(evt);
223            }
224        });
225        jMenu2.add(jMenuItem2);
226
227        jMenuBar1.add(jMenu2);
228
229        setJMenuBar(jMenuBar1);
230
231        pack();
232    }// </editor-fold>
233
234    private void jMenuItem1ActionPerformed(java.awt.event.ActionEvent evt) {
235        this.dispose();
236    }
237
238    private void jMenuItem2ActionPerformed(java.awt.event.ActionEvent evt) {
239        String cableLength = (String) JOptionPane.showInputDialog(this, "Enten current cable length (Meters)", "Calibration", JOptionPane.PLAIN_MESSAGE, null
240        try {
241            System.out.println(Float.valueOf(cableLength));
242        } catch (NumberFormatException | NullPointerException ex) {
243            System.out.println("Invalid or no input");
244        }
245    }
246
247    /**
248     * Runs the EchoSounderFrame thread.
249     */
250    @Override
251    public void run() {
252
253        DecimalFormat df = new DecimalFormat();
254        df.setMaximumFractionDigits(2);
255        double time = 0.0;
256        double time2 = time + data.getTimeBetweenBoatAndRov();
257        String rovDepthValue = "0.0";
258        String depthBeneathBoatValue = "0.0";
```

```
259         double counter = 0;
260         double counter2 = -25.0;
261         double amount = -1.0;
262         double amount2 = -0.1;
263 
264         while (true) {
265 //          for (int i = 1; i < 10; i++)
266 //          {
267 //
268             try {
269 
270                 data.updateRovDepthDataList(String.valueOf(df.format(time)), rovDepthValue);
271                 data.updateDepthBeneathBoatDataList(String.valueOf(df.format(time2)), depthBeneathBoatValue);
272                 Thread.sleep(100);
273                 time = time + 0.1;
274                 time2 = time + data.getTimeBetweenBoatAndRov();
275 
276                 rovDepthValue = String.valueOf(df.format(data.getRovDepth()));
277                 depthBeneathBoatValue = String.valueOf(df.format(data.getDepthBeneathBoat()));
278 
279 //                //---------------------------------------------------------------------
280 //                // COMMENT OUT THIS SECTION AND UNCOMMENT THE TWO LINES ABOVE TO GET REAL DATA FROM THE ROV
281 //                counter = (counter + amount) * 1.05;
282 //                counter2 = counter2 + amount2;
283 //                rovDepthValue = String.valueOf(df.format(counter));
284 //                depthBeneathBoatValue = String.valueOf(df.format(counter2));
285 //
286 //                if (counter <= -22.0)
287 //                {
288 //                    counter = -20.0;
289 //                }
290 //                if (counter <= -20.0)
291 //                {
292 //                    amount = +1.0;
293 //                } else if (counter >= 0.0)
294 //                {
295 //                    amount = -1.0;
296 //                }
297 //
298 //                if (counter2 <= -27.0)
299 //                {
300 //                    counter2 = -25.0;
301 //                }
302 //                if (counter2 <= -26.0)
303 //                {
304 //                    amount2 = 0.1;
305 //                } else if (counter2 >= -24.0)
306 //                {
307 //                    amount2 = -0.1;
308 //                }
309 //                // ---------------------------------------------------------------------
310                 this.plot.setDataset(createDatasetLive());
311 
312             } catch (InterruptedException ex) {
313                 Logger.getLogger(EchoSounderFrame.class.getName()).log(Level.SEVERE, null, ex);
314             }
315         }
316     }
317 
318 
319     // Variables declaration - do not modify
320     private javax.swing.JMenu jMenu1;
321     private javax.swing.JMenu jMenu2;
322     private javax.swing.JMenuBar jMenuBar1;
323     private javax.swing.JMenuItem jMenuItem1;
324     private javax.swing.JMenuItem jMenuItem2;
325     private javax.swing.JPanel jPanel1;
326     // End of variables declaration
327 
328     /**
329      * This is from last year. Not used.
330      *
331      * @param o
332      * @param arg
333      */
334     @Override
335     public void update(Observable o, Object arg) {
336         //int depth = Math.round(data.getDepth() * 100);
337         // refreshGraph(0, depth);
338     }
339 }
340 
```

D:\Dokumenter\Skule\04 - NTNU\Bachelor\Github\TowedROV_GUI\src\ntnusubsea\gui\FtpClient.java

```
1 /*
2  * This code is for the bachelor thesis named "Towed-ROV".
3  * The purpose is to build a ROV which will be towed behind a surface vessel
4  * and act as a multi-sensor platform, were it shall be easy to place new
5  * sensors. There will also be a video stream from the ROV.
6  *
7  * The system consists of two Raspberry Pis in the ROV that is connected to
8  * several Arduino micro controllers. These micro controllers are connected to
9  * feedback from the actuators, the echo sounder and extra optional sensors.
10  * The external computer which is on the surface vessel is connected to a GPS,
11  * echo sounder over USB, and the ROV over ethernet. It will present and
12  * log data in addition to handle user commands for controlling the ROV.
13  */
14 package ntnusubsea.gui;
15
16 import java.io.File;
17 import java.io.FileOutputStream;
18 import java.io.IOException;
19 import java.io.PrintWriter;
20 import java.nio.file.Files;
21 import java.nio.file.Path;
22 import java.nio.file.Paths;
23 import java.util.Arrays;
24 import java.util.Collection;
25 import java.util.stream.Collectors;
26 import org.apache.commons.net.PrintCommandListener;
27 import org.apache.commons.net.ftp.FTP;
28 import org.apache.commons.net.ftp.FTPClient;
29 import org.apache.commons.net.ftp.FTPFile;
30 import org.apache.commons.net.ftp.FTPReply;
31
32 /**
33  * Creates an instance of the FtpClient class. It has the responsibility of
34  * retrieving the images from the camera RPi.
35  */
36 public class FtpClient implements Runnable {
37
38     private String server;
39     private int port;
40     private String user;
41     private String password;
42     private FTPClient ftp;
43
44     /**
45      * Constructor used to create the FtpClient.
46      *
47      * @param IP The IP of the FTP server to connect to.
48      */
49     public FtpClient(String IP) {
50         this.server = IP; //The IP address for the camera RPi.
51         this.port = 21; // The FTP port
52         this.user = "pi";
53         this.password = "";
54
55     }
```

```
56
57  /**
58   * Runs the FtpClient thread.
59   */
60  @Override
61  public void run() {
62      //this.open();
63      while (true) {
64          // be alive
65      }
66  }
67
68  /**
69   * Opens a connection to the FTP server.
70   */
71  public void open() {
72      try {
73          ftp = new FTPClient();
74
75          ftp.addProtocolCommandListener(new PrintCommandListener(new PrintWriter(System.out)));
76
77          ftp.connect(server, port);
78          int reply = ftp.getReplyCode();
79          if (!FTPReply.isPositiveCompletion(reply)) {
80              ftp.disconnect();
81              throw new IOException("Exception in connecting to FTP Server");
82          }
83
84          ftp.login(user, password);
85      } catch (IOException ex) {
86          System.out.println("IOException in FtpClient.open(): " + ex.getMessage());
87      }
88
89  }
90
91  /**
92   * Returns a list of all the files in the given path.
93   *
94   * @param path the path to list from
95   * @return a list of all the files in the given path.
96   */
97  public Collection<String> getFileList(String path) {
98      Collection<String> fileList = null;
99      try {
100         FTPFile[] files = ftp.listFiles(path);
101         fileList = Arrays.stream(files)
102                 .map(FTPFile::getName)
103                 .collect(Collectors.toList());
104
105     } catch (Exception ex) {
106         System.out.println(ex.getMessage());
107     }
108     return fileList;
109 }
110
111 /**
112  * Downloads every file from the given folder path.
113  *
```

```
114     * @param source the source
115     * @param destination the destination path of the image
116     * @param folderPath the folder path of the images
117     */
118    public void downloadFile(String source, String destination, String folderPath) {
119        try {
120            Path destinationPath = Paths.get(folderPath);
121            if (Files.notExists(destinationPath)) {
122                boolean success = (new File(folderPath)).mkdirs();
123                if (!success) {
124                    System.out.println("Directory creation failed!");
125                } else {
126                    System.out.println("Directory created at " + destination);
127                }
128            }
129            ftp.setFileType(FTP.BINARY_FILE_TYPE);
130            FileOutputStream out = new FileOutputStream(destination);
131            ftp.retrieveFile(source, out);
132        } catch (IOException ex) {
133            System.out.println(ex.getMessage());
134        }
135    }
136
137    /**
138     * Disconnects from the FTP server.
139     *
140     */
141    public void disconnect() {
142        if (ftp.isConnected()) {
143            try {
144                ftp.disconnect();
145            } catch (Exception ex) {
146                System.out.println(ex.getMessage());
147            }
148        }
149    }
150
151    /**
152     * Closes the FTP connection.
153     *
154     * @throws IOException
155     */
156    public void close() throws IOException {
157        try {
158            ftp.disconnect();
159        } catch (Exception ex) {
160            System.out.println(ex.getMessage());
161        }
162    }
163 }
164
```

D:\Dokumenter\Skule\04 - NTNU\Bachelor\Github\TowedROV_GUI\src\ntnusubsea\gui\ImagePanel.java

```
1 /*
2  * This code is for the bachelor thesis named "Towed-ROV".
3  * The purpose is to build a ROV which will be towed behind a surface vessel
4  * and act as a multi-sensor platform, were it shall be easy to place new
5  * sensors. There will also be a video stream from the ROV.
6  *
7  * The system consists of two Raspberry Pis in the ROV that is connected to
8  * several Arduino micro controllers. These micro controllers are connected to
9  * feedback from the actuators, the echo sounder and extra optional sensors.
10  * The external computer which is on the surface vessel is connected to a GPS,
11  * echo sounder over USB, and the ROV over ethernet. It will present and
12  * log data in addition to handle user commands for controlling the ROV.
13  */
14 package ntnusubsea.gui;
15
16 import java.awt.Graphics;
17 import java.awt.image.BufferedImage;
18 import javax.swing.JPanel;
19
20 /**
21  * This class is an extended version of JPanel, with the added methods required
22  * to display BufferedImages in the panel.
23  *
24  */
25 public class ImagePanel extends JPanel {
26
27     private BufferedImage image;
28     private int width, height;
29
30     /**
31      * Constructor used to create the Sheet object
32      *
33      * @param width The width of the sheet
34      * @param height The height of the sheet
35      */
36     public ImagePanel(int width, int height) {
37         setSize(width, height);
38     }
39
40     /**
41      * This methods updates the sheet to display the image used as a input
42      * parameter
43      *
44      * @param img Image to display in the component
45      */
46     public void paintSheet(BufferedImage img) {
47         image = null;
48         image = img;
49         repaint();
50     }
51
52     /**
53      * Uses the the paintComponent method of the super class and makes the
54      * component compatible with bufferedImage
55      *
```

```
56     * @param g A graphics context onto which a bufferedImage can be drawn
57     */
58    @Override
59    protected void paintComponent(Graphics g) {
60        super.paintComponent(g);
61        g.drawImage(image, 0, 0, this);
62    }
63 }
64
```

D:\Dokumenter\Skule\04 - NTNU\Bachelor\Github\TowedROV_GUI\src\ntnusubsea\gui\ImageUtils.java

```
1 /*
2  * This code is for the bachelor thesis named "Towed-ROV".
3  * The purpose is to build a ROV which will be towed behind a surface vessel
4  * and act as a multi-sensor platform, were it shall be easy to place new
5  * sensors. There will also be a video stream from the ROV.
6  *
7  * The system consists of two Raspberry Pis in the ROV that is connected to
8  * several Arduino micro controllers. These micro controllers are connected to
9  * feedback from the actuators, the echo sounder and extra optional sensors.
10  * The external computer which is on the surface vessel is connected to a GPS,
11  * echo sounder over USB, and the ROV over ethernet. It will present and
12  * log data in addition to handle user commands for controlling the ROV.
13  */
14 package ntnusubsea.gui;
15
16 import java.awt.Graphics;
17 import java.awt.Graphics2D;
18 import java.awt.GraphicsConfiguration;
19 import java.awt.GraphicsDevice;
20 import java.awt.GraphicsEnvironment;
21 import java.awt.HeadlessException;
22 import java.awt.Image;
23 import java.awt.RenderingHints;
24 import java.awt.Transparency;
25 import java.awt.image.BufferedImage;
26 import java.awt.image.ColorModel;
27 import java.awt.image.PixelGrabber;
28 import javax.swing.ImageIcon;
29
30 /**
31  * This class contains utilities to change and handle images.
32  */
33 public class ImageUtils {
34
35     /**
36      * Rotates an image the given amount of degrees
37      *
38      * @param img Image to rotate
39      * @param degree Number of degrees to rotate
40      * @return The rotated image
41      */
42     public static Image rotateImage(Image img, double degree) {
43         BufferedImage bufImg = toBufferedImage(img);
44         double angle = Math.toRadians(degree);
45         return tilt(bufImg, angle);
46     }
47
48     /**
49      * Tilts an image in a given angle
50      *
51      * @param image Image to tilt
52      * @param angle Angle to tilt the image
53      * @return The tilted image
54      */
55     public static BufferedImage tilt(BufferedImage image, double angle) {
56         double sin = Math.abs(Math.sin(angle)), cos = Math.abs(Math.cos(angle));
57         int w = image.getWidth(), h = image.getHeight();
58         int neww = (int) Math.floor(w * cos + h * sin), newh = (int) Math.floor(h * cos + w * sin);
59         GraphicsConfiguration gc = getDefaultConfiguration();
60         BufferedImage result = gc.createCompatibleImage(neww, newh, Transparency.TRANSLUCENT);
61         Graphics2D g = result.createGraphics();
62         // g.translate((neww-w)/2, (newh-h)/2);
63         g.rotate(angle, w / 2, h / 2);
64         g.drawRenderedImage(image, null);
65         g.dispose();
66         return result;
67     }
68
69     /**
70      * Returns the default configuration of the graphics device
71      *
72      * @return Default configuration of the graphics device
```

```
73     */
74    public static GraphicsConfiguration getDefaultConfiguration() {
75        GraphicsEnvironment ge = GraphicsEnvironment.getLocalGraphicsEnvironment();
76        GraphicsDevice gd = ge.getDefaultScreenDevice();
77        return gd.getDefaultConfiguration();
78    }
79
80    /**
81     * Converts an image to a bufferedImage
82     *
83     * @param image Image to convert
84     * @return The corresponding bufferedImage
85     */
86    public static BufferedImage toBufferedImage(Image image) {
87        if (image instanceof BufferedImage) {
88            return (BufferedImage) image;
89        }
90
91        // This code ensures that all the pixels in the image are loaded
92        image = new ImageIcon(image).getImage();
93
94        // Determine if the image has transparent pixels; for this method's
95        // implementation, see e661 Determining If an Image Has Transparent Pixels
96        boolean hasAlpha = hasAlpha(image);
97
98        // Create a buffered image with a format that's compatible with the screen
99        BufferedImage bimage = null;
100       GraphicsEnvironment ge = GraphicsEnvironment.getLocalGraphicsEnvironment();
101       try {
102           // Determine the type of transparency of the new buffered image
103           int transparency = Transparency.OPAQUE;
104           if (hasAlpha) {
105               transparency = Transparency.BITMASK;
106           }
107
108           // Create the buffered image
109           GraphicsDevice gs = ge.getDefaultScreenDevice();
110           GraphicsConfiguration gc = gs.getDefaultConfiguration();
111           bimage = gc.createCompatibleImage(
112                   image.getWidth(null), image.getHeight(null), transparency);
113       } catch (HeadlessException e) {
114           // The system does not have a screen
115       }
116
117       if (bimage == null) {
118           // Create a buffered image using the default color model
119           int type = BufferedImage.TYPE_INT_RGB;
120           if (hasAlpha) {
121               type = BufferedImage.TYPE_INT_ARGB;
122           }
123           bimage = new BufferedImage(image.getWidth(null), image.getHeight(null), type);
124       }
125
126       // Copy image to buffered image
127       Graphics g = bimage.createGraphics();
128
129       // Paint the image onto the buffered image
130       g.drawImage(image, 0, 0, null);
131       g.dispose();
132
133       return bimage;
134   }
135
136   // http://www.exampledepot.com/egs/java.awt.image/HasAlpha.html
137   // This method returns true if the specified image has transparent pixels
138   /**
139    * Returns true if the specified image has transparent pixels, false if not
140    *
141    * @param image the given image to check
142    * @return true if the specified image has transparent pixels, false if not
143    */
144   public static boolean hasAlpha(Image image) {
145       // If buffered image, the color model is readily available
146       if (image instanceof BufferedImage) {
147           BufferedImage bimage = (BufferedImage) image;
```

```
148             return bimage.getColorModel().hasAlpha();
149         }
150 
151         // Use a pixel grabber to retrieve the image's color model;
152         // grabbing a single pixel is usually sufficient
153         PixelGrabber pg = new PixelGrabber(image, 0, 0, 1, 1, false);
154         try {
155             pg.grabPixels();
156         } catch (InterruptedException e) {
157         }
158 
159         // Get the image's color model
160         ColorModel cm = pg.getColorModel();
161         return cm.hasAlpha();
162     }
163 
164     /**
165      * Resizes a BufferedImage. Used to make an image fit it's container.
166      *
167      * @param image BufferedImage to resize
168      * @param width Desired width of the image
169      * @param height Desired height of the image
170      * @return The resulting BufferedImage in the new size
171      */
172     public static BufferedImage resize(BufferedImage image, int width, int height) {
173         BufferedImage bi = new BufferedImage(width, height, BufferedImage.TRANSLUCENT);
174         Graphics2D g2d = (Graphics2D) bi.createGraphics();
175         g2d.addRenderingHints(new RenderingHints(RenderingHints.KEY_RENDERING, RenderingHints.VALUE_RENDER_QUALITY));
176         g2d.drawImage(image, 0, 0, width, height, null);
177         g2d.dispose();
178         return bi;
179     }
180 }
181 
```

D:\Dokumenter\Skule\04 - NTNU\Bachelor\Github\TowedROV_GUI\src\inputcontroller\InputController.java

```
1  /*
2   * This code is for the bachelor thesis named "Towed-ROV".
3   * The purpose is to build a ROV which will be towed behind a surface vessel
4   * and act as a multi-sensor platform, were it shall be easy to place new
5   * sensors. There will also be a video stream from the ROV.
6   *
7   * The system consists of two Raspberry Pis in the ROV that is connected to
8   * several Arduino micro controllers. These micro controllers are connected to
9   * feedback from the actuators, the echo sounder and extra optional sensors.
10  * The external computer which is on the surface vessel is connected to a GPS,
11  * echo sounder over USB, and the ROV over ethernet. It will present and
12  * log data in addition to handle user commands for controlling the ROV.
13  */
14 package InputController;
15
16 import com.exlumina.j360.Controller;
17 import com.exlumina.j360.ValueListener;
18 import java.io.IOException;
19 import ntnusubsea.gui.Data;
20 import ntnusubsea.gui.TCPClient;
21
22 /**
23  * This class is responsible for handling the inputs from the Xbox controller.
24  */
25 public class InputController implements Runnable {
26
27     private double btnLyGUI = 0;
28     private int btnLy = 0;
29     private int btnLx = 0;
30     private int btnRy = 0;
31     private int btnRx = 0;
32     private int angle = 0;
33
34     private static int lastAngle = 0;
35     private static float lastVal = 0;
36     private int oldPS = 0;
37     private int oldSB = 0;
38
39     private String ipAddress = "";
40     private int sendPort = 0;
41     private Data data;
42     private TCPClient client_ROV;
43
44     /**
45      * The constructor of the InputController class.
46      *
47      * @param data the Data object
48      * @param client_ROV The ROV TCP client
49      */
50     public InputController(Data data, TCPClient client_ROV) {
51         this.data = data;
52         this.client_ROV = client_ROV;
53     }
54
```

```
55  /**
56   * Runs the InputController thread. Sends the input values to the ROV TCP
57   * client to run the actuators.
58   *
59   */
60  @Override
61  public void run() {
62      ValueListener Ly = new LeftThumbYListener(this);
63      //ValueListener Lx = new LeftThumbXListener(this);
64      //ValueListener Ry = new RightThumbYListener(this);
65      ValueListener Rx = new RightThumbXListener(this);
66      Controller c1 = Controller.C1;
67
68      c1.leftThumbY.addValueChangedListener(Ly);
69      //c1.leftThumbX.addValueChangedListener(Lx);
70      //c1.rightThumbY.addValueChangedListener(Ry);
71      c1.rightThumbX.addValueChangedListener(Rx);
72
73      for (;;) {
74          try {
75              if (data.isControllerEnabled() && data.isManualMode()) {
76  //                int ps = 0;
77  //                int sb = 0;
78  //                if (btnLy <= 127)
79  //                {
80  //                    ps = btnLy + btnRx;
81  //                    sb = btnLy - btnRx;
82  //                }
83  //                if (btnLy > 127)
84  //                {
85  //                    ps = btnLy - (btnRx);
86  //                    sb = btnLy;
87  //                }
88
89                  int ps = btnLy;
90                  int sb = btnLy;
91                  if (ps != oldPS) {
92                      this.client_ROV.sendCommand("cmd_actuatorPS:" + String.valueOf(ps));
93                      oldPS = ps;
94                  }
95                  if (sb != oldSB) {
96                      this.client_ROV.sendCommand("cmd_actuatorSB:" + String.valueOf(sb));
97                      oldSB = sb;
98                  }
99
100             }
101             Thread.sleep(10);
102         } catch (InterruptedException ex) {
103             System.out.println("InterruptedException: " + ex.getMessage());
104         } catch (IOException ex) {
105             System.out.println("IOException: " + ex.getMessage());
106         } catch (Exception ex) {
107             System.out.println("Exception: " + ex.getMessage());
108         }
109     }
110 }
111
112 /**
```

```
113     * Calculates the angle by the given x and y point.
114     *
115     * @param x given x point
116     * @param y given y point
117     * @return the angle by the given x and y point.
118     */
119    public static int FindDegree(int x, int y) {
120        float value = (float) ((Math.atan2(x, y) / Math.PI) * 180f);
121        if ((x < 98 && x > -98) && (y < 98 && y > -98)) {
122            value = lastVal;
123        } else if (value < 0) {
124            value += 360f;
125        }
126        lastVal = value;
127        return Math.round(value);
128    }
129
130    /**
131     * Returns the y value of the left thumb stick
132     *
133     * @return the y value of the left thumb stick
134     */
135    public int getBtnLy() {
136        return btnLy;
137    }
138
139    /**
140     * Sets the y value of the left thumb stick
141     *
142     * @param btnLy the y value of the left thumb stick
143     */
144    public void setBtnLy(int btnLy) {
145        this.btnLy = btnLy;
146        this.btnLyGUI = (double) (this.btnLy / 100.0);
147        //System.out.println("L_Y: " + btnLy);
148    }
149
150    /**
151     * Returns the x value of the left thumb stick
152     *
153     * @return the x value of the left thumb stick
154     */
155    public int getBtnLx() {
156        return btnLx;
157    }
158
159    /**
160     * Sets the x value of the left thumb stick
161     *
162     * @param btnLx
163     */
164    public void setBtnLx(int btnLx) {
165        this.btnLx = btnLx;
166    }
167
168    /**
169     * Returns the y value of the right thumb stick
170     *
```

```
171     * @return the y value of the right thumb stick
172     */
173    public int getBtnRy() {
174        return btnRy;
175    }
176
177    /**
178     * Sets the y value of the right thumb stick
179     *
180     * @param btnRy the y value of the right thumb stick
181     */
182    public void setBtnRy(int btnRy) {
183        this.btnRy = btnRy;
184    }
185
186    /**
187     * Returns the x value of the right thumb stick
188     *
189     * @return the x value of the right thumb stick
190     */
191    public int getBtnRx() {
192        return btnRx;
193    }
194
195    /**
196     * Sets the x value of the right thumb stick
197     *
198     * @param btnRx the x value of the right thumb stick
199     */
200    public void setBtnRx(int btnRx) {
201        //System.out.println("R_X: " + btnRx);
202        this.btnRx = btnRx;
203    }
204
205    /**
206     * Returns the angle of the right thumb stick x and y values
207     *
208     * @return the angle of the right thumb stick x and y values
209     */
210    public int getAngle() {
211        this.angle = this.FindDegree(btnRx, btnRy);
212        if (this.angle != this.lastAngle) {
213            //System.out.println(angle);
214            this.lastAngle = this.angle;
215        }
216        return this.angle;
217    }
218
219    /**
220     * Returns the angle of the right thumb stick x and y values for the GUI
221     *
222     * @return the angle of the right thumb stick x and y values for the GUI
223     */
224    public int getAngleForGUI() {
225        this.angle = this.FindDegree(btnRx, btnRy);
226        if (this.angle != this.lastAngle) {
227            //System.out.println(angle);
228            this.lastAngle = this.angle;
```

```
229         }
230         int returnAngle = this.angle - 15;
231         if (returnAngle < 0) {
232             returnAngle = returnAngle + 360;
233         }
234         return returnAngle;
235     }
236 
237 }
238 
```

D:\Dokumenter\Skule\04 - NTNU\Bachelor\Github\TowedROV_GUI\src\ntnusubsea\gui\IOControlFrame.java

```
/*
 * This code is for the bachelor thesis named "Towed-ROV".
 * The purpose is to build a ROV which will be towed behind a surface vessel
 * and act as a multi-sensor platform, were it shall be easy to place new
 * sensors. There will also be a video stream from the ROV.
 *
 * The system consists of two Raspberry Pis in the ROV that is connected to
 * several Arduino micro controllers. These micro controllers are connected to
 * feedback from the actuators, the echo sounder and extra optional sensors.
 * The external computer which is on the surface vessel is connected to a GPS,
 * echo sounder over USB, and the ROV over ethernet. It will present and
 * log data in addition to handle user commands for controlling the ROV.
 */
package ntnusubsea.gui;

import java.io.File;
import java.io.IOException;
import java.util.Observable;
import java.util.Observer;
import javax.imageio.ImageIO;
import javax.swing.ImageIcon;
import javax.swing.JFrame;

/**
 * This frame lets the user control the different inputs and outputs. Sends a
 * new command whenever a button is pressed.
 *
 */
public class IOControlFrame extends javax.swing.JFrame implements Runnable, Observer {

    private Data data;
    private TCPClient client;
    private final String DIGITALOUTID = "<DigitalOut>";
    private int outputValue = 0;

    /**
     * Creates new form IOControlFrame
     *
     * @param data Data containing shared variables
     * @param client TCP client that sends commands
     */
    public IOControlFrame(Data data, TCPClient client) {
        initComponents();
        this.pack();
        this.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
        this.setLocationRelativeTo(null);
        this.data = data;
        this.client = client;
        jLabelChannel1Value.setText(data.getChannel(1));
        jLabelChannel2Value.setText(data.getChannel(2));
        jLabelChannel3Value.setText(data.getChannel(3));
        jLabelChannel4Value.setText(data.getChannel(4));
        jLabelChannel5Value.setText(data.getChannel(5));
        jLabelChannel6Value.setText(data.getChannel(6));
        jLabelChannel7Value.setText(data.getChannel(7));
        jLabelChannel8Value.setText(data.getChannel(8));
        enableIO();
    }

    /**
     * This method is called from within the constructor to initialize the form.
     * WARNING: Do NOT modify this code. The content of this method is always
     * regenerated by the Form Editor.
     */
    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed" desc="Generated Code">
    private void initComponents()
    {

        jPanel9 = new javax.swing.JPanel();
        jPanelChannel5 = new javax.swing.JPanel();
        jLabelChannel5Header = new javax.swing.JLabel();
        jToggleButtonChannel5 = new javax.swing.JToggleButton();
        jLabelChannel5Value = new javax.swing.JLabel();
        jLabelOnChannel5 = new javax.swing.JLabel();
        jLabelOffChannel5 = new javax.swing.JLabel();
        jLabelIndicatorChannel5 = new javax.swing.JLabel();
        jPanelChannel6 = new javax.swing.JPanel();
        jLabelChannel6Header = new javax.swing.JLabel();
        jToggleButtonChannel6 = new javax.swing.JToggleButton();
        jLabelChannel6Value = new javax.swing.JLabel();
        jLabelOnChannel6 = new javax.swing.JLabel();
        jLabelOffChannel6 = new javax.swing.JLabel();
        jLabelIndicatorChannel6 = new javax.swing.JLabel();
```

```
85      jPanelChannel7 = new javax.swing.JPanel();
86      jLabelChannel7Header = new javax.swing.JLabel();
87      jToggleButtonChannel7 = new javax.swing.JToggleButton();
88      jLabelChannel7Value = new javax.swing.JLabel();
89      jLabelOnChannel7 = new javax.swing.JLabel();
90      jLabelOffChannel7 = new javax.swing.JLabel();
91      jLabelIndicatorChannel7 = new javax.swing.JLabel();
92      jPanelChannel8 = new javax.swing.JPanel();
93      jLabelChannel8Header = new javax.swing.JLabel();
94      jToggleButtonChannel8 = new javax.swing.JToggleButton();
95      jLabelChannel8Value = new javax.swing.JLabel();
96      jLabelOnChannel8 = new javax.swing.JLabel();
97      jLabelOffChannel8 = new javax.swing.JLabel();
98      jLabelIndicatorChannel8 = new javax.swing.JLabel();
99      jPanelChannel1 = new javax.swing.JPanel();
100     jLabelChannel1Header = new javax.swing.JLabel();
101     jLabelChannel1Value = new javax.swing.JLabel();
102     jToggleButtonChannel1 = new javax.swing.JToggleButton();
103     jLabelIndicatorChannel1 = new javax.swing.JLabel();
104     jLabelOnChannel1 = new javax.swing.JLabel();
105     jLabelOffChannel1 = new javax.swing.JLabel();
106     jPanelChannel2 = new javax.swing.JPanel();
107     jLabelChannel2Header = new javax.swing.JLabel();
108     jLabelChannel2Value = new javax.swing.JLabel();
109     jToggleButtonChannel2 = new javax.swing.JToggleButton();
110     jLabelIndicatorChannel2 = new javax.swing.JLabel();
111     jLabelOnChannel2 = new javax.swing.JLabel();
112     jLabelOffChannel2 = new javax.swing.JLabel();
113     jPanelChannel3 = new javax.swing.JPanel();
114     jLabelChannel3Header = new javax.swing.JLabel();
115     jLabelChannel3Value = new javax.swing.JLabel();
116     jToggleButtonChannel3 = new javax.swing.JToggleButton();
117     jLabelIndicatorChannel3 = new javax.swing.JLabel();
118     jLabelOnChannel3 = new javax.swing.JLabel();
119     jLabelOffChannel3 = new javax.swing.JLabel();
120     jPanelChannel4 = new javax.swing.JPanel();
121     jLabelChannel4Header = new javax.swing.JLabel();
122     jLabelChannel4Value = new javax.swing.JLabel();
123     jToggleButtonChannel4 = new javax.swing.JToggleButton();
124     jLabelIndicatorChannel4 = new javax.swing.JLabel();
125     jLabelOnChannel4 = new javax.swing.JLabel();
126     jLabelOffChannel4 = new javax.swing.JLabel();
127
128     setDefaultCloseOperation(javax.swing.WindowConstants.DISPOSE_ON_CLOSE);
129     setTitle("I/O Controller");
130     setBackground(new java.awt.Color(39, 44, 50));
131     setForeground(new java.awt.Color(28, 28, 28));
132     setResizable(false);
133
134     jPanel9.setBackground(new java.awt.Color(39, 44, 50));
135
136     jPanelChannel5.setBackground(new java.awt.Color(35, 39, 43));
137     jPanelChannel5.setPreferredSize(new java.awt.Dimension(170, 191));
138
139     jLabelChannel5Header.setBackground(new java.awt.Color(28, 28, 28));
140     jLabelChannel5Header.setFont(new java.awt.Font("Tahoma", 0, 14)); // NOI18N
141     jLabelChannel5Header.setForeground(new java.awt.Color(255, 255, 255));
142     jLabelChannel5Header.setText("Channel 5 (Output)");
143
144     jToggleButtonChannel5.setBackground(new java.awt.Color(35, 39, 43));
145     jToggleButtonChannel5.setForeground(new java.awt.Color(35, 39, 43));
146     jToggleButtonChannel5.setIcon(new javax.swing.ImageIcon(getClass().getResource("/ntnusubsea/gui/Images/switch-off-vertical.gif"))); // NOI18N
147     jToggleButtonChannel5.setBorder(null);
148     jToggleButtonChannel5.setContentAreaFilled(false);
149     jToggleButtonChannel5.setEnabled(false);
150     jToggleButtonChannel5.setSelectedIcon(new javax.swing.ImageIcon(getClass().getResource("/ntnusubsea/gui/Images/switch-on-vertical.gif"))); // NOI18N
151     jToggleButtonChannel5.addActionListener(new java.awt.event.ActionListener()
152     {
153         public void actionPerformed(java.awt.event.ActionEvent evt)
154         {
155             jToggleButtonChannel5ActionPerformed(evt);
156         }
157     });
158
159     jLabelChannel5Value.setBackground(new java.awt.Color(28, 28, 28));
160     jLabelChannel5Value.setFont(new java.awt.Font("Tahoma", 0, 14)); // NOI18N
161     jLabelChannel5Value.setForeground(new java.awt.Color(255, 255, 255));
162     jLabelChannel5Value.setText("jLabel2");
163
164     jLabelOnChannel5.setBackground(new java.awt.Color(28, 28, 28));
165     jLabelOnChannel5.setForeground(new java.awt.Color(255, 255, 255));
166     jLabelOnChannel5.setText("On");
167
168     jLabelOffChannel5.setBackground(new java.awt.Color(28, 28, 28));
169     jLabelOffChannel5.setForeground(new java.awt.Color(255, 255, 255));
170     jLabelOffChannel5.setText("Off");
171
```

```
172     jLabelIndicatorChannel5.setBackground(new java.awt.Color(35, 39, 43));
173     jLabelIndicatorChannel5.setIcon(new javax.swing.ImageIcon(getClass().getResource("/ntnusubsea/gui/Images/IO-Off.gif"))); // NOI18N
174     jLabelIndicatorChannel5.setOpaque(true);
175
176     javax.swing.GroupLayout jPanelChannel5Layout = new javax.swing.GroupLayout(jPanelChannel5);
177     jPanelChannel5.setLayout(jPanelChannel5Layout);
178     jPanelChannel5Layout.setHorizontalGroup(
179         jPanelChannel5Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
180         .addComponent(jLabelChannel5Header, javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
181         .addGroup(jPanelChannel5Layout.createSequentialGroup()
182             .addContainerGap()
183             .addComponent(jToggleButtonChannel5, javax.swing.GroupLayout.PREFERRED_SIZE, 46, javax.swing.GroupLayout.PREFERRED_SIZE)
184             .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
185             .addComponent(jLabelIndicatorChannel5, javax.swing.GroupLayout.PREFERRED_SIZE, 54, javax.swing.GroupLayout.PREFERRED_SIZE)
186             .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
187             .addGroup(jPanelChannel5Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
188                 .addComponent(jLabelOnChannel5, javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE
189                 .addComponent(jLabelOffChannel5, javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE
190         .addComponent(jLabelChannel5Value, javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
191     );
192     jPanelChannel5Layout.setVerticalGroup(
193         jPanelChannel5Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
194         .addGroup(jPanelChannel5Layout.createSequentialGroup()
195             .addComponent(jLabelChannel5Header, javax.swing.GroupLayout.PREFERRED_SIZE, 28, javax.swing.GroupLayout.PREFERRED_SIZE)
196             .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
197             .addComponent(jLabelChannel5Value, javax.swing.GroupLayout.PREFERRED_SIZE, 50, javax.swing.GroupLayout.PREFERRED_SIZE)
198             .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
199             .addGroup(jPanelChannel5Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
200                 .addGroup(jPanelChannel5Layout.createSequentialGroup()
201                     .addGap(5, 5, 5)
202                     .addComponent(jLabelOnChannel5, javax.swing.GroupLayout.PREFERRED_SIZE, 20, javax.swing.GroupLayout.PREFERRED_SIZE)
203                     .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
204                     .addComponent(jLabelOffChannel5, javax.swing.GroupLayout.PREFERRED_SIZE, 20, javax.swing.GroupLayout.PREFERRED_SIZE))
205                 .addComponent(jToggleButtonChannel5, javax.swing.GroupLayout.PREFERRED_SIZE, 63, javax.swing.GroupLayout.PREFERRED_SIZE)
206                 .addComponent(jLabelIndicatorChannel5, javax.swing.GroupLayout.PREFERRED_SIZE, 63, javax.swing.GroupLayout.PREFERRED_SIZE))
207             .addContainerGap())
208     );
209
210     jPanelChannel6.setBackground(new java.awt.Color(35, 39, 43));
211     jPanelChannel6.setPreferredSize(new java.awt.Dimension(170, 191));
212
213     jLabelChannel6Header.setBackground(new java.awt.Color(28, 28, 28));
214     jLabelChannel6Header.setFont(new java.awt.Font("Tahoma", 0, 14)); // NOI18N
215     jLabelChannel6Header.setForeground(new java.awt.Color(255, 255, 255));
216     jLabelChannel6Header.setText("Channel 6 (Output)");
217
218     jToggleButtonChannel6.setBackground(new java.awt.Color(35, 39, 43));
219     jToggleButtonChannel6.setForeground(new java.awt.Color(35, 39, 43));
220     jToggleButtonChannel6.setIcon(new javax.swing.ImageIcon(getClass().getResource("/ntnusubsea/gui/Images/switch-off-vertical.gif"))); // NOI18N
221     jToggleButtonChannel6.setBorder(null);
222     jToggleButtonChannel6.setContentAreaFilled(false);
223     jToggleButtonChannel6.setEnabled(false);
224     jToggleButtonChannel6.setSelectedIcon(new javax.swing.ImageIcon(getClass().getResource("/ntnusubsea/gui/Images/switch-on-vertical.gif"))); // NOI18N
225     jToggleButtonChannel6.addActionListener(new java.awt.event.ActionListener()
226     {
227         public void actionPerformed(java.awt.event.ActionEvent evt)
228         {
229             jToggleButtonChannel6ActionPerformed(evt);
230         }
231     });
232
233     jLabelChannel6Value.setBackground(new java.awt.Color(28, 28, 28));
234     jLabelChannel6Value.setFont(new java.awt.Font("Tahoma", 0, 14)); // NOI18N
235     jLabelChannel6Value.setForeground(new java.awt.Color(255, 255, 255));
236     jLabelChannel6Value.setText("jLabel2");
237
238     jLabelOnChannel6.setBackground(new java.awt.Color(28, 28, 28));
239     jLabelOnChannel6.setForeground(new java.awt.Color(255, 255, 255));
240     jLabelOnChannel6.setText("On");
241
242     jLabelOffChannel6.setBackground(new java.awt.Color(28, 28, 28));
243     jLabelOffChannel6.setForeground(new java.awt.Color(255, 255, 255));
244     jLabelOffChannel6.setText("Off");
245
246     jLabelIndicatorChannel6.setBackground(new java.awt.Color(35, 39, 43));
247     jLabelIndicatorChannel6.setIcon(new javax.swing.ImageIcon(getClass().getResource("/ntnusubsea/gui/Images/IO-Off.gif"))); // NOI18N
248     jLabelIndicatorChannel6.setText("jLabel26");
249     jLabelIndicatorChannel6.setOpaque(true);
250
251     javax.swing.GroupLayout jPanelChannel6Layout = new javax.swing.GroupLayout(jPanelChannel6);
252     jPanelChannel6.setLayout(jPanelChannel6Layout);
253     jPanelChannel6Layout.setHorizontalGroup(
254         jPanelChannel6Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
255         .addComponent(jLabelChannel6Header, javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
256         .addGroup(jPanelChannel6Layout.createSequentialGroup()
257             .addContainerGap()
258             .addComponent(jToggleButtonChannel6, javax.swing.GroupLayout.PREFERRED_SIZE, 46, javax.swing.GroupLayout.PREFERRED_SIZE)
```

```
            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
            .addComponent(jLabelIndicatorChannel6, javax.swing.GroupLayout.PREFERRED_SIZE, 58, javax.swing.GroupLayout.PREFERRED_SIZE)
            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
            .addGroup(jPanelChannel6Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                .addComponent(jLabelOffChannel6, javax.swing.GroupLayout.DEFAULT_SIZE, 48, Short.MAX_VALUE)
                .addComponent(jLabelOnChannel6, javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE
        .addComponent(jLabelChannel6Value, javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
    );
    jPanelChannel6Layout.setVerticalGroup(
        jPanelChannel6Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addGroup(jPanelChannel6Layout.createSequentialGroup()
            .addComponent(jLabelChannel6Header, javax.swing.GroupLayout.PREFERRED_SIZE, 28, javax.swing.GroupLayout.PREFERRED_SIZE)
            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
            .addComponent(jLabelChannel6Value, javax.swing.GroupLayout.PREFERRED_SIZE, 50, javax.swing.GroupLayout.PREFERRED_SIZE)
            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
            .addGroup(jPanelChannel6Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                .addGroup(jPanelChannel6Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                    .addGroup(javax.swing.GroupLayout.Alignment.TRAILING, jPanelChannel6Layout.createSequentialGroup()
                        .addComponent(jLabelOnChannel6, javax.swing.GroupLayout.PREFERRED_SIZE, 20, javax.swing.GroupLayout.PREFERRED_SIZE)
                        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                        .addComponent(jLabelOffChannel6, javax.swing.GroupLayout.PREFERRED_SIZE, 20, javax.swing.GroupLayout.PREFERRED_SIZE)
                        .addGap(11, 11, 11))
                    .addComponent(jLabelIndicatorChannel6, javax.swing.GroupLayout.PREFERRED_SIZE, 63, javax.swing.GroupLayout.PREFERRED_SIZE))
                .addComponent(jToggleButtonChannel6, javax.swing.GroupLayout.PREFERRED_SIZE, 63, javax.swing.GroupLayout.PREFERRED_SIZE))
            .addContainerGap())
    );

    jPanelChannel7.setBackground(new java.awt.Color(35, 39, 43));
    jPanelChannel7.setPreferredSize(new java.awt.Dimension(170, 191));

    jLabelChannel7Header.setBackground(new java.awt.Color(28, 28, 28));
    jLabelChannel7Header.setFont(new java.awt.Font("Tahoma", 0, 14)); // NOI18N
    jLabelChannel7Header.setForeground(new java.awt.Color(255, 255, 255));
    jLabelChannel7Header.setText("Channel 7 (Output)");

    jToggleButtonChannel7.setBackground(new java.awt.Color(35, 39, 43));
    jToggleButtonChannel7.setForeground(new java.awt.Color(35, 39, 43));
    jToggleButtonChannel7.setIcon(new javax.swing.ImageIcon(getClass().getResource("/ntnusubsea/gui/Images/switch-off-vertical.gif"))); // NOI18N
    jToggleButtonChannel7.setBorder(null);
    jToggleButtonChannel7.setContentAreaFilled(false);
    jToggleButtonChannel7.setEnabled(false);
    jToggleButtonChannel7.setSelectedIcon(new javax.swing.ImageIcon(getClass().getResource("/ntnusubsea/gui/Images/switch-on-vertical.gif"))); // NOI18N
    jToggleButtonChannel7.addActionListener(new java.awt.event.ActionListener()
    {
        public void actionPerformed(java.awt.event.ActionEvent evt)
        {
            jToggleButtonChannel7ActionPerformed(evt);
        }
    });

    jLabelChannel7Value.setBackground(new java.awt.Color(28, 28, 28));
    jLabelChannel7Value.setFont(new java.awt.Font("Tahoma", 0, 14)); // NOI18N
    jLabelChannel7Value.setForeground(new java.awt.Color(255, 255, 255));
    jLabelChannel7Value.setText("jLabel2");

    jLabelOnChannel7.setBackground(new java.awt.Color(28, 28, 28));
    jLabelOnChannel7.setForeground(new java.awt.Color(255, 255, 255));
    jLabelOnChannel7.setText("On");

    jLabelOffChannel7.setBackground(new java.awt.Color(28, 28, 28));
    jLabelOffChannel7.setForeground(new java.awt.Color(255, 255, 255));
    jLabelOffChannel7.setText("Off");

    jLabelIndicatorChannel7.setBackground(new java.awt.Color(35, 39, 43));
    jLabelIndicatorChannel7.setIcon(new javax.swing.ImageIcon(getClass().getResource("/ntnusubsea/gui/Images/IO-Off.gif"))); // NOI18N
    jLabelIndicatorChannel7.setOpaque(true);

    javax.swing.GroupLayout jPanelChannel7Layout = new javax.swing.GroupLayout(jPanelChannel7);
    jPanelChannel7.setLayout(jPanelChannel7Layout);
    jPanelChannel7Layout.setHorizontalGroup(
        jPanelChannel7Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addComponent(jLabelChannel7Header, javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
        .addGroup(jPanelChannel7Layout.createSequentialGroup()
            .addContainerGap()
            .addComponent(jToggleButtonChannel7, javax.swing.GroupLayout.PREFERRED_SIZE, 46, javax.swing.GroupLayout.PREFERRED_SIZE)
            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
            .addComponent(jLabelIndicatorChannel7, javax.swing.GroupLayout.PREFERRED_SIZE, 58, javax.swing.GroupLayout.PREFERRED_SIZE)
            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
            .addGroup(jPanelChannel7Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                .addComponent(jLabelOffChannel7, javax.swing.GroupLayout.DEFAULT_SIZE, 48, Short.MAX_VALUE)
                .addComponent(jLabelOnChannel7, javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE
        .addComponent(jLabelChannel7Value, javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
    );
    jPanelChannel7Layout.setVerticalGroup(
        jPanelChannel7Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addGroup(jPanelChannel7Layout.createSequentialGroup()
            .addComponent(jLabelChannel7Header, javax.swing.GroupLayout.PREFERRED_SIZE, 28, javax.swing.GroupLayout.PREFERRED_SIZE)
```

```
346             .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
347             .addComponent(jLabelChannel7Value, javax.swing.GroupLayout.PREFERRED_SIZE, 50, javax.swing.GroupLayout.PREFERRED_SIZE)
348             .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
349             .addGroup(jPanelChannel7Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
350                 .addGroup(javax.swing.GroupLayout.Alignment.TRAILING, jPanelChannel7Layout.createSequentialGroup()
351                     .addComponent(jLabelOnChannel7, javax.swing.GroupLayout.PREFERRED_SIZE, 20, javax.swing.GroupLayout.PREFERRED_SIZE)
352                     .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
353                     .addComponent(jLabelOffChannel7, javax.swing.GroupLayout.PREFERRED_SIZE, 20, javax.swing.GroupLayout.PREFERRED_SIZE)
354                     .addGap(11, 11, 11))
355                 .addComponent(jToggleButtonChannel7, javax.swing.GroupLayout.Alignment.TRAILING, javax.swing.GroupLayout.PREFERRED_SIZE, 63, java
356                 .addComponent(jLabelIndicatorChannel7, javax.swing.GroupLayout.Alignment.TRAILING, javax.swing.GroupLayout.PREFERRED_SIZE, 63, jav
357     );
358
359     jPanelChannel8.setBackground(new java.awt.Color(35, 39, 43));
360     jPanelChannel8.setPreferredSize(new java.awt.Dimension(170, 191));
361
362     jLabelChannel8Header.setBackground(new java.awt.Color(28, 28, 28));
363     jLabelChannel8Header.setFont(new java.awt.Font("Tahoma", 0, 14)); // NOI18N
364     jLabelChannel8Header.setForeground(new java.awt.Color(255, 255, 255));
365     jLabelChannel8Header.setText("Channel 8 (Output)");
366
367     jToggleButtonChannel8.setBackground(new java.awt.Color(35, 39, 43));
368     jToggleButtonChannel8.setForeground(new java.awt.Color(35, 39, 43));
369     jToggleButtonChannel8.setIcon(new javax.swing.ImageIcon(getClass().getResource("/ntnusubsea/gui/Images/switch-off-vertical.gif"))); // NOI18N
370     jToggleButtonChannel8.setBorder(null);
371     jToggleButtonChannel8.setContentAreaFilled(false);
372     jToggleButtonChannel8.setEnabled(false);
373     jToggleButtonChannel8.setSelectedIcon(new javax.swing.ImageIcon(getClass().getResource("/ntnusubsea/gui/Images/switch-on-vertical.gif"))); // NOI18N
374     jToggleButtonChannel8.addActionListener(new java.awt.event.ActionListener()
375     {
376         public void actionPerformed(java.awt.event.ActionEvent evt)
377         {
378             jToggleButtonChannel8ActionPerformed(evt);
379         }
380     });
381
382     jLabelChannel8Value.setBackground(new java.awt.Color(28, 28, 28));
383     jLabelChannel8Value.setFont(new java.awt.Font("Tahoma", 0, 14)); // NOI18N
384     jLabelChannel8Value.setForeground(new java.awt.Color(255, 255, 255));
385     jLabelChannel8Value.setText("jLabel2");
386
387     jLabelOnChannel8.setBackground(new java.awt.Color(28, 28, 28));
388     jLabelOnChannel8.setForeground(new java.awt.Color(255, 255, 255));
389     jLabelOnChannel8.setText("On");
390
391     jLabelOffChannel8.setBackground(new java.awt.Color(28, 28, 28));
392     jLabelOffChannel8.setForeground(new java.awt.Color(255, 255, 255));
393     jLabelOffChannel8.setText("Off");
394
395     jLabelIndicatorChannel8.setBackground(new java.awt.Color(35, 39, 43));
396     jLabelIndicatorChannel8.setIcon(new javax.swing.ImageIcon(getClass().getResource("/ntnusubsea/gui/Images/IO-Off.gif"))); // NOI18N
397     jLabelIndicatorChannel8.setOpaque(true);
398
399     javax.swing.GroupLayout jPanelChannel8Layout = new javax.swing.GroupLayout(jPanelChannel8);
400     jPanelChannel8.setLayout(jPanelChannel8Layout);
401     jPanelChannel8Layout.setHorizontalGroup(
402         jPanelChannel8Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
403         .addComponent(jLabelChannel8Header, javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
404         .addGroup(jPanelChannel8Layout.createSequentialGroup()
405             .addContainerGap()
406             .addComponent(jToggleButtonChannel8, javax.swing.GroupLayout.PREFERRED_SIZE, 46, javax.swing.GroupLayout.PREFERRED_SIZE)
407             .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
408             .addComponent(jLabelIndicatorChannel8, javax.swing.GroupLayout.PREFERRED_SIZE, 58, javax.swing.GroupLayout.PREFERRED_SIZE)
409             .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
410             .addGroup(jPanelChannel8Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
411                 .addComponent(jLabelOffChannel8, javax.swing.GroupLayout.DEFAULT_SIZE, 48, Short.MAX_VALUE)
412                 .addComponent(jLabelOnChannel8, javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE
413         .addComponent(jLabelChannel8Value, javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
414     );
415     jPanelChannel8Layout.setVerticalGroup(
416         jPanelChannel8Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
417         .addGroup(jPanelChannel8Layout.createSequentialGroup()
418             .addComponent(jLabelChannel8Header, javax.swing.GroupLayout.PREFERRED_SIZE, 28, javax.swing.GroupLayout.PREFERRED_SIZE)
419             .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
420             .addComponent(jLabelChannel8Value, javax.swing.GroupLayout.PREFERRED_SIZE, 50, javax.swing.GroupLayout.PREFERRED_SIZE)
421             .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
422             .addGroup(jPanelChannel8Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
423                 .addComponent(jLabelIndicatorChannel8, javax.swing.GroupLayout.Alignment.TRAILING, javax.swing.GroupLayout.PREFERRED_SIZE, 63, jav
424                 .addComponent(jToggleButtonChannel8, javax.swing.GroupLayout.Alignment.TRAILING, javax.swing.GroupLayout.PREFERRED_SIZE, 63, java
425                 .addGroup(javax.swing.GroupLayout.Alignment.TRAILING, jPanelChannel8Layout.createSequentialGroup()
426                     .addComponent(jLabelOnChannel8, javax.swing.GroupLayout.PREFERRED_SIZE, 20, javax.swing.GroupLayout.PREFERRED_SIZE)
427                     .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
428                     .addComponent(jLabelOffChannel8, javax.swing.GroupLayout.PREFERRED_SIZE, 20, javax.swing.GroupLayout.PREFERRED_SIZE)
429                     .addGap(11, 11, 11))))
430     );
431
432     jPanelChannel1.setBackground(new java.awt.Color(35, 39, 43));
```

```
433     jPanelChannel1.setPreferredSize(new java.awt.Dimension(170, 191));
434
435     jLabelChannel1Header.setBackground(new java.awt.Color(28, 28, 28));
436     jLabelChannel1Header.setFont(new java.awt.Font("Tahoma", 0, 14)); // NOI18N
437     jLabelChannel1Header.setForeground(new java.awt.Color(255, 255, 255));
438     jLabelChannel1Header.setText("Channel 1 (Input)");
439
440     jLabelChannel1Value.setBackground(new java.awt.Color(28, 28, 28));
441     jLabelChannel1Value.setFont(new java.awt.Font("Tahoma", 0, 14)); // NOI18N
442     jLabelChannel1Value.setForeground(new java.awt.Color(255, 255, 255));
443     jLabelChannel1Value.setText("jLabel2");
444
445     jToggleButtonChannel1.setBackground(new java.awt.Color(35, 39, 43));
446     jToggleButtonChannel1.setForeground(new java.awt.Color(35, 39, 43));
447     jToggleButtonChannel1.setIcon(new javax.swing.ImageIcon(getClass().getResource("/ntnusubsea/gui/Images/switch-off-vertical.gif"))); // NOI18N
448     jToggleButtonChannel1.setBorder(null);
449     jToggleButtonChannel1.setContentAreaFilled(false);
450     jToggleButtonChannel1.setEnabled(false);
451     jToggleButtonChannel1.setSelectedIcon(new javax.swing.ImageIcon(getClass().getResource("/ntnusubsea/gui/Images/switch-on-vertical.gif"))); // NOI18N
452     jToggleButtonChannel1.addActionListener(new java.awt.event.ActionListener()
453     {
454         public void actionPerformed(java.awt.event.ActionEvent evt)
455         {
456             jToggleButtonChannel1ActionPerformed(evt);
457         }
458     });
459
460     jLabelIndicatorChannel1.setBackground(new java.awt.Color(35, 39, 43));
461     jLabelIndicatorChannel1.setIcon(new javax.swing.ImageIcon(getClass().getResource("/ntnusubsea/gui/Images/IO-Off.gif"))); // NOI18N
462     jLabelIndicatorChannel1.setOpaque(true);
463
464     jLabelOnChannel1.setBackground(new java.awt.Color(28, 28, 28));
465     jLabelOnChannel1.setForeground(new java.awt.Color(255, 255, 255));
466     jLabelOnChannel1.setText("On");
467
468     jLabelOffChannel1.setBackground(new java.awt.Color(28, 28, 28));
469     jLabelOffChannel1.setForeground(new java.awt.Color(255, 255, 255));
470     jLabelOffChannel1.setText("Off");
471
472     javax.swing.GroupLayout jPanelChannel1Layout = new javax.swing.GroupLayout(jPanelChannel1);
473     jPanelChannel1.setLayout(jPanelChannel1Layout);
474     jPanelChannel1Layout.setHorizontalGroup(
475         jPanelChannel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
476         .addComponent(jLabelChannel1Header, javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
477         .addComponent(jLabelChannel1Value, javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
478         .addGroup(jPanelChannel1Layout.createSequentialGroup()
479             .addContainerGap()
480             .addComponent(jToggleButtonChannel1, javax.swing.GroupLayout.PREFERRED_SIZE, 46, javax.swing.GroupLayout.PREFERRED_SIZE)
481             .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
482             .addComponent(jLabelIndicatorChannel1, javax.swing.GroupLayout.PREFERRED_SIZE, 54, javax.swing.GroupLayout.PREFERRED_SIZE)
483             .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
484             .addGroup(jPanelChannel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
485                 .addComponent(jLabelOffChannel1, javax.swing.GroupLayout.DEFAULT_SIZE, 46, Short.MAX_VALUE)
486                 .addComponent(jLabelOnChannel1, javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE
487             .addContainerGap())
488     );
489     jPanelChannel1Layout.setVerticalGroup(
490         jPanelChannel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
491         .addGroup(jPanelChannel1Layout.createSequentialGroup()
492             .addGroup(jPanelChannel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING)
493                 .addComponent(jLabelIndicatorChannel1, javax.swing.GroupLayout.PREFERRED_SIZE, 63, javax.swing.GroupLayout.PREFERRED_SIZE)
494                 .addGroup(jPanelChannel1Layout.createSequentialGroup()
495                     .addComponent(jLabelChannel1Header, javax.swing.GroupLayout.PREFERRED_SIZE, 28, javax.swing.GroupLayout.PREFERRED_SIZE)
496                     .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
497                     .addComponent(jLabelChannel1Value, javax.swing.GroupLayout.PREFERRED_SIZE, 60, javax.swing.GroupLayout.PREFERRED_SIZE)
498                     .addGroup(jPanelChannel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
499                         .addGroup(jPanelChannel1Layout.createSequentialGroup()
500                             .addGap(21, 21, 21)
501                             .addComponent(jLabelOnChannel1, javax.swing.GroupLayout.PREFERRED_SIZE, 20, javax.swing.GroupLayout.PREFERRED_SIZE)
502                             .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
503                             .addComponent(jLabelOffChannel1, javax.swing.GroupLayout.PREFERRED_SIZE, 20, javax.swing.GroupLayout.PREFERRED_SIZE)
504                             .addGap(8, 8, 8))
505                         .addGroup(javax.swing.GroupLayout.Alignment.TRAILING, jPanelChannel1Layout.createSequentialGroup()
506                             .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
507                             .addComponent(jToggleButtonChannel1, javax.swing.GroupLayout.PREFERRED_SIZE, 63, javax.swing.GroupLayout.PREFERRED_SIZE
508             .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
509     );
510
511     jPanelChannel2.setBackground(new java.awt.Color(35, 39, 43));
512     jPanelChannel2.setPreferredSize(new java.awt.Dimension(170, 191));
513
514     jLabelChannel2Header.setBackground(new java.awt.Color(28, 28, 28));
515     jLabelChannel2Header.setFont(new java.awt.Font("Tahoma", 0, 14)); // NOI18N
516     jLabelChannel2Header.setForeground(new java.awt.Color(255, 255, 255));
517     jLabelChannel2Header.setText("Channel 2 (Input)");
518
519     jLabelChannel2Value.setBackground(new java.awt.Color(28, 28, 28));
```

```
520     jLabelChannel2Value.setFont(new java.awt.Font("Tahoma", 0, 14)); // NOI18N
521     jLabelChannel2Value.setForeground(new java.awt.Color(255, 255, 255));
522     jLabelChannel2Value.setText("jLabel2");
523
524     jToggleButtonChannel2.setBackground(new java.awt.Color(35, 39, 43));
525     jToggleButtonChannel2.setForeground(new java.awt.Color(35, 39, 43));
526     jToggleButtonChannel2.setIcon(new javax.swing.ImageIcon(getClass().getResource("/ntnusubsea/gui/Images/switch-off-vertical.gif"))); // NOI18N
527     jToggleButtonChannel2.setBorder(null);
528     jToggleButtonChannel2.setContentAreaFilled(false);
529     jToggleButtonChannel2.setEnabled(false);
530     jToggleButtonChannel2.setSelectedIcon(new javax.swing.ImageIcon(getClass().getResource("/ntnusubsea/gui/Images/switch-on-vertical.gif"))); // NOI18N
531     jToggleButtonChannel2.addActionListener(new java.awt.event.ActionListener()
532     {
533        public void actionPerformed(java.awt.event.ActionEvent evt)
534        {
535           jToggleButtonChannel2ActionPerformed(evt);
536        }
537     });
538
539     jLabelIndicatorChannel2.setBackground(new java.awt.Color(35, 39, 43));
540     jLabelIndicatorChannel2.setIcon(new javax.swing.ImageIcon(getClass().getResource("/ntnusubsea/gui/Images/IO-Off.gif"))); // NOI18N
541     jLabelIndicatorChannel2.setOpaque(true);
542
543     jLabelOnChannel2.setBackground(new java.awt.Color(28, 28, 28));
544     jLabelOnChannel2.setForeground(new java.awt.Color(255, 255, 255));
545     jLabelOnChannel2.setText("On");
546
547     jLabelOffChannel2.setBackground(new java.awt.Color(28, 28, 28));
548     jLabelOffChannel2.setForeground(new java.awt.Color(255, 255, 255));
549     jLabelOffChannel2.setText("Off");
550
551     javax.swing.GroupLayout jPanelChannel2Layout = new javax.swing.GroupLayout(jPanelChannel2);
552     jPanelChannel2.setLayout(jPanelChannel2Layout);
553     jPanelChannel2Layout.setHorizontalGroup(
554        jPanelChannel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
555        .addComponent(jLabelChannel2Header, javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
556        .addComponent(jLabelChannel2Value, javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
557        .addGroup(jPanelChannel2Layout.createSequentialGroup()
558           .addContainerGap()
559           .addComponent(jToggleButtonChannel2, javax.swing.GroupLayout.PREFERRED_SIZE, 46, javax.swing.GroupLayout.PREFERRED_SIZE)
560           .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
561           .addComponent(jLabelIndicatorChannel2, javax.swing.GroupLayout.PREFERRED_SIZE, 54, javax.swing.GroupLayout.PREFERRED_SIZE)
562           .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
563           .addGroup(jPanelChannel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
564              .addGroup(jPanelChannel2Layout.createSequentialGroup()
565                 .addComponent(jLabelOnChannel2, javax.swing.GroupLayout.DEFAULT_SIZE, 36, Short.MAX_VALUE)
566                 .addGap(16, 16, 16))
567              .addGroup(jPanelChannel2Layout.createSequentialGroup()
568                 .addComponent(jLabelOffChannel2, javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALU
569                 .addContainerGap())))
570     );
571     jPanelChannel2Layout.setVerticalGroup(
572        jPanelChannel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
573        .addGroup(jPanelChannel2Layout.createSequentialGroup()
574           .addComponent(jLabelChannel2Header, javax.swing.GroupLayout.PREFERRED_SIZE, 28, javax.swing.GroupLayout.PREFERRED_SIZE)
575           .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
576           .addComponent(jLabelChannel2Value, javax.swing.GroupLayout.PREFERRED_SIZE, 60, javax.swing.GroupLayout.PREFERRED_SIZE)
577           .addGroup(jPanelChannel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
578              .addGroup(jPanelChannel2Layout.createSequentialGroup()
579                 .addGap(19, 19, 19)
580                 .addComponent(jLabelOnChannel2, javax.swing.GroupLayout.PREFERRED_SIZE, 20, javax.swing.GroupLayout.PREFERRED_SIZE)
581                 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
582                 .addComponent(jLabelOffChannel2, javax.swing.GroupLayout.PREFERRED_SIZE, 20, javax.swing.GroupLayout.PREFERRED_SIZE))
583              .addGroup(jPanelChannel2Layout.createSequentialGroup()
584                 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
585                 .addGroup(jPanelChannel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
586                    .addComponent(jLabelIndicatorChannel2, javax.swing.GroupLayout.PREFERRED_SIZE, 63, javax.swing.GroupLayout.PREFERRED_SIZE)
587                    .addComponent(jToggleButtonChannel2, javax.swing.GroupLayout.PREFERRED_SIZE, 63, javax.swing.GroupLayout.PREFERRED_SIZE)))
588           .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
589     );
590
591     jPanelChannel3.setBackground(new java.awt.Color(35, 39, 43));
592     jPanelChannel3.setPreferredSize(new java.awt.Dimension(170, 191));
593
594     jLabelChannel3Header.setBackground(new java.awt.Color(28, 28, 28));
595     jLabelChannel3Header.setFont(new java.awt.Font("Tahoma", 0, 14)); // NOI18N
596     jLabelChannel3Header.setForeground(new java.awt.Color(255, 255, 255));
597     jLabelChannel3Header.setText("Channel 3 (Input)");
598
599     jLabelChannel3Value.setBackground(new java.awt.Color(28, 28, 28));
600     jLabelChannel3Value.setFont(new java.awt.Font("Tahoma", 0, 14)); // NOI18N
601     jLabelChannel3Value.setForeground(new java.awt.Color(255, 255, 255));
602     jLabelChannel3Value.setText("jLabel2");
603
604     jToggleButtonChannel3.setBackground(new java.awt.Color(35, 39, 43));
605     jToggleButtonChannel3.setForeground(new java.awt.Color(35, 39, 43));
606     jToggleButtonChannel3.setIcon(new javax.swing.ImageIcon(getClass().getResource("/ntnusubsea/gui/Images/switch-off-vertical.gif"))); // NOI18N
```

```
607     jToggleButtonChannel3.setBorder(null);
608     jToggleButtonChannel3.setContentAreaFilled(false);
609     jToggleButtonChannel3.setEnabled(false);
610     jToggleButtonChannel3.setSelectedIcon(new javax.swing.ImageIcon(getClass().getResource("/ntnusubsea/gui/Images/switch-on-vertical.gif"))); // NOI18N
611     jToggleButtonChannel3.addActionListener(new java.awt.event.ActionListener()
612     {
613         public void actionPerformed(java.awt.event.ActionEvent evt)
614         {
615             jToggleButtonChannel3ActionPerformed(evt);
616         }
617     });
618
619     jLabelIndicatorChannel3.setBackground(new java.awt.Color(35, 39, 43));
620     jLabelIndicatorChannel3.setIcon(new javax.swing.ImageIcon(getClass().getResource("/ntnusubsea/gui/Images/IO-Off.gif"))); // NOI18N
621     jLabelIndicatorChannel3.setOpaque(true);
622
623     jLabelOnChannel3.setBackground(new java.awt.Color(28, 28, 28));
624     jLabelOnChannel3.setForeground(new java.awt.Color(255, 255, 255));
625     jLabelOnChannel3.setText("On");
626
627     jLabelOffChannel3.setBackground(new java.awt.Color(28, 28, 28));
628     jLabelOffChannel3.setForeground(new java.awt.Color(255, 255, 255));
629     jLabelOffChannel3.setText("Off");
630
631     javax.swing.GroupLayout jPanelChannel3Layout = new javax.swing.GroupLayout(jPanelChannel3);
632     jPanelChannel3.setLayout(jPanelChannel3Layout);
633     jPanelChannel3Layout.setHorizontalGroup(
634         jPanelChannel3Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
635         .addComponent(jLabelChannel3Header, javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
636         .addComponent(jLabelChannel3Value, javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
637         .addGroup(jPanelChannel3Layout.createSequentialGroup()
638             .addContainerGap()
639             .addComponent(jToggleButtonChannel3, javax.swing.GroupLayout.PREFERRED_SIZE, 46, javax.swing.GroupLayout.PREFERRED_SIZE)
640             .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
641             .addComponent(jLabelIndicatorChannel3, javax.swing.GroupLayout.PREFERRED_SIZE, 54, javax.swing.GroupLayout.PREFERRED_SIZE)
642             .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
643             .addGroup(jPanelChannel3Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
644                 .addGroup(jPanelChannel3Layout.createSequentialGroup()
645                     .addComponent(jLabelOnChannel3, javax.swing.GroupLayout.DEFAULT_SIZE, 38, Short.MAX_VALUE)
646                     .addGap(14, 14, 14))
647                 .addGroup(jPanelChannel3Layout.createSequentialGroup()
648                     .addComponent(jLabelOffChannel3, javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALU
649                     .addContainerGap())))
650     );
651     jPanelChannel3Layout.setVerticalGroup(
652         jPanelChannel3Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
653         .addGroup(jPanelChannel3Layout.createSequentialGroup()
654             .addGroup(jPanelChannel3Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING)
655                 .addComponent(jToggleButtonChannel3, javax.swing.GroupLayout.PREFERRED_SIZE, 63, javax.swing.GroupLayout.PREFERRED_SIZE)
656                 .addGroup(jPanelChannel3Layout.createSequentialGroup()
657                     .addComponent(jLabelChannel3Header, javax.swing.GroupLayout.PREFERRED_SIZE, 28, javax.swing.GroupLayout.PREFERRED_SIZE)
658                     .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
659                     .addComponent(jLabelChannel3Value, javax.swing.GroupLayout.PREFERRED_SIZE, 60, javax.swing.GroupLayout.PREFERRED_SIZE)
660                     .addGroup(jPanelChannel3Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
661                         .addGroup(jPanelChannel3Layout.createSequentialGroup()
662                             .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
663                             .addComponent(jLabelIndicatorChannel3, javax.swing.GroupLayout.PREFERRED_SIZE, 63, javax.swing.GroupLayout.PREFERRED_SIZI
664                         .addGroup(jPanelChannel3Layout.createSequentialGroup()
665                             .addGap(19, 19, 19)
666                             .addComponent(jLabelOnChannel3, javax.swing.GroupLayout.PREFERRED_SIZE, 20, javax.swing.GroupLayout.PREFERRED_SIZE)
667                             .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
668                             .addComponent(jLabelOffChannel3, javax.swing.GroupLayout.PREFERRED_SIZE, 20, javax.swing.GroupLayout.PREFERRED_SIZE)))))
669             .addContainerGap(22, Short.MAX_VALUE))
670     );
671
672     jPanelChannel4.setBackground(new java.awt.Color(35, 39, 43));
673     jPanelChannel4.setPreferredSize(new java.awt.Dimension(170, 191));
674
675     jLabelChannel4Header.setBackground(new java.awt.Color(28, 28, 28));
676     jLabelChannel4Header.setFont(new java.awt.Font("Tahoma", 0, 14)); // NOI18N
677     jLabelChannel4Header.setForeground(new java.awt.Color(255, 255, 255));
678     jLabelChannel4Header.setText("Channel 4 (Input)");
679
680     jLabelChannel4Value.setBackground(new java.awt.Color(28, 28, 28));
681     jLabelChannel4Value.setFont(new java.awt.Font("Tahoma", 0, 14)); // NOI18N
682     jLabelChannel4Value.setForeground(new java.awt.Color(255, 255, 255));
683     jLabelChannel4Value.setText("jLabel2");
684
685     jToggleButtonChannel4.setBackground(new java.awt.Color(35, 39, 43));
686     jToggleButtonChannel4.setForeground(new java.awt.Color(35, 39, 43));
687     jToggleButtonChannel4.setIcon(new javax.swing.ImageIcon(getClass().getResource("/ntnusubsea/gui/Images/switch-off-vertical.gif"))); // NOI18N
688     jToggleButtonChannel4.setBorder(null);
689     jToggleButtonChannel4.setContentAreaFilled(false);
690     jToggleButtonChannel4.setEnabled(false);
691     jToggleButtonChannel4.setSelectedIcon(new javax.swing.ImageIcon(getClass().getResource("/ntnusubsea/gui/Images/switch-on-vertical.gif"))); // NOI18N
692     jToggleButtonChannel4.addActionListener(new java.awt.event.ActionListener()
693     {
```

```
694         public void actionPerformed(java.awt.event.ActionEvent evt)
695         {
696             jToggleButtonChannel4ActionPerformed(evt);
697         }
698     });
699 
700     jLabelIndicatorChannel4.setBackground(new java.awt.Color(35, 39, 43));
701     jLabelIndicatorChannel4.setIcon(new javax.swing.ImageIcon(getClass().getResource("/ntnusubsea/gui/Images/IO-Off.gif"))); // NOI18N
702     jLabelIndicatorChannel4.setOpaque(true);
703 
704     jLabelOnChannel4.setBackground(new java.awt.Color(28, 28, 28));
705     jLabelOnChannel4.setForeground(new java.awt.Color(255, 255, 255));
706     jLabelOnChannel4.setText("On");
707 
708     jLabelOffChannel4.setBackground(new java.awt.Color(28, 28, 28));
709     jLabelOffChannel4.setForeground(new java.awt.Color(255, 255, 255));
710     jLabelOffChannel4.setText("Off");
711 
712     javax.swing.GroupLayout jPanelChannel4Layout = new javax.swing.GroupLayout(jPanelChannel4);
713     jPanelChannel4.setLayout(jPanelChannel4Layout);
714     jPanelChannel4Layout.setHorizontalGroup(
715         jPanelChannel4Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
716         .addComponent(jLabelChannel4Header, javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
717         .addComponent(jLabelChannel4Value, javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
718         .addGroup(jPanelChannel4Layout.createSequentialGroup()
719             .addContainerGap()
720             .addComponent(jToggleButtonChannel4, javax.swing.GroupLayout.PREFERRED_SIZE, 46, javax.swing.GroupLayout.PREFERRED_SIZE)
721             .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
722             .addComponent(jLabelIndicatorChannel4, javax.swing.GroupLayout.PREFERRED_SIZE, 54, javax.swing.GroupLayout.PREFERRED_SIZE)
723             .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED, javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
724             .addGroup(jPanelChannel4Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
725                 .addComponent(jLabelOffChannel4, javax.swing.GroupLayout.PREFERRED_SIZE, 50, javax.swing.GroupLayout.PREFERRED_SIZE)
726                 .addComponent(jLabelOnChannel4, javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE
727             .addContainerGap())
728     );
729     jPanelChannel4Layout.setVerticalGroup(
730         jPanelChannel4Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
731         .addGroup(jPanelChannel4Layout.createSequentialGroup()
732             .addGroup(jPanelChannel4Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING)
733                 .addComponent(jToggleButtonChannel4, javax.swing.GroupLayout.PREFERRED_SIZE, 63, javax.swing.GroupLayout.PREFERRED_SIZE)
734                 .addGroup(jPanelChannel4Layout.createSequentialGroup()
735                     .addComponent(jLabelChannel4Header, javax.swing.GroupLayout.PREFERRED_SIZE, 28, javax.swing.GroupLayout.PREFERRED_SIZE)
736                     .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
737                     .addComponent(jLabelChannel4Value, javax.swing.GroupLayout.PREFERRED_SIZE, 60, javax.swing.GroupLayout.PREFERRED_SIZE)
738                     .addGroup(jPanelChannel4Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
739                         .addGroup(jPanelChannel4Layout.createSequentialGroup()
740                             .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
741                             .addComponent(jLabelIndicatorChannel4, javax.swing.GroupLayout.PREFERRED_SIZE, 63, javax.swing.GroupLayout.PREFERRED_SIZI
742                         .addGroup(jPanelChannel4Layout.createSequentialGroup()
743                             .addGap(19, 19, 19)
744                             .addComponent(jLabelOnChannel4, javax.swing.GroupLayout.PREFERRED_SIZE, 20, javax.swing.GroupLayout.PREFERRED_SIZE)
745                             .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
746                             .addComponent(jLabelOffChannel4, javax.swing.GroupLayout.PREFERRED_SIZE, 20, javax.swing.GroupLayout.PREFERRED_SIZE)))))
747             .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
748     );
749 
750     javax.swing.GroupLayout jPanel9Layout = new javax.swing.GroupLayout(jPanel9);
751     jPanel9.setLayout(jPanel9Layout);
752     jPanel9Layout.setHorizontalGroup(
753         jPanel9Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
754         .addGroup(jPanel9Layout.createSequentialGroup()
755             .addContainerGap()
756             .addGroup(jPanel9Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
757                 .addComponent(jPanelChannel5, javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
758                 .addComponent(jPanelChannel1, javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
759             .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
760             .addGroup(jPanel9Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
761                 .addGroup(jPanel9Layout.createSequentialGroup()
762                     .addComponent(jPanelChannel2, javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
763                     .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED, javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALU
764                     .addComponent(jPanelChannel3, javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
765                     .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED, javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALU
766                     .addComponent(jPanelChannel4, javax.swing.GroupLayout.DEFAULT_SIZE, 174, Short.MAX_VALUE))
767                 .addGroup(jPanel9Layout.createSequentialGroup()
768                     .addComponent(jPanelChannel6, javax.swing.GroupLayout.DEFAULT_SIZE, 171, Short.MAX_VALUE)
769                     .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
770                     .addComponent(jPanelChannel7, javax.swing.GroupLayout.DEFAULT_SIZE, 171, Short.MAX_VALUE)
771                     .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
772                     .addComponent(jPanelChannel8, javax.swing.GroupLayout.DEFAULT_SIZE, 172, Short.MAX_VALUE)))
773             .addContainerGap())
774     );
775     jPanel9Layout.setVerticalGroup(
776         jPanel9Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
777         .addGroup(jPanel9Layout.createSequentialGroup()
778             .addGap(10, 10, 10)
779             .addGroup(jPanel9Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING, false)
780                 .addComponent(jPanelChannel3, javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
```

```
781                 .addComponent(jPanelChannel2, javax.swing.GroupLayout.Alignment.LEADING, javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupL
782                 .addComponent(jPanelChannel1, javax.swing.GroupLayout.Alignment.LEADING, javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupL
783                 .addComponent(jPanelChannel4, javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
784             .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED, 14, Short.MAX_VALUE)
785             .addGroup(jPanel9Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING)
786                 .addComponent(jPanelChannel7, javax.swing.GroupLayout.Alignment.LEADING, javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupL
787                 .addComponent(jPanelChannel6, javax.swing.GroupLayout.Alignment.LEADING, javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupL
788                 .addComponent(jPanelChannel5, javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
789                 .addComponent(jPanelChannel8, javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
790             .addGap(15, 15, 15))
791     );
792 
793     javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());
794     getContentPane().setLayout(layout);
795     layout.setHorizontalGroup(
796         layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
797         .addComponent(jPanel9, javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
798     );
799     layout.setVerticalGroup(
800         layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
801         .addComponent(jPanel9, javax.swing.GroupLayout.PREFERRED_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFEl
802     );
803 
804     pack();
805 }// </editor-fold>
806 
807 private void jToggleButtonChannel5ActionPerformed(java.awt.event.ActionEvent evt) {
808     try {
809         if (jToggleButtonChannel5.isSelected()) {
810             setBit(4, 1);
811             //client.sendCommand(DIGITALOUTID + outputValue);
812             jLabelIndicatorChannel5.setIcon(new ImageIcon(ImageIO.read(new File("src/ntnusubsea/gui/Images/IO-On.gif"))));
813         } else {
814             setBit(4, 0);
815             //client.sendCommand(DIGITALOUTID + outputValue);
816             jLabelIndicatorChannel5.setIcon(new ImageIcon(ImageIO.read(new File("src/ntnusubsea/gui/Images/IO-Off.gif"))));
817         }
818         System.out.println(outputValue);
819     } catch (IOException ex) {
820         System.out.println(ex.getMessage());
821     }
822 }
823 
824 private void jToggleButtonChannel6ActionPerformed(java.awt.event.ActionEvent evt) {
825     try {
826         if (jToggleButtonChannel6.isSelected()) {
827             setBit(5, 1);
828             //client.sendCommand(DIGITALOUTID + outputValue);
829             jLabelIndicatorChannel6.setIcon(new ImageIcon(ImageIO.read(new File("src/ntnusubsea/gui/Images/IO-On.gif"))));
830         } else {
831             setBit(5, 0);
832             //client.sendCommand(DIGITALOUTID + outputValue);
833             jLabelIndicatorChannel6.setIcon(new ImageIcon(ImageIO.read(new File("src/ntnusubsea/gui/Images/IO-Off.gif"))));
834         }
835         System.out.println(outputValue);
836     } catch (IOException ex) {
837         System.out.println(ex.getMessage());
838     }
839 }
840 
841 private void jToggleButtonChannel7ActionPerformed(java.awt.event.ActionEvent evt) {
842     try {
843         if (jToggleButtonChannel7.isSelected()) {
844             setBit(6, 1);
845             //client.sendCommand(DIGITALOUTID + outputValue);
846             jLabelIndicatorChannel7.setIcon(new ImageIcon(ImageIO.read(new File("src/ntnusubsea/gui/Images/IO-On.gif"))));
847         } else {
848             setBit(6, 0);
849             //client.sendCommand(DIGITALOUTID + outputValue);
850             jLabelIndicatorChannel7.setIcon(new ImageIcon(ImageIO.read(new File("src/ntnusubsea/gui/Images/IO-Off.gif"))));
851         }
852         System.out.println(outputValue);
853     } catch (IOException ex) {
854         System.out.println(ex.getMessage());
855     }
856 }
857 
858 private void jToggleButtonChannel8ActionPerformed(java.awt.event.ActionEvent evt) {
859     try {
860         if (jToggleButtonChannel8.isSelected()) {
861             setBit(7, 1);
862             //client.sendCommand(DIGITALOUTID + outputValue);
863             jLabelIndicatorChannel8.setIcon(new ImageIcon(ImageIO.read(new File("src/ntnusubsea/gui/Images/IO-On.gif"))));
864         } else {
865             setBit(7, 0);
866             //client.sendCommand(DIGITALOUTID + outputValue);
867             jLabelIndicatorChannel8.setIcon(new ImageIcon(ImageIO.read(new File("src/ntnusubsea/gui/Images/IO-Off.gif"))));
```

```
868         }
869         System.out.println(outputValue);
870     } catch (IOException ex) {
871         System.out.println(ex.getMessage());
872     }
873 }
874
875 private void jToggleButtonChannel1ActionPerformed(java.awt.event.ActionEvent evt) {
876     try {
877         if (jToggleButtonChannel1.isSelected()) {
878             setBit(0, 1);
879             //client.sendCommand(DIGITALOUTID + outputValue);
880             jLabelIndicatorChannel1.setIcon(new ImageIcon(ImageIO.read(new File("src/ntnusubsea/gui/Images/IO-On.gif"))));
881         } else {
882             setBit(0, 0);
883             //client.sendCommand(DIGITALOUTID + outputValue);
884             jLabelIndicatorChannel1.setIcon(new ImageIcon(ImageIO.read(new File("src/ntnusubsea/gui/Images/IO-Off.gif"))));
885         }
886         System.out.println(outputValue);
887     } catch (IOException ex) {
888         System.out.println(ex.getMessage());
889     }
890 }
891
892 private void jToggleButtonChannel2ActionPerformed(java.awt.event.ActionEvent evt) {
893     try {
894         if (jToggleButtonChannel2.isSelected()) {
895             setBit(1, 1);
896             //client.sendCommand(DIGITALOUTID + outputValue);
897             jLabelIndicatorChannel2.setIcon(new ImageIcon(ImageIO.read(new File("src/ntnusubsea/gui/Images/IO-On.gif"))));
898         } else {
899             setBit(1, 0);
900             //client.sendCommand(DIGITALOUTID + outputValue);
901             jLabelIndicatorChannel2.setIcon(new ImageIcon(ImageIO.read(new File("src/ntnusubsea/gui/Images/IO-Off.gif"))));
902         }
903         System.out.println(outputValue);
904     } catch (IOException ex) {
905         System.out.println(ex.getMessage());
906     }
907 }
908
909 private void jToggleButtonChannel3ActionPerformed(java.awt.event.ActionEvent evt) {
910     try {
911         if (jToggleButtonChannel3.isSelected()) {
912             setBit(2, 1);
913             //client.sendCommand(DIGITALOUTID + outputValue);
914             jLabelIndicatorChannel3.setIcon(new ImageIcon(ImageIO.read(new File("src/ntnusubsea/gui/Images/IO-On.gif"))));
915         } else {
916             setBit(2, 0);
917             //client.sendCommand(DIGITALOUTID + outputValue);
918             jLabelIndicatorChannel3.setIcon(new ImageIcon(ImageIO.read(new File("src/ntnusubsea/gui/Images/IO-Off.gif"))));
919         }
920         System.out.println(outputValue);
921     } catch (IOException ex) {
922         System.out.println(ex.getMessage());
923     }
924 }
925
926 private void jToggleButtonChannel4ActionPerformed(java.awt.event.ActionEvent evt) {
927     try {
928         if (jToggleButtonChannel4.isSelected()) {
929             setBit(3, 1);
930             //client.sendCommand(DIGITALOUTID + outputValue);
931             jLabelIndicatorChannel4.setIcon(new ImageIcon(ImageIO.read(new File("src/ntnusubsea/gui/Images/IO-On.gif"))));
932         } else {
933             setBit(3, 0);
934             //client.sendCommand(DIGITALOUTID + outputValue);
935             jLabelIndicatorChannel4.setIcon(new ImageIcon(ImageIO.read(new File("src/ntnusubsea/gui/Images/IO-Off.gif"))));
936         }
937         System.out.println(outputValue);
938     } catch (IOException ex) {
939         System.out.println(ex.getMessage());
940     }
941 }
942
943 /**
944  * Enables the I/O buttons
945  */
946 public void enableIO() {
947     jToggleButtonChannel1.setEnabled(true);
948     jToggleButtonChannel2.setEnabled(true);
949     jToggleButtonChannel3.setEnabled(true);
950     jToggleButtonChannel4.setEnabled(true);
951     jToggleButtonChannel5.setEnabled(true);
952     jToggleButtonChannel6.setEnabled(true);
953     jToggleButtonChannel7.setEnabled(true);
954     jToggleButtonChannel8.setEnabled(true);
```

```
955     }
956
957     /**
958      * Disables the I/O buttons
959      */
960     public void disableIO() {
961         jToggleButtonChannel1.setEnabled(false);
962         jToggleButtonChannel2.setEnabled(false);
963         jToggleButtonChannel3.setEnabled(false);
964         jToggleButtonChannel4.setEnabled(false);
965         jToggleButtonChannel5.setEnabled(false);
966         jToggleButtonChannel6.setEnabled(false);
967         jToggleButtonChannel7.setEnabled(false);
968         jToggleButtonChannel8.setEnabled(false);
969     }
970
971     // Variables declaration - do not modify
972     private javax.swing.JLabel jLabelChannel1Header;
973     private javax.swing.JLabel jLabelChannel1Value;
974     private javax.swing.JLabel jLabelChannel2Header;
975     private javax.swing.JLabel jLabelChannel2Value;
976     private javax.swing.JLabel jLabelChannel3Header;
977     private javax.swing.JLabel jLabelChannel3Value;
978     private javax.swing.JLabel jLabelChannel4Header;
979     private javax.swing.JLabel jLabelChannel4Value;
980     private javax.swing.JLabel jLabelChannel5Header;
981     private javax.swing.JLabel jLabelChannel5Value;
982     private javax.swing.JLabel jLabelChannel6Header;
983     private javax.swing.JLabel jLabelChannel6Value;
984     private javax.swing.JLabel jLabelChannel7Header;
985     private javax.swing.JLabel jLabelChannel7Value;
986     private javax.swing.JLabel jLabelChannel8Header;
987     private javax.swing.JLabel jLabelChannel8Value;
988     private javax.swing.JLabel jLabelIndicatorChannel1;
989     private javax.swing.JLabel jLabelIndicatorChannel2;
990     private javax.swing.JLabel jLabelIndicatorChannel3;
991     private javax.swing.JLabel jLabelIndicatorChannel4;
992     private javax.swing.JLabel jLabelIndicatorChannel5;
993     private javax.swing.JLabel jLabelIndicatorChannel6;
994     private javax.swing.JLabel jLabelIndicatorChannel7;
995     private javax.swing.JLabel jLabelIndicatorChannel8;
996     private javax.swing.JLabel jLabelOffChannel1;
997     private javax.swing.JLabel jLabelOffChannel2;
998     private javax.swing.JLabel jLabelOffChannel3;
999     private javax.swing.JLabel jLabelOffChannel4;
1000    private javax.swing.JLabel jLabelOffChannel5;
1001    private javax.swing.JLabel jLabelOffChannel6;
1002    private javax.swing.JLabel jLabelOffChannel7;
1003    private javax.swing.JLabel jLabelOffChannel8;
1004    private javax.swing.JLabel jLabelOnChannel1;
1005    private javax.swing.JLabel jLabelOnChannel2;
1006    private javax.swing.JLabel jLabelOnChannel3;
1007    private javax.swing.JLabel jLabelOnChannel4;
1008    private javax.swing.JLabel jLabelOnChannel5;
1009    private javax.swing.JLabel jLabelOnChannel6;
1010    private javax.swing.JLabel jLabelOnChannel7;
1011    private javax.swing.JLabel jLabelOnChannel8;
1012    private javax.swing.JPanel jPanel9;
1013    private javax.swing.JPanel jPanelChannel1;
1014    private javax.swing.JPanel jPanelChannel2;
1015    private javax.swing.JPanel jPanelChannel3;
1016    private javax.swing.JPanel jPanelChannel4;
1017    private javax.swing.JPanel jPanelChannel5;
1018    private javax.swing.JPanel jPanelChannel6;
1019    private javax.swing.JPanel jPanelChannel7;
1020    private javax.swing.JPanel jPanelChannel8;
1021    private javax.swing.JToggleButton jToggleButtonChannel1;
1022    private javax.swing.JToggleButton jToggleButtonChannel2;
1023    private javax.swing.JToggleButton jToggleButtonChannel3;
1024    private javax.swing.JToggleButton jToggleButtonChannel4;
1025    private javax.swing.JToggleButton jToggleButtonChannel5;
1026    private javax.swing.JToggleButton jToggleButtonChannel6;
1027    private javax.swing.JToggleButton jToggleButtonChannel7;
1028    private javax.swing.JToggleButton jToggleButtonChannel8;
1029    // End of variables declaration
1030
1031    /**
1032     * Runs the IOControlFrame thread.
1033     */
1034    @Override
1035    public void run() {
1036        this.setVisible(false);
1037    }
1038
1039    /**
1040     * Updates the channel values by observing the object
1041     *
```

```
1042     * @param o
1043     * @param arg
1044     */
1045    @Override
1046    public void update(Observable o, Object arg) {
1047        jLabelChannel1Value.setText(data.getChannel(1));
1048        jLabelChannel2Value.setText(data.getChannel(2));
1049        jLabelChannel3Value.setText(data.getChannel(3));
1050        jLabelChannel4Value.setText(data.getChannel(4));
1051    }
1052
1053    /**
1054     * Sets the given value to the given bit
1055     *
1056     * @param bit the given bit
1057     * @param value the given value
1058     */
1059    private void setBit(int bit, int value) {
1060        if (value == 0) {
1061            outputValue = outputValue & ~(1 << bit);
1062        } else {
1063            outputValue = outputValue | (1 << bit);
1064        }
1065    }
1066 }
1067
```

D:\Dokumenter\Skule\04 - NTNU\Bachelor\Github\TowedROV_GUI\src\inputcontroller\LeftThumbXListener.java

```java
/*
 * This code is for the bachelor thesis named "Towed-ROV".
 * The purpose is to build a ROV which will be towed behind a surface vessel
 * and act as a multi-sensor platform, were it shall be easy to place new
 * sensors. There will also be a video stream from the ROV.
 *
 * The system consists of two Raspberry Pis in the ROV that is connected to
 * several Arduino micro controllers. These micro controllers are connected to
 * feedback from the actuators, the echo sounder and extra optional sensors.
 * The external computer which is on the surface vessel is connected to a GPS,
 * echo sounder over USB, and the ROV over ethernet. It will present and
 * log data in addition to handle user commands for controlling the ROV.
 */
package InputController;

import com.exlumina.j360.ValueListener;

/**
 * This class is responsible for handling the input from the X axis of the left
 * stick on the Xbox 360 controller.
 */
class LeftThumbXListener implements ValueListener {

    private InputController ic;

    /**
     * The constructor of the LeftThumbXListener class
     *
     * @param ic the InputController
     */
    public LeftThumbXListener(InputController ic) {
        this.ic = ic;
    }

    /**
     * Sets the new value from the Xbox controller
     *
     * @param newValue the new value
     */
    @Override
    public void value(int newValue) {
        newValue = map(newValue, -32768, 32768, -100, 100);
        this.ic.setBtnLx(newValue);
        //System.out.printf("Lx: " + "%6d\n", newValue);
    }

    /**
     * Maps the given value range to the given output range
     *
     * @param x the value to be mapped
     * @param in_min the minimum value of the input range
     * @param in_max the maximum value of the input range
     * @param out_min the minimum value of the output range
     * @param out_max the maximum value of the output range
```

```
55     * @return the mapped value
56     */
57    private int map(int x, int in_min, int in_max, int out_min, int out_max) {
58        return (x - in_min) * (out_max - out_min) / (in_max - in_min) + out_min;
59    }
60
61 }
62
```

D:\Dokumenter\Skule\04 - NTNU\Bachelor\Github\TowedROV_GUI\src\inputcontroller\LeftThumbYListener.java

```java
/*
 * This code is for the bachelor thesis named "Towed-ROV".
 * The purpose is to build a ROV which will be towed behind a surface vessel
 * and act as a multi-sensor platform, were it shall be easy to place new
 * sensors. There will also be a video stream from the ROV.
 *
 * The system consists of two Raspberry Pis in the ROV that is connected to
 * several Arduino micro controllers. These micro controllers are connected to
 * feedback from the actuators, the echo sounder and extra optional sensors.
 * The external computer which is on the surface vessel is connected to a GPS,
 * echo sounder over USB, and the ROV over ethernet. It will present and
 * log data in addition to handle user commands for controlling the ROV.
 */
package InputController;

import com.exlumina.j360.ValueListener;

/**
 * This class is responsible for handling the input from the Y axis of the left
 * stick on the Xbox 360 controller.
 */
class LeftThumbYListener implements ValueListener {

    private InputController ic;

    /**
     * The constructor of the LeftThumbYListener class
     *
     * @param ic the InputController
     */
    public LeftThumbYListener(InputController ic) {
        this.ic = ic;
    }

    /**
     * Sets the new value from the Xbox controller
     *
     * @param newValue the new value
     */
    @Override
    public void value(int newValue) {
        newValue = map(newValue, -32768, 32768, 254, 1);
        if (newValue > 115 && newValue < 139) {
            newValue = 127;
        }
        this.ic.setBtnLy(newValue);
        System.out.printf("Ly: " + "%6d\n", newValue);
    }

    /**
     * Maps the given value range to the given output range
     *
     * @param x the value to be mapped
     * @param in_min the minimum value of the input range
```

```
55     * @param in_max the maximum value of the input range
56     * @param out_min the minimum value of the output range
57     * @param out_max the maximum value of the output range
58     * @return the mapped value
59     */
60    private int map(int x, int in_min, int in_max, int out_min, int out_max) {
61        return (x - in_min) * (out_max - out_min) / (in_max - in_min) + out_min;
62    }
63 }
64
```

D:\Dokumenter\Skule\04 - NTNU\Bachelor\Github\TowedROV_GUI\src\basestation_rov\LogFileHandler.java

```
1  /*
2   * This code is for the bachelor thesis named "Towed-ROV".
3   * The purpose is to build a ROV which will be towed behind a surface vessel
4   * and act as a multi-sensor platform, were it shall be easy to place new
5   * sensors. There will also be a video stream from the ROV.
6   *
7   * The system consists of two Raspberry Pis in the ROV that is connected to
8   * several Arduino micro controllers. These micro controllers are connected to
9   * feedback from the actuators, the echo sounder and extra optional sensors.
10  * The external computer which is on the surface vessel is connected to a GPS,
11  * echo sounder over USB, and the ROV over ethernet. It will present and
12  * log data in addition to handle user commands for controlling the ROV.
13  */
14 package basestation_rov;
15
16 import java.io.BufferedWriter;
17 import java.io.File;
18 import java.io.FileWriter;
19 import java.text.SimpleDateFormat;
20 import java.util.Date;
21 import ntnusubsea.gui.Data;
22 import org.apache.commons.io.FileUtils;
23
24 /**
25  * This class is responsible for logging the data, ship position, exif and
26  * telementry to seperate .csv files.
27  */
28 public class LogFileHandler implements Runnable {
29
30     //User settings
31     int pointFreqMillis = 5000;
32     int lenghtOfUmbillicalCord = 500;
33     //End of user settings
34
35     Data data;
36
37     long lastTime = 0;
38     long timeDifference = 0;
39
40     double adjustedCoordinateRovOffset = ((lenghtOfUmbillicalCord / 100) * 0.000892);
41
42     String shipTrack = "null";
43     String Data = "null";
44     String telementry = "null";
45     String photoLocationTrack = "null";
46     String exif = "null";
47
48     int shipTrackPointNumb = 1;
49     int DataPointNumb = 1;
50     int photoLocationNumb = 1;
51
52     String timeStampString = "";
53     SimpleDateFormat timeAndDateCSV;
54     SimpleDateFormat exifDateAndTime;
55
56     String logStorageLocation = "C:\\TowedROV\\Log\\";
57
58     String photoPosLog = "";
59     String shipPosLog = "";
60     String dataLog = "";
61     String telementryLog = "";
62     String exifLog = "";
63     boolean setupIsDone = false;
```

```
64
65  File shipPosLogFile = null;
66  File dataLogFile = null;
67  File telementryLogFile = null;
68  File exifLogFile = null;
69
70  Date date;
71  Date dateCSV;
72  Date dateExif;
73
74  BufferedWriter outputWriterShipPos = null;
75  BufferedWriter outputWriterData = null;
76  BufferedWriter outputWriterTelementry = null;
77  BufferedWriter outputWriterExif = null;
78
79  int lastImageNumber = 0;
80  boolean exifSetup = false;
81
82  /**
83   * The constructor of the LogFileHandler class
84   *
85   * @param data the shared resource Data class
86   */
87  public LogFileHandler(Data data) {
88      this.data = data;
89  }
90
91  /**
92   * Runs the LogFileHandler thread.
93   */
94  public void run() {
95      if (!exifSetup || data.isImagesCleared()) {
96          try {
97              lastImageNumber = 0;
98              //SimpleDateFormat exifTime = new SimpleDateFormat("yyyyMMddHHmmss");
99              exifDateAndTime = new SimpleDateFormat("yyyyMMddHHmmss");
100             dateExif = new Date(System.currentTimeMillis());
101             exifLogFile = new File(logStorageLocation + "EXIF_LOG_" + exifDateAndTime.format(dateExif) + ".csv");
102             FileUtils.touch(exifLogFile);
103
104             outputWriterExif = new BufferedWriter(new FileWriter(exifLogFile));
105             outputWriterExif.append("Latitude,Longtitude,Time");
106             outputWriterExif.flush();
107             exifSetup = true;
108             data.setImagesCleared(true);
109
110         } catch (Exception e) {
111         }
112
113     }
114     if (data.getImageNumber() != lastImageNumber) {
115         exifDateAndTime = new SimpleDateFormat("yyyyMMddHHmmss");
116         dateExif = new Date(System.currentTimeMillis());
117         exifDateAndTime.format(dateExif);
118         lastImageNumber++;
119         logExifData();
120
121     }
122
123     if (data.startLogging) {
124
125         if (!setupIsDone) {
126             try {
127                 SimpleDateFormat formatter = new SimpleDateFormat("yyyy-MM-dd_HHmmss");
128                 date = new Date(System.currentTimeMillis());
129
```

```
130                 shipPosLogFile = new File(logStorageLocation + "ShipPos_LOG_" + formatter.format(date) + ".csv");
131                 FileUtils.touch(shipPosLogFile);
132 
133 //              File photoPosLog = new File(logStorageLocation + "PhotoPos_LOG_" + formatter.format(date) + ".csv");
134 //              FileUtils.touch(photoPosLog);
135                 dataLogFile = new File(logStorageLocation + "Data_LOG_" + formatter.format(date) + ".csv");
136                 FileUtils.touch(dataLogFile);
137 
138                 telementryLogFile = new File(logStorageLocation + "Telementry_LOG_" + formatter.format(date) + ".csv");
139                 FileUtils.touch(telementryLogFile);
140 
141                 outputWriterShipPos = new BufferedWriter(new FileWriter(shipPosLogFile));
142 
143                 outputWriterData = new BufferedWriter(new FileWriter(dataLogFile));
144 
145                 outputWriterTelementry = new BufferedWriter(new FileWriter(telementryLogFile));
146 
147                 outputWriterShipPos.append("Point,Time,Latitude,Longtitude,Speed,ROV Depth,GPSHeading");
148                 outputWriterShipPos.flush();
149 
150                 outputWriterData.append("Point,Time,Roll,Pitch,Depth,"
151                         + "DepthToSeaFloor,ROV_Depth,ActuatorPS_feedback,"
152                         + "ActuatorSB_feedback,ActuatorPS_command,"
153                         + "ActuatorSB_command,Voltage,Emergency, outsideTemp,"
154                         + "insideTempCameraHouse, humidity, tempElBoxFromt,"
155                         + "tempElBoxRear, I2CError, LeakDetection");
156                 outputWriterData.flush();
157 
158                 outputWriterTelementry.append("Latitude,Longtitude, Elevation, Time");
159 //                  + "Elevation,Heading,Time");
160                 outputWriterTelementry.flush();
161 
162                 setupIsDone = true;
163 
164             } catch (Exception ex) {
165                 System.out.println("ERROR: " + ex);
166             }
167         }
168 
169         dateCSV = new Date(System.currentTimeMillis());
170         timeStampString = String.valueOf(java.time.LocalTime.now());
171         timeAndDateCSV = new SimpleDateFormat("yyyy-MM-dd'T'HH:mm:ss.SSS");
172 
173         logShipPosition();
174         logData();
175         logTelementry();
176 
177     } else {
178         setupIsDone = false;
179     }
180 }
181 
182 //  private void logPhotoPosition(BufferedWriter bw)
183 //  {
184 //      try
185 //      {
186 //          double shipsLatitude = data.getLatitude();
187 //          double shipsLontitude = data.getLongitude();
188 //          double rovHeadingRelToShip = data.getHeading() - 180;
189 //          if (rovHeadingRelToShip < 0)
190 //          {
191 //              rovHeadingRelToShip = rovHeadingRelToShip + 360;
192 //          }
193 //          //convert to radians
194 //
195 //          double normalizedHeading = Math.atan2(Math.sin(rovHeadingRelToShip), Math.cos(rovHeadingRelToShip));
```

```
196 //          double rovHeadingRelToShipSin = sin(normalizedHeading * PI / 180);
197 //          double rovHeadingRelToShipSinRest = 0;
198 //
199 //          if (rovHeadingRelToShipSin < 0)
200 //          {
201 //             rovHeadingRelToShipSinRest = 1 + rovHeadingRelToShipSin;
202 //
203 //          }
204 //
205 //          double rovLatitude = 0;
206 //          double rovLongtitude = 0;
207 //
208 //          rovLatitude = data.getLatitude() - (adjustedCoordinateRovOffset * rovHeadingRelToShipSinRest);
209 //          if (rovHeadingRelToShipSinRest != 0)
210 //          {
211 //             rovLongtitude = data.getLongitude() + (adjustedCoordinateRovOffset * rovHeadingRelToShipSin);
212 //          } else
213 //          {
214 //             rovLongtitude = data.getLongitude();
215 //          }
216 //
217 //          photoLocationTrack = "";
218 //          photoLocationTrack = photoLocationNumb + ","
219 //                + data.getLatitude() + "," + data.getLongitude() + ","
220 //                + data.getSpeed() + "," + data.getRovDepth();
221 //
222 //          bw.append(photoLocationTrack);
223 //          bw.append('\n');
224 //          bw.flush();
225 //
226 //          photoLocationNumb++;
227 //       } catch (Exception e)
228 //       {
229 //          System.out.println("Error: " + e);
230 //       }
231 //
232 //    }
233     /**
234      * Closes the BufferedWriter for each log file.
235      */
236     public void closeLog() {
237        try {
238           outputWriterShipPos.close();
239           outputWriterData.close();
240           outputWriterExif.close();
241           outputWriterTelementry.close();
242        } catch (Exception e) {
243           System.out.println("Problem closing log file");
244        }
245
246     }
247
248     /**
249      * Logs the exif data to file
250      */
251     private void logExifData() {
252        try {
253           exifLog = "";
254           exifLog = data.getLatitude() + ","
255                 + data.getLongitude();
256
257           outputWriterExif.append('\n');
258           outputWriterExif.append(telementryLog);
259           outputWriterExif.flush();
260        } catch (Exception e) {
261        }
```

```
262
263     }
264
265     /**
266      * Logs the telementry data to file
267      */
268     private void logTelementry() {
269         try {
270             telementryLog = "";
271             telementryLog = data.getLatitude() + ","
272                     + data.getLongitude() + ","
273                     + data.getDepth() + ","
274                     + timeAndDateCSV.format(dateCSV);
275
276             outputWriterTelementry.append('\n');
277             outputWriterTelementry.append(telementryLog);
278             outputWriterTelementry.flush();
279
280         } catch (Exception e) {
281             System.out.println("Error writing telementry...");
282         }
283
284     }
285
286     /**
287      * Logs the data to file
288      */
289     private void logData() {
290         try {
291             dataLog = "";
292
293             dataLog = DataPointNumb + ","
294                     + timeStampString + ","
295                     + String.valueOf(data.getRollAngle()) + ","
296                     + String.valueOf(data.getPitchAngle()) + ","
297                     + String.valueOf(data.getDepthBeneathBoat()) + ","
298                     + String.valueOf(data.getDepthBeneathRov()) + ","
299                     + String.valueOf(data.getRovDepth()) + ","
300                     + String.valueOf(data.getFb_actuatorPSPos()) + ","
301                     + String.valueOf(data.getFb_actuatorSBPos()) + ","
302                     + String.valueOf(data.getFb_actuatorPScmd()) + ","
303                     + String.valueOf(data.getFb_actuatorSBcmd()) + ","
304                     + String.valueOf(data.getVoltage()) + ","
305                     + String.valueOf(data.getOutsideTemp()) + ","
306                     + String.valueOf(data.getInsideTemp()) + ","
307                     + String.valueOf(data.getHumidity()) + ","
308                     + String.valueOf(data.getFb_tempElBoxFront()) + ","
309                     + String.valueOf(data.getFb_tempElBoxRear()) + ","
310                     + String.valueOf(data.isI2cError()) + ","
311                     + String.valueOf(data.getLeakStatus()) + ",";
312
313 //          outputWriterData.append(String.valueOf(DataPointNumb));
314 //          outputWriterData.append(',');
315 //          outputWriterData.append(String.valueOf(timeStampString));
316 //          outputWriterData.append(',');
317 //          outputWriterData.append(String.valueOf(data.getRollAngle()));
318 //          outputWriterData.append(',');
319 //          outputWriterData.append(String.valueOf(data.getPitchAngle()));
320 //          outputWriterData.append(',');
321 //          outputWriterData.append(String.valueOf(data.getDepth()));
322 //          outputWriterData.append(',');
323 //          outputWriterData.append(String.valueOf(data.getDepthBeneathRov()));
324 //          outputWriterData.append(',');
325 //          outputWriterData.append(String.valueOf(data.getRovDepth()));
326 //          outputWriterData.append(',');
327 //          outputWriterData.append(String.valueOf(data.getFb_actuatorPSPos()));
```

```
328 //          outputWriterData.append(',');
329 //          outputWriterData.append(String.valueOf(data.getFb_actuatorSBPos()));
330 //          outputWriterData.append(',');
331 //          outputWriterData.append(String.valueOf(data.getFb_actuatorPScmd()));
332 //          outputWriterData.append(',');
333 //          outputWriterData.append(String.valueOf(data.getFb_actuatorSBcmd()));
334 //          outputWriterData.append(',');
335 //          outputWriterData.append(String.valueOf(data.getVoltage()));
336           outputWriterData.append('\n');
337           outputWriterData.append(dataLog);
338           outputWriterData.flush();
339           DataPointNumb++;
340       } catch (Exception e) {
341       }
342   }
343
344   /**
345    * Logs the error data to file. Not finished
346    */
347   private void logErrorMessages() {
348
349   }
350
351   /**
352    * Logs the ship position data to file
353    */
354   private void logShipPosition() {
355       try {
356           shipTrack = "";
357           shipTrack = shipTrackPointNumb + "," + timeStampString + ","
358                   + data.getLatitude() + "," + data.getLongitude() + ","
359                   + data.getSpeed() + "," + data.getRovDepth() + "," + data.getGPSAngle();
360           outputWriterShipPos.append('\n');
361           outputWriterShipPos.append(shipTrack);
362           outputWriterShipPos.flush();
363           shipTrackPointNumb++;
364
365       } catch (Exception e) {
366           System.out.println("Error: " + e);
367       }
368   }
369 }
370
```

D:\Dokumenter\Skule\04 - NTNU\Bachelor\Github\TowedROV_GUI\src\ntnusubsea\gui\OptionsFrame.java

```
1 /*
2  * This code is for the bachelor thesis named "Towed-ROV".
3  * The purpose is to build a ROV which will be towed behind a surface vessel
4  * and act as a multi-sensor platform, were it shall be easy to place new
5  * sensors. There will also be a video stream from the ROV.
6  *
7  * The system consists of two Raspberry Pis in the ROV that is connected to
8  * several Arduino micro controllers. These micro controllers are connected to
9  * feedback from the actuators, the echo sounder and extra optional sensors.
10 * The external computer which is on the surface vessel is connected to a GPS,
11 * echo sounder over USB, and the ROV over ethernet. It will present and
12 * log data in addition to handle user commands for controlling the ROV.
13 */
14 package ntnusubsea.gui;
15
16 import java.io.BufferedReader;
17 import java.io.BufferedWriter;
18 import java.io.File;
19 import java.io.FileReader;
20 import java.io.FileWriter;
21 import java.io.IOException;
22 import java.io.InputStreamReader;
23 import java.util.ArrayList;
24 import java.util.logging.Level;
25 import java.util.logging.Logger;
26 import javax.swing.JFrame;
27 import org.apache.commons.io.FileUtils;
28
29 /**
30  * This class allows the user to change options between every time the program
31  * is loaded. It reads and writes from the ROV Options text file from the
32  * program folder
33  */
34 public class OptionsFrame extends javax.swing.JFrame {
35
36     File file = null;
37     String IP_Rov;
38     String IP_Camera;
39     ArrayList channels = new ArrayList<String>();
40     private Data data;
41     private TCPClient client_ROV;
42     private double Kp;
43     private double Ki;
44     private double Kd;
45
46     /**
47      * Creates new form OptionsFrame
48      *
49      * @param data Data containing shared variables.
50      * @param client_ROV
51      */
52     public OptionsFrame(Data data, TCPClient client_ROV) {
53         initComponents();
54         try {
55             this.file = new File("ROV Options.txt");
56             FileUtils.touch(file);
57         } catch (IOException ex) {
58             Logger.getLogger(OptionsFrame.class.getName()).log(Level.SEVERE, null, ex);
59         }
60         //setSize(640, 480);
61         this.pack();
62         this.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
63         this.setLocationRelativeTo(null);
64         this.data = data;
65         this.client_ROV = client_ROV;
66         getOptionsFromFile();
67     }
68
69     /**
70      * This method is called from within the constructor to initialize the form.
71      * WARNING: Do NOT modify this code. The content of this method is always
72      * regenerated by the Form Editor.
73      */
74     @SuppressWarnings("unchecked")
75     // <editor-fold defaultstate="collapsed" desc="Generated Code">
76     private void initComponents() {
77
78         jLabelIPHeader = new javax.swing.JLabel();
79         jButtonOK = new javax.swing.JButton();
80         jButtonApply = new javax.swing.JButton();
81         jButtonCanel = new javax.swing.JButton();
82         jTextFieldIP_Rov = new javax.swing.JTextField();
83         jSeparator1 = new javax.swing.JSeparator();
84         jLabelChannelsHeader = new javax.swing.JLabel();
```

```
85      jLabel1 = new javax.swing.JLabel();
86      jLabel2 = new javax.swing.JLabel();
87      jLabel3 = new javax.swing.JLabel();
88      jLabel4 = new javax.swing.JLabel();
89      jLabel5 = new javax.swing.JLabel();
90      jLabel6 = new javax.swing.JLabel();
91      jLabel7 = new javax.swing.JLabel();
92      jLabel8 = new javax.swing.JLabel();
93      jTextFieldChannel1 = new javax.swing.JTextField();
94      jTextFieldChannel2 = new javax.swing.JTextField();
95      jTextFieldChannel4 = new javax.swing.JTextField();
96      jTextFieldChannel3 = new javax.swing.JTextField();
97      jTextFieldChannel5 = new javax.swing.JTextField();
98      jTextFieldChannel6 = new javax.swing.JTextField();
99      jTextFieldChannel7 = new javax.swing.JTextField();
100     jTextFieldChannel8 = new javax.swing.JTextField();
101     jTextFieldIP_Camera = new javax.swing.JTextField();
102     jLabelIPHeader1 = new javax.swing.JLabel();
103     jLabel9 = new javax.swing.JLabel();
104     KpTextField = new javax.swing.JTextField();
105     jLabel10 = new javax.swing.JLabel();
106     jLabel11 = new javax.swing.JLabel();
107     KiTextField = new javax.swing.JTextField();
108     jLabel12 = new javax.swing.JLabel();
109     KdTextField = new javax.swing.JTextField();
110     jLabel13 = new javax.swing.JLabel();
111     jLabel14 = new javax.swing.JLabel();
112     offset1TextField = new javax.swing.JTextField();
113     jLabel15 = new javax.swing.JLabel();
114     offset2TextField = new javax.swing.JTextField();
115
116     setDefaultCloseOperation(javax.swing.WindowConstants.DISPOSE_ON_CLOSE);
117     setTitle("Options");
118     setBackground(new java.awt.Color(39, 44, 50));
119     setForeground(new java.awt.Color(39, 44, 50));
120     setMaximumSize(new java.awt.Dimension(432, 333));
121     setMinimumSize(new java.awt.Dimension(432, 333));
122     setResizable(false);
123
124     jLabelIPHeader.setFont(new java.awt.Font("Tahoma", 0, 14)); // NOI18N
125     jLabelIPHeader.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
126     jLabelIPHeader.setText("ROV IP");
127
128     jButtonOK.setText("OK");
129     jButtonOK.setPreferredSize(new java.awt.Dimension(65, 25));
130     jButtonOK.addActionListener(new java.awt.event.ActionListener() {
131         public void actionPerformed(java.awt.event.ActionEvent evt) {
132             jButtonOKActionPerformed(evt);
133         }
134     });
135
136     jButtonApply.setText("Apply");
137     jButtonApply.setPreferredSize(new java.awt.Dimension(65, 25));
138     jButtonApply.addActionListener(new java.awt.event.ActionListener() {
139         public void actionPerformed(java.awt.event.ActionEvent evt) {
140             jButtonApplyActionPerformed(evt);
141         }
142     });
143
144     jButtonCanel.setText("Cancel");
145     jButtonCanel.setPreferredSize(new java.awt.Dimension(65, 25));
146     jButtonCanel.addActionListener(new java.awt.event.ActionListener() {
147         public void actionPerformed(java.awt.event.ActionEvent evt) {
148             jButtonCanelActionPerformed(evt);
149         }
150     });
151
152     jTextFieldIP_Rov.setText("123.123.123.123");
153     jTextFieldIP_Rov.setToolTipText("");
154
155     jLabelChannelsHeader.setFont(new java.awt.Font("Tahoma", 0, 14)); // NOI18N
156     jLabelChannelsHeader.setText("Custom I/O");
157
158     jLabel1.setText("<html>Channel 1<br/>Input");
159
160     jLabel2.setText("<html>Channel 2<br/>Input");
161
162     jLabel3.setText("<html>Channel 5<br/>Output");
163
164     jLabel4.setText("<html>Channel 3<br/>Input");
165
166     jLabel5.setText("<html>Channel 4<br/>Input");
167
168     jLabel6.setText("<html>Channel 7<br/>Output");
169
170     jLabel7.setText("<html>Channel 6<br/>Output");
171
```

```
172     jLabel8.setText("<html>Channel 8<br/>Output");
173
174     jTextFieldChannel1.setText("Channel 1");
175
176     jTextFieldChannel2.setText("Channel 2");
177
178     jTextFieldChannel4.setText("Channel 4");
179
180     jTextFieldChannel3.setText("Channel 3");
181
182     jTextFieldChannel5.setText("Channel 5");
183     jTextFieldChannel5.addActionListener(new java.awt.event.ActionListener() {
184         public void actionPerformed(java.awt.event.ActionEvent evt) {
185             jTextFieldChannel5ActionPerformed(evt);
186         }
187     });
188
189     jTextFieldChannel6.setText("Channel 6");
190     jTextFieldChannel6.addActionListener(new java.awt.event.ActionListener() {
191         public void actionPerformed(java.awt.event.ActionEvent evt) {
192             jTextFieldChannel6ActionPerformed(evt);
193         }
194     });
195
196     jTextFieldChannel7.setText("Channel 7");
197
198     jTextFieldChannel8.setText("Channel 8");
199
200     jTextFieldIP_Camera.setText("123.123.123.123");
201     jTextFieldIP_Camera.setToolTipText("");
202
203     jLabelIPHeader1.setFont(new java.awt.Font("Tahoma", 0, 14)); // NOI18N
204     jLabelIPHeader1.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
205     jLabelIPHeader1.setText("Camera IP");
206
207     jLabel9.setFont(new java.awt.Font("Tahoma", 0, 14)); // NOI18N
208     jLabel9.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
209     jLabel9.setText("PID: ");
210
211     KpTextField.setText("Kp");
212
213     jLabel10.setText("Kp");
214
215     jLabel11.setText("Ki");
216
217     KiTextField.setText("Ki");
218
219     jLabel12.setText("Kd");
220
221     KdTextField.setText("Kd");
222
223     jLabel13.setFont(new java.awt.Font("Tahoma", 0, 14)); // NOI18N
224     jLabel13.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
225     jLabel13.setText("Offset:");
226
227     jLabel14.setText("Depth beneath ROV");
228
229     offset1TextField.setText("0.00");
230
231     jLabel15.setText("ROV Depth");
232
233     offset2TextField.setText("0.00");
234
235     javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());
236     getContentPane().setLayout(layout);
237     layout.setHorizontalGroup(
238         layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
239         .addGroup(layout.createSequentialGroup()
240             .addContainerGap()
241             .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING)
242                 .addGroup(layout.createSequentialGroup()
243                     .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
244                         .addComponent(jSeparator1)
245                         .addGroup(layout.createSequentialGroup()
246                             .addComponent(jTextFieldChannel5, javax.swing.GroupLayout.PREFERRED_SIZE, 100, javax.swing.GroupLayout.PREFERRED_SIZE)
247                             .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
248                             .addComponent(jTextFieldChannel6)
249                             .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
250                             .addComponent(jTextFieldChannel7)
251                             .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
252                             .addComponent(jTextFieldChannel8))
253                         .addGroup(javax.swing.GroupLayout.Alignment.TRAILING, layout.createSequentialGroup()
254                             .addGap(0, 0, Short.MAX_VALUE)
255                             .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
256                                 .addGroup(javax.swing.GroupLayout.Alignment.TRAILING, layout.createSequentialGroup()
257                                     .addComponent(jButtonOK, javax.swing.GroupLayout.PREFERRED_SIZE, 75, javax.swing.GroupLayout.PREFERRED_SIZE)
258                                     .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
```

```
                        .addComponent(jButtonApply, javax.swing.GroupLayout.PREFERRED_SIZE, 75, javax.swing.GroupLayout.PREFERRED_SIZE)
                        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                        .addComponent(jButtonCanel, javax.swing.GroupLayout.PREFERRED_SIZE, 75, javax.swing.GroupLayout.PREFERRED_SIZE))
                    .addGroup(javax.swing.GroupLayout.Alignment.TRAILING, layout.createSequentialGroup()
                        .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING, false)
                            .addComponent(jLabel1, javax.swing.GroupLayout.DEFAULT_SIZE, 100, Short.MAX_VALUE)
                            .addComponent(jLabel3)
                            .addComponent(jTextFieldChannel1))
                        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                        .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING, false)
                            .addComponent(jTextFieldChannel2, javax.swing.GroupLayout.DEFAULT_SIZE, 100, Short.MAX_VALUE)
                            .addComponent(jLabel7, javax.swing.GroupLayout.Alignment.TRAILING)
                            .addComponent(jLabel2, javax.swing.GroupLayout.Alignment.TRAILING))
                        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                        .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING, false)
                            .addComponent(jLabel6, javax.swing.GroupLayout.DEFAULT_SIZE, 100, Short.MAX_VALUE)
                            .addComponent(jLabel4)
                            .addComponent(jTextFieldChannel3))
                        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                        .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING, false)
                            .addComponent(jTextFieldChannel4, javax.swing.GroupLayout.DEFAULT_SIZE, 100, Short.MAX_VALUE)
                            .addComponent(jLabel8, javax.swing.GroupLayout.Alignment.TRAILING)
                            .addComponent(jLabel5, javax.swing.GroupLayout.Alignment.TRAILING))))))
                .addContainerGap())
            .addGroup(javax.swing.GroupLayout.Alignment.LEADING, layout.createSequentialGroup()
                .addComponent(jLabelChannelsHeader, javax.swing.GroupLayout.PREFERRED_SIZE, 80, javax.swing.GroupLayout.PREFERRED_SIZE)
                .addGap(340, 346, Short.MAX_VALUE))))
        .addGroup(layout.createSequentialGroup()
            .addGap(76, 76, 76)
            .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING)
                .addGroup(layout.createSequentialGroup()
                    .addComponent(jLabel9, javax.swing.GroupLayout.PREFERRED_SIZE, 56, javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                    .addComponent(jLabel10)
                    .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                    .addComponent(KpTextField, javax.swing.GroupLayout.PREFERRED_SIZE, 40, javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addGap(20, 20, 20)
                    .addComponent(jLabel11)
                    .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                    .addComponent(KiTextField, javax.swing.GroupLayout.PREFERRED_SIZE, 40, javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addGap(20, 20, 20)
                    .addComponent(jLabel12)
                    .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                    .addComponent(KdTextField, javax.swing.GroupLayout.PREFERRED_SIZE, 40, javax.swing.GroupLayout.PREFERRED_SIZE))
                .addGroup(layout.createSequentialGroup()
                    .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING, false)
                        .addComponent(jTextFieldIP_Rov)
                        .addComponent(jLabelIPHeader, javax.swing.GroupLayout.PREFERRED_SIZE, 140, javax.swing.GroupLayout.PREFERRED_SIZE))
                    .addGap(7, 7, 7)
                    .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING, false)
                        .addComponent(jTextFieldIP_Camera)
                        .addComponent(jLabelIPHeader1, javax.swing.GroupLayout.PREFERRED_SIZE, 140, javax.swing.GroupLayout.PREFERRED_SIZE))))
            .addGap(0, 0, Short.MAX_VALUE))
        .addGroup(javax.swing.GroupLayout.Alignment.TRAILING, layout.createSequentialGroup()
            .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
            .addComponent(jLabel13, javax.swing.GroupLayout.PREFERRED_SIZE, 56, javax.swing.GroupLayout.PREFERRED_SIZE)
            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
            .addComponent(jLabel14)
            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
            .addComponent(offset1TextField, javax.swing.GroupLayout.PREFERRED_SIZE, 40, javax.swing.GroupLayout.PREFERRED_SIZE)
            .addGap(18, 18, 18)
            .addComponent(jLabel15)
            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
            .addComponent(offset2TextField, javax.swing.GroupLayout.PREFERRED_SIZE, 40, javax.swing.GroupLayout.PREFERRED_SIZE)
            .addGap(36, 36, 36))
    );
    layout.setVerticalGroup(
        layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addGroup(layout.createSequentialGroup()
            .addContainerGap()
            .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING)
                .addGroup(layout.createSequentialGroup()
                    .addComponent(jLabelIPHeader, javax.swing.GroupLayout.PREFERRED_SIZE, 24, javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                    .addComponent(jTextFieldIP_Rov, javax.swing.GroupLayout.PREFERRED_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.Group
                .addGroup(layout.createSequentialGroup()
                    .addComponent(jLabelIPHeader1, javax.swing.GroupLayout.PREFERRED_SIZE, 24, javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                    .addComponent(jTextFieldIP_Camera, javax.swing.GroupLayout.PREFERRED_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.Gr
            .addGap(18, 18, 18)
            .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING, false)
                .addComponent(KdTextField, javax.swing.GroupLayout.Alignment.TRAILING)
                .addComponent(KiTextField, javax.swing.GroupLayout.Alignment.TRAILING)
                .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                    .addComponent(jLabel9, javax.swing.GroupLayout.PREFERRED_SIZE, 23, javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addComponent(KpTextField, javax.swing.GroupLayout.PREFERRED_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayo
                .addComponent(jLabel10, javax.swing.GroupLayout.Alignment.TRAILING, javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.D
```

```
346                 .addComponent(jLabel11, javax.swing.GroupLayout.Alignment.TRAILING, javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.D
347                 .addComponent(jLabel12, javax.swing.GroupLayout.Alignment.TRAILING, javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.D
348             .addGap(18, 18, 18)
349             .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
350                 .addComponent(jLabel13, javax.swing.GroupLayout.PREFERRED_SIZE, 23, javax.swing.GroupLayout.PREFERRED_SIZE)
351                 .addGroup(javax.swing.GroupLayout.Alignment.TRAILING, layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
352                     .addComponent(jLabel14)
353                     .addComponent(offset1TextField, javax.swing.GroupLayout.PREFERRED_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupL
354                     .addComponent(jLabel15, javax.swing.GroupLayout.PREFERRED_SIZE, 24, javax.swing.GroupLayout.PREFERRED_SIZE)
355                     .addComponent(offset2TextField, javax.swing.GroupLayout.PREFERRED_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupL
356             .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
357             .addComponent(jSeparator1, javax.swing.GroupLayout.PREFERRED_SIZE, 10, javax.swing.GroupLayout.PREFERRED_SIZE)
358             .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
359             .addComponent(jLabelChannelsHeader, javax.swing.GroupLayout.PREFERRED_SIZE, 28, javax.swing.GroupLayout.PREFERRED_SIZE)
360             .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
361             .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
362                 .addComponent(jLabel1, javax.swing.GroupLayout.PREFERRED_SIZE, 30, javax.swing.GroupLayout.PREFERRED_SIZE)
363                 .addComponent(jLabel2, javax.swing.GroupLayout.PREFERRED_SIZE, 30, javax.swing.GroupLayout.PREFERRED_SIZE)
364                 .addComponent(jLabel4, javax.swing.GroupLayout.PREFERRED_SIZE, 30, javax.swing.GroupLayout.PREFERRED_SIZE)
365                 .addComponent(jLabel5, javax.swing.GroupLayout.PREFERRED_SIZE, 30, javax.swing.GroupLayout.PREFERRED_SIZE))
366             .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
367             .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
368                 .addComponent(jTextFieldChannel1, javax.swing.GroupLayout.PREFERRED_SIZE, 32, javax.swing.GroupLayout.PREFERRED_SIZE)
369                 .addComponent(jTextFieldChannel2, javax.swing.GroupLayout.PREFERRED_SIZE, 32, javax.swing.GroupLayout.PREFERRED_SIZE)
370                 .addComponent(jTextFieldChannel4, javax.swing.GroupLayout.PREFERRED_SIZE, 32, javax.swing.GroupLayout.PREFERRED_SIZE)
371                 .addComponent(jTextFieldChannel3, javax.swing.GroupLayout.PREFERRED_SIZE, 32, javax.swing.GroupLayout.PREFERRED_SIZE))
372             .addGap(23, 23, 23)
373             .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
374                 .addComponent(jLabel3, javax.swing.GroupLayout.PREFERRED_SIZE, 30, javax.swing.GroupLayout.PREFERRED_SIZE)
375                 .addComponent(jLabel7, javax.swing.GroupLayout.PREFERRED_SIZE, 30, javax.swing.GroupLayout.PREFERRED_SIZE)
376                 .addComponent(jLabel6, javax.swing.GroupLayout.PREFERRED_SIZE, 30, javax.swing.GroupLayout.PREFERRED_SIZE)
377                 .addComponent(jLabel8, javax.swing.GroupLayout.PREFERRED_SIZE, 30, javax.swing.GroupLayout.PREFERRED_SIZE))
378             .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
379             .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
380                 .addComponent(jTextFieldChannel5, javax.swing.GroupLayout.PREFERRED_SIZE, 32, javax.swing.GroupLayout.PREFERRED_SIZE)
381                 .addComponent(jTextFieldChannel6, javax.swing.GroupLayout.PREFERRED_SIZE, 32, javax.swing.GroupLayout.PREFERRED_SIZE)
382                 .addComponent(jTextFieldChannel7, javax.swing.GroupLayout.PREFERRED_SIZE, 32, javax.swing.GroupLayout.PREFERRED_SIZE)
383                 .addComponent(jTextFieldChannel8, javax.swing.GroupLayout.PREFERRED_SIZE, 32, javax.swing.GroupLayout.PREFERRED_SIZE))
384             .addGap(24, 24, 24)
385             .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
386                 .addComponent(jButtonOK, javax.swing.GroupLayout.PREFERRED_SIZE, 25, javax.swing.GroupLayout.PREFERRED_SIZE)
387                 .addComponent(jButtonApply, javax.swing.GroupLayout.PREFERRED_SIZE, 25, javax.swing.GroupLayout.PREFERRED_SIZE)
388                 .addComponent(jButtonCanel, javax.swing.GroupLayout.PREFERRED_SIZE, 25, javax.swing.GroupLayout.PREFERRED_SIZE))
389             .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
390     );
391 
392     pack();
393 }// </editor-fold>
394 
395 private void jButtonOKActionPerformed(java.awt.event.ActionEvent evt) {
396     BufferedWriter writer = null;
397     try {
398         writer = new BufferedWriter(new FileWriter(file, false));
399         writer.write(jTextFieldIP_Rov.getText() + System.getProperty("line.separator"));
400         writer.write(jTextFieldIP_Camera.getText() + System.getProperty("line.separator"));
401         writer.write(jTextFieldChannel1.getText() + System.getProperty("line.separator"));
402         writer.write(jTextFieldChannel2.getText() + System.getProperty("line.separator"));
403         writer.write(jTextFieldChannel3.getText() + System.getProperty("line.separator"));
404         writer.write(jTextFieldChannel4.getText() + System.getProperty("line.separator"));
405         writer.write(jTextFieldChannel5.getText() + System.getProperty("line.separator"));
406         writer.write(jTextFieldChannel6.getText() + System.getProperty("line.separator"));
407         writer.write(jTextFieldChannel7.getText() + System.getProperty("line.separator"));
408         writer.write(jTextFieldChannel8.getText() + System.getProperty("line.separator"));
409         writer.write(KpTextField.getText() + System.getProperty("line.separator"));
410         writer.write(KiTextField.getText() + System.getProperty("line.separator"));
411         writer.write(KdTextField.getText() + System.getProperty("line.separator"));
412         writer.write(offset1TextField.getText() + System.getProperty("line.separator"));
413         writer.write(offset2TextField.getText() + System.getProperty("line.separator"));
414 
415         data.setKp(KpTextField.getText());
416         data.setKi(KiTextField.getText());
417         data.setKd(KdTextField.getText());
418         data.setOffsetDepthBeneathROV(offset1TextField.getText());
419         data.setOffsetROVdepth(offset2TextField.getText());
420         data.setIP_Rov(jTextFieldIP_Rov.getText());
421         data.setIP_Camera(jTextFieldIP_Camera.getText());
422         data.setIOLabels(jTextFieldChannel1.getText(), jTextFieldChannel2.getText(), jTextFieldChannel3.getText(), jTextFieldChannel4.getText(), jTextFieldCha
423         this.client_ROV.sendCommand("cmd_pid_p:" + KpTextField.getText());
424         this.client_ROV.sendCommand("cmd_pid_i:" + KiTextField.getText());
425         this.client_ROV.sendCommand("cmd_pid_d:" + KdTextField.getText());
426         this.client_ROV.sendCommand("cmd_offsetDepthBeneathROV:" + KdTextField.getText());
427         this.client_ROV.sendCommand("cmd_offsetROVdepth:" + KdTextField.getText());
428 
429     } catch (Exception e) {
430         System.out.println(e.getMessage());
431     } finally {
432         try {
```

```
433             writer.close();
434         } catch (Exception e) {
435             System.out.println(e.getMessage());
436         }
437     }
438     this.dispose();
439 }
440 
441 private void jButtonApplyActionPerformed(java.awt.event.ActionEvent evt) {
442     BufferedWriter writer = null;
443     try {
444         writer = new BufferedWriter(new FileWriter(file, false));
445         writer.write(jTextFieldIP_Rov.getText() + System.getProperty("line.separator"));
446         writer.write(jTextFieldIP_Camera.getText() + System.getProperty("line.separator"));
447         writer.write(jTextFieldChannel1.getText() + System.getProperty("line.separator"));
448         writer.write(jTextFieldChannel2.getText() + System.getProperty("line.separator"));
449         writer.write(jTextFieldChannel3.getText() + System.getProperty("line.separator"));
450         writer.write(jTextFieldChannel4.getText() + System.getProperty("line.separator"));
451         writer.write(jTextFieldChannel5.getText() + System.getProperty("line.separator"));
452         writer.write(jTextFieldChannel6.getText() + System.getProperty("line.separator"));
453         writer.write(jTextFieldChannel7.getText() + System.getProperty("line.separator"));
454         writer.write(jTextFieldChannel8.getText() + System.getProperty("line.separator"));
455         writer.write(KpTextField.getText() + System.getProperty("line.separator"));
456         writer.write(KiTextField.getText() + System.getProperty("line.separator"));
457         writer.write(KdTextField.getText() + System.getProperty("line.separator"));
458         writer.write(offset1TextField.getText() + System.getProperty("line.separator"));
459         writer.write(offset2TextField.getText() + System.getProperty("line.separator"));
460 
461         data.setKp(KpTextField.getText());
462         data.setKi(KiTextField.getText());
463         data.setKd(KdTextField.getText());
464         data.setOffsetDepthBeneathROV(offset1TextField.getText());
465         data.setOffsetROVdepth(offset2TextField.getText());
466         data.setIP_Rov(jTextFieldIP_Rov.getText());
467         data.setIP_Camera(jTextFieldIP_Camera.getText());
468         data.setIOLabels(jTextFieldChannel1.getText(), jTextFieldChannel2.getText(), jTextFieldChannel3.getText(), jTextFieldChannel4.getText(), jTextFieldCha
469         this.client_ROV.sendCommand("cmd_pid_p:" + KpTextField.getText());
470         this.client_ROV.sendCommand("cmd_pid_i:" + KiTextField.getText());
471         this.client_ROV.sendCommand("cmd_pid_d:" + KdTextField.getText());
472         this.client_ROV.sendCommand("cmd_offsetDepthBeneathROV:" + KdTextField.getText());
473         this.client_ROV.sendCommand("cmd_offsetROVdepth:" + KdTextField.getText());
474 
475     } catch (Exception e) {
476         System.out.println(e.getMessage());
477     } finally {
478         try {
479             writer.close();
480         } catch (Exception e) {
481             System.out.println(e.getMessage());
482         }
483     }
484 }
485 
486 /**
487  * Reads the option files and displays the current settings
488  */
489 public void getOptionsFromFile() {
490     try (BufferedReader br = new BufferedReader(new FileReader(new File("ROV Options.txt")))) {
491         this.IP_Rov = br.readLine();
492         jTextFieldIP_Rov.setText(this.IP_Rov);
493         this.IP_Camera = br.readLine();
494         jTextFieldIP_Camera.setText(this.IP_Camera);
495         for (int i = 0; i < 8; i++) {
496             channels.add(i, br.readLine());
497         }
498         jTextFieldChannel1.setText((String) channels.get(0));
499         jTextFieldChannel2.setText((String) channels.get(1));
500         jTextFieldChannel3.setText((String) channels.get(2));
501         jTextFieldChannel4.setText((String) channels.get(3));
502         jTextFieldChannel5.setText((String) channels.get(4));
503         jTextFieldChannel6.setText((String) channels.get(5));
504         jTextFieldChannel7.setText((String) channels.get(6));
505         jTextFieldChannel8.setText((String) channels.get(7));
506         KpTextField.setText(br.readLine());
507         KiTextField.setText(br.readLine());
508         KdTextField.setText(br.readLine());
509         offset1TextField.setText(br.readLine());
510         offset2TextField.setText(br.readLine());
511     } catch (Exception e) {
512         System.out.println(e.getMessage());
513     }
514 }
515 
516 private void jButtonCanelActionPerformed(java.awt.event.ActionEvent evt) {
517     this.dispose();
518 }
519 
```

```
520     private void jTextFieldChannel5ActionPerformed(java.awt.event.ActionEvent evt) {
521         // TODO add your handling code here:
522     }
523 
524     private void jTextFieldChannel6ActionPerformed(java.awt.event.ActionEvent evt) {
525         // TODO add your handling code here:
526     }
527 
528 
529     // Variables declaration - do not modify
530     private javax.swing.JTextField KdTextField;
531     private javax.swing.JTextField KiTextField;
532     private javax.swing.JTextField KpTextField;
533     private javax.swing.JButton jButtonApply;
534     private javax.swing.JButton jButtonCanel;
535     private javax.swing.JButton jButtonOK;
536     private javax.swing.JLabel jLabel1;
537     private javax.swing.JLabel jLabel10;
538     private javax.swing.JLabel jLabel11;
539     private javax.swing.JLabel jLabel12;
540     private javax.swing.JLabel jLabel13;
541     private javax.swing.JLabel jLabel14;
542     private javax.swing.JLabel jLabel15;
543     private javax.swing.JLabel jLabel2;
544     private javax.swing.JLabel jLabel3;
545     private javax.swing.JLabel jLabel4;
546     private javax.swing.JLabel jLabel5;
547     private javax.swing.JLabel jLabel6;
548     private javax.swing.JLabel jLabel7;
549     private javax.swing.JLabel jLabel8;
550     private javax.swing.JLabel jLabel9;
551     private javax.swing.JLabel jLabelChannelsHeader;
552     private javax.swing.JLabel jLabelIPHeader;
553     private javax.swing.JLabel jLabelIPHeader1;
554     private javax.swing.JSeparator jSeparator1;
555     private javax.swing.JTextField jTextFieldChannel1;
556     private javax.swing.JTextField jTextFieldChannel2;
557     private javax.swing.JTextField jTextFieldChannel3;
558     private javax.swing.JTextField jTextFieldChannel4;
559     private javax.swing.JTextField jTextFieldChannel5;
560     private javax.swing.JTextField jTextFieldChannel6;
561     private javax.swing.JTextField jTextFieldChannel7;
562     private javax.swing.JTextField jTextFieldChannel8;
563     private javax.swing.JTextField jTextFieldIP_Camera;
564     private javax.swing.JTextField jTextFieldIP_Rov;
565     private javax.swing.JTextField offset1TextField;
566     private javax.swing.JTextField offset2TextField;
567     // End of variables declaration
568 }
569 
```

D:\Dokumenter\Skule\04 - NTNU\Bachelor\Github\TowedROV_GUI\src\inputcontroller\RightThumbXListener.java

```java
/*
 * This code is for the bachelor thesis named "Towed-ROV".
 * The purpose is to build a ROV which will be towed behind a surface vessel
 * and act as a multi-sensor platform, were it shall be easy to place new
 * sensors. There will also be a video stream from the ROV.
 *
 * The system consists of two Raspberry Pis in the ROV that is connected to
 * several Arduino micro controllers. These micro controllers are connected to
 * feedback from the actuators, the echo sounder and extra optional sensors.
 * The external computer which is on the surface vessel is connected to a GPS,
 * echo sounder over USB, and the ROV over ethernet. It will present and
 * log data in addition to handle user commands for controlling the ROV.
 */
package InputController;

import com.exlumina.j360.ValueListener;

/**
 * This class is responsible for handling the input from the X axis of the right
 * stick on the Xbox 360 controller.
 */
class RightThumbXListener implements ValueListener {

    private InputController ic;

    /**
     * The constructor of the RightThumbXListener class
     *
     * @param ic the InputController
     */
    public RightThumbXListener(InputController ic) {
        this.ic = ic;
    }

    /**
     * Sets the new value from the Xbox controller
     *
     * @param newValue the new value
     */
    @Override
    public void value(int newValue) {
        newValue = map(newValue, -32768, 32768, 126, -126);
        if (newValue > -17 && newValue < 12) {
            newValue = 0;
        }
        this.ic.setBtnRx(newValue);
        System.out.printf("Rx: " + "%6d\n", newValue);
    }

    /**
     * Maps the given value range to the given output range
     *
     * @param x the value to be mapped
     * @param in_min the minimum value of the input range
```

```
55     * @param in_max the maximum value of the input range
56     * @param out_min the minimum value of the output range
57     * @param out_max the maximum value of the output range
58     * @return the mapped value
59     */
60    private int map(int x, int in_min, int in_max, int out_min, int out_max) {
61        return (x - in_min) * (out_max - out_min) / (in_max - in_min) + out_min;
62    }
63 }
64
```

D:\Dokumenter\Skule\04 - NTNU\Bachelor\Github\TowedROV_GUI\src\inputcontroller\RightThumbYListener.java

```
1  /*
2   * This code is for the bachelor thesis named "Towed-ROV".
3   * The purpose is to build a ROV which will be towed behind a surface vessel
4   * and act as a multi-sensor platform, were it shall be easy to place new
5   * sensors. There will also be a video stream from the ROV.
6   *
7   * The system consists of two Raspberry Pis in the ROV that is connected to
8   * several Arduino micro controllers. These micro controllers are connected to
9   * feedback from the actuators, the echo sounder and extra optional sensors.
10  * The external computer which is on the surface vessel is connected to a GPS,
11  * echo sounder over USB, and the ROV over ethernet. It will present and
12  * log data in addition to handle user commands for controlling the ROV.
13  */
14 package InputController;
15
16 import InputController.InputController;
17 import com.exlumina.j360.ValueListener;
18
19 /**
20  * This class is responsible for handling the input from the Y axis of the right
21  * stick on the Xbox 360 controller.
22  */
23 class RightThumbYListener implements ValueListener {
24
25     private InputController ic;
26
27     /**
28      * The constructor of the RightThumbYListener class
29      *
30      * @param ic the InputController
31      */
32     public RightThumbYListener(InputController ic) {
33         this.ic = ic;
34     }
35
36     /**
37      * Sets the new value from the Xbox controller
38      *
39      * @param newValue the new value
40      */
41     @Override
42     public void value(int newValue) {
43         newValue = map(newValue, -32768, 32768, -100, 100);
44         this.ic.setBtnRy(newValue);
45         //System.out.printf("Ry: " + "%6d\n", newValue);
46     }
47
48     /**
49      * Maps the given value range to the given output range
50      *
51      * @param x the value to be mapped
52      * @param in_min the minimum value of the input range
53      * @param in_max the maximum value of the input range
54      * @param out_min the minimum value of the output range
```

```
55     * @param out_max the maximum value of the output range
56     * @return the mapped value
57     */
58    private int map(int x, int in_min, int in_max, int out_min, int out_max) {
59        return (x - in_min) * (out_max - out_min) / (in_max - in_min) + out_min;
60    }
61 }
62
```

D:\Dokumenter\Skule\04 - NTNU\Bachelor\Github\TowedROV_GUI\src\basestation_rov\ReadSerialData.java

```
1  /*
2   * This code is for the bachelor thesis named "Towed-ROV".
3   * The purpose is to build a ROV which will be towed behind a surface vessel
4   * and act as a multi-sensor platform, were it shall be easy to place new
5   * sensors. There will also be a video stream from the ROV.
6   *
7   * The system consists of two Raspberry Pis in the ROV that is connected to
8   * several Arduino micro controllers. These micro controllers are connected to
9   * feedback from the actuators, the echo sounder and extra optional sensors.
10  * The external computer which is on the surface vessel is connected to a GPS,
11  * echo sounder over USB, and the ROV over ethernet. It will present and
12  * log data in addition to handle user commands for controlling the ROV.
13  */
14 package basestation_rov;
15
16 import java.util.HashMap;
17 import java.util.Map;
18 import java.util.concurrent.ConcurrentHashMap;
19 import jssc.SerialPort;
20 import jssc.SerialPortList;
21 import jssc.SerialPortException;
22 import ntnusubsea.gui.Data;
23
24 /**
25  * Responsible for reading serial data from the GPS, Sonar and IMU values on the
26  * base station.
27  */
28 public class ReadSerialData implements Runnable {
29
30     boolean portIsOpen = false;
31     String comPort = "";
32     String myName = "";
33     int baudRate = 0;
34     Data data = null;
35     public HashMap<String, String> incommingData = new HashMap<>();
36     private static volatile double depth;
37     private static volatile double tempC;
38
39     /**
40      * The constructor of the ReadSerialData class.
41      *
42      * @param data the shared resource Data class
43      * @param comPort the given com port to read from
44      * @param baudRate the given baud rate
45      * @param myName the name of the com port
46      */
47     public ReadSerialData(Data data, String comPort, int baudRate, String myName) {
48         this.myName = myName;
49
50         this.comPort = comPort;
51         this.baudRate = baudRate;
52         this.data = data;
53     }
54
```

```
55  /**
56   * Runs the ReadSerialData thread. Reads serial data form the given com port
57   * and at the given baud rate.
58   */
59  @Override
60  public void run() {
61      while (true) {
62          try {
63
64              readData(comPort, baudRate);
65          } catch (Exception e) {
66          }
67
68      }
69  }
70
71  /**
72   * Returns the available com ports
73   *
74   * @return the available com ports
75   */
76  public String[] getAvailableComPorts() {
77      String[] portNames = SerialPortList.getPortNames();
78
79      if (portNames.length == 0) {
80          // System.out.println("There are no serial-ports available!");
81          // System.out.println("Press enter to exit...");
82
83          try {
84              System.in.read();
85          } catch (Exception e) {
86              e.printStackTrace();
87          }
88      }
89
90      for (int i = 0; i < portNames.length; i++) {
91          //System.out.println(portNames[i]);
92      }
93      return portNames;
94  }
95
96  /**
97   * Sets the depth to the shared resource Data class
98   */
99  public void sendDepth() {
100     data.setDepth((float) depth);
101  }
102
103  /**
104   * Sets the temperature to the shared resource Data class
105   */
106  public void sendTempC() {
107     data.setTemperature((float) tempC);
108  }
109
110  /**
111   * Reads serial data form the given com port and at the given baud rate.
112   *
```

```
113     * @param comPort the given com port
114     * @param baudRate the given baud rate
115     */
116    public void readData(String comPort, int baudRate) {
117
118        // long lastTime = System.nanoTime();
119        // ConcurrentHashMap<String, String> SerialDataList = new ConcurrentHashMap<>();
120        boolean recievedData = false;
121        //Declare special symbol used in serial data stream from Arduino
122        String startChar = "<";
123        String endChar = ">";
124        String seperationChar = ":";
125
126        SerialPort serialPort = new SerialPort(comPort);
127
128        if (!portIsOpen) {
129            try {
130                serialPort.openPort();
131                portIsOpen = true;
132                // System.out.println(comPort + " is open");
133            } catch (SerialPortException ex) {
134                System.out.println(ex.getMessage());
135            }
136        }
137
138        while (recievedData == false) {
139            try {
140                Thread.sleep(100);
141            } catch (Exception ex) {
142            }
143            String buffer;
144
145            try {
146                serialPort.setParams(baudRate, 8, 1, 0);
147                buffer = serialPort.readString();
148
149                // System.out.println(buffer);
150                boolean dataNotNull = false;
151                boolean dataHasFormat = false;
152
153                if ((buffer != null)) {
154                    dataHasFormat = true;
155                } else {
156                    dataHasFormat = false;
157                    dataNotNull = false;
158                }
159                if (dataHasFormat) {
160                    if (buffer.contains("<") && buffer.contains(">")) {
161                        String dataStream = buffer;
162                        dataStream = dataStream.substring(dataStream.indexOf(startChar) + 1);
163                        dataStream = dataStream.substring(0, dataStream.indexOf(endChar));
164                        //dataStream = dataStream.replace("?", "");
165                        String[] data = dataStream.split(seperationChar);
166
167                        for (int i = 0; i < data.length; i = i + 2) {
168                            //this.data.data.put(data[i], data[i + 1]);
169                            incommingData.put(data[i], data[i + 1]);
170                        }
```

```
171                     //recievedData = true;
172                     //this.data.handleDataFromRemote();
173                     sendIncommingDataToDataHandler();
174                 }
175             }
176 
177 //          if (elapsedTimer != 0)
178 //          {
179 //              System.out.println("Data is recieved in: " + elapsedTimer + " millis"
180 //                      + " or with: " + 1000 / elapsedTimer + " Hz");
181 //          } else
182 //          {
183 //              System.out.println("Data is recieved in: " + elapsedTimer + " millis"
184 //                      + " or with: unlimited Hz!");
185 //          }
186         } catch (Exception ex) {
187             System.out.println("Lost connection to " + myName + "    Ex: " + ex);
188         }
189     }
190 }
191 
192 /**
193  * Compare keys to control values coming in from remote, and puts the
194  * correct value to correct variable in the shared resource Data class.
195  */
196 private void sendIncommingDataToDataHandler() {
197     for (Map.Entry e : incommingData.entrySet()) {
198         String key = (String) e.getKey();
199         String value = (String) e.getValue();
200 
201         switch (key) {
202             case "Satellites":
203                 data.setSatellites(Integer.parseInt(value));
204                 // setSatellites(Integer.parseInt(value));
205                 break;
206             case "Altitude":
207                 data.setAltitude(Float.parseFloat(value));
208                 //setAltitude(Float.parseFloat(value));
209                 break;
210             case "GPSAngle":
211                 data.setGPSAngle(Double.parseDouble(value));
212                 //setAngle(Float.parseFloat(value));
213                 break;
214             case "Speed":
215                 data.setSpeed(Float.parseFloat(value));
216                 //setSpeed(Float.parseFloat(value));
217                 break;
218             case "Latitude":
219                 data.setLatitude(Float.parseFloat(value));
220                 //setLatitude(Float.parseFloat(value));
221                 break;
222             case "Longitude":
223                 data.setLongitude(Float.parseFloat(value));
224                 //setLongitude(Float.parseFloat(value));
225                 break;
226             case "D":
227                 double doubleDepth = Double.parseDouble(value) * -1;
228                 data.setDepthBeneathBoat(doubleDepth);
```

```
229                 //setDepth(Float.parseFloat(value));
230                 break;
231             case "Temp":
232                 data.setTemperature(Float.parseFloat(value));
233                 //setTemperature(Float.parseFloat(value));
234                 break;
235             case "Roll":
236                 data.setRoll(Double.parseDouble(value));
237                 //setRoll(Integer.parseInt(value));
238                 break;
239             case "Pitch":
240                 data.setPitch(Double.parseDouble(value));
241                 //setPitch(Integer.parseInt(value));
242                 break;
243             case "Heading":
244 //                data.setHeading(Integer.parseInt(value));
245                 //setHeading(Integer.parseInt(value));
246                 break;
247             case "Voltage":
248                 data.setVoltage(Double.parseDouble(value));
249                 break;
250             case "TestDepth":
251                 data.setTestDepth(Double.parseDouble(value));
252                 break;
253         }
254     }
255 }
256 }
257
```

D:\Dokumenter\Skule\04 - NTNU\Bachelor\Github\TowedROV_GUI\src\ntnusubsea\gui\ROVFrame.java

```
/*
 * This code is for the bachelor thesis named "Towed-ROV".
 * The purpose is to build a ROV which will be towed behind a surface vessel
 * and act as a multi-sensor platform, were it shall be easy to place new
 * sensors. There will also be a video stream from the ROV.
 *
 * The system consists of two Raspberry Pis in the ROV that is connected to
 * several Arduino micro controllers. These micro controllers are connected to
 * feedback from the actuators, the echo sounder and extra optional sensors.
 * The external computer which is on the surface vessel is connected to a GPS,
 * echo sounder over USB, and the ROV over ethernet. It will present and
 * log data in addition to handle user commands for controlling the ROV.
 */
package ntnusubsea.gui;

import basestation_rov.LogFileHandler;
import java.awt.Color;
import java.awt.GraphicsEnvironment;
import java.awt.Image;
import java.awt.Rectangle;
import java.awt.event.ActionEvent;
import java.awt.event.KeyEvent;
import java.awt.image.BufferedImage;
import java.io.File;
import java.io.IOException;
import java.text.DecimalFormat;
import java.text.ParseException;
import java.util.Collection;
import java.util.Observable;
import java.util.Observer;
import java.util.concurrent.Executors;
import java.util.concurrent.ScheduledExecutorService;
import java.util.concurrent.TimeUnit;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.imageio.ImageIO;
import javax.swing.AbstractAction;
import javax.swing.Action;
import javax.swing.JOptionPane;
import javax.swing.ImageIcon;
import javax.swing.JFrame;
import javax.swing.KeyStroke;

/**
 * Main frame of the application. Lets the user connect, watch the video stream,
 * observe sensor values, control the lights, open all the extra tools etc.
 *
 */
public class ROVFrame extends javax.swing.JFrame implements Runnable, Observer {

    ImagePanel videoSheet;
    ImagePanel fullscreenVideoSheet;
    private BufferedImage videoImage;
    private Data data;
    private Double setpoint = 0.00;
    private int targetMode = 0;
    private EchoSounderFrame echoSounder;
    private OptionsFrame options;
    private Thread sounderThread;
    private TCPpinger client_Pinger;
    private TCPClient client_ROV;
    private TCPClient client_Camera;
    private UDPServer udpServer;
    private Sounder sounder;
    private LogFileHandler lgh;
    private VideoEncoder encoder;
    private ScheduledExecutorService clientThreadExecutor;
    private ScheduledExecutorService encoderThreadExecutor;
    private IOControlFrame io;
    private int cameraPitchValue = 0;
    private double photoModeDelay = 1.0;
    private static DecimalFormat df2 = new DecimalFormat("#.##");
    private boolean debugMode = false;

    private int cmd_actuatorPS = 0;
    private int cmd_actuatorSB = 0;

    /**
     * Creates new form ROVFrame
     *
     * @param echoSounder Echo sounder frame to show graphs
     * @param data Data containing shared variables
     * @param client_Pinger the ping TCP client
     * @param io I/O frame to control inputs and outputs
```

```
85      * @param client_ROV the ROV TCP client
86      * @param client_Camera the camera TCP client
87      * @param sounder the alarm sounder
88      * @param udpServer the camera UDP server
89      * @param lgh the log file handler
90      */
91     public ROVFrame(EchoSounderFrame echoSounder, Data data, IOControlFrame io, TCPpinger client_Pinger, TCPClient client_ROV, TCPClient client_Camer
92         this.clientThreadExecutor = null;
93         this.encoderThreadExecutor = null;
94         initComponents();
95         this.data = data;
96         this.echoSounder = echoSounder;
97         this.client_Pinger = client_Pinger;
98         this.client_ROV = client_ROV;
99         this.client_Camera = client_Camera;
100        this.udpServer = udpServer;
101        this.options = new OptionsFrame(this.data, this.client_ROV);
102        this.io = io;
103        this.sounder = sounder;
104        this.lgh = lgh;
105        this.getContentPane().setBackground(new Color(39, 44, 50));
106        this.setExtendedState(JFrame.MAXIMIZED_BOTH);
107        videoSheet = new ImagePanel(cameraPanel.getWidth(), cameraPanel.getHeight());
108        videoSheet.setBackground(cameraPanel.getBackground());
109        fullscreenVideoSheet = new ImagePanel(cameraPanel1.getWidth(), cameraPanel1.getHeight());
110        fullscreenVideoSheet.setBackground(cameraPanel1.getBackground());
111        videoSheet.setOpaque(false);
112        fullscreenVideoSheet.setOpaque(false);
113        cameraPanel.add(videoSheet);
114        cameraPanel1.add(fullscreenVideoSheet);
115        setpointLabel.setText("Current setpoint: " + setpoint + "m");
116        exitFullscreenButton.getInputMap().put(KeyStroke.getKeyStroke(KeyEvent.VK_ESCAPE, 0), "exitFullscreen");
117 //       depthInputTextField.getInputMap().put(KeyStroke.getKeyStroke(KeyEvent.VK_ENTER, 0), "sendInput");
118        exitFullscreenButton.getActionMap().put("exitFullscreen", exitFullscreenAction);
119 //       depthInputTextField.getActionMap().put("sendInput", sendInputAction);
120
121        if (this.debugMode) {
122            // ROV RPi:
123            emergencyStopButton.setEnabled(true);
124            lightSwitchBlueLED.setEnabled(true);
125            targetDistanceTextField.setEnabled(true);
126            depthModeButton.setEnabled(true);
127            seafloorModeButton.setEnabled(true);
128            InputControllerButton.setEnabled(true);
129            manualControlButton.setEnabled(true);
130            resetManualControlButton.setEnabled(true);
131            lockButton.setEnabled(true);
132            io.enableIO();
133            // Camera RPi:
134            lightSwitch_lbl.setEnabled(true);
135            lightSwitch.setEnabled(true);
136            lightSlider.setEnabled(true);
137            getPhotosButton.setEnabled(true);
138            clearImagesButton.setEnabled(true);
139            photoModeButton.setEnabled(true);
140            cameraPitchSlider.setEnabled(true);
141            cameraPitchTextField.setEnabled(true);
142            delayTextField.setEnabled(true);
143
144            // Setup for report:
145 //           manualControlButton.setSelected(true);
146 //           InputControllerButton.setSelected(true);
147 //           lightSwitchBlueLED.setSelected(true);
148 //           photoModeDelay_FB_Label.setText("0.02 s");
149 //           jMenuConnect.setText("Connected 2/2");
150 //           jMenuCalibrate.setText("Calibrated!");
151 //           jMenuRovReady.setText("ROV Ready!");
152 //           jMenuLogger.setText("Logging!");
153 //           jMenuVoltage.setText("Voltage: 38.65 V");
154 //           jMenuPing.setText("Ping (ROV): 3.24 ms");
155 //           actuatorPosLabel.setText("<html>PS: 85<br/><br/>SB: 85");
156 //           actuatorSBPosBar.setValue(85);
157 //           actuatorPSPosBar.setValue(85);
158 //           rollLabel.setText("Roll Angle: 2");
159 //           pitchLabel.setText("Pitch Angle: -26");
160 //           wingLabel.setText("Wing Angle: -40");
161 //           actuatorPSPosLabel.setText("PS Actuator Pos: 86");
162 //           actuatorSBPosLabel.setText("SB Actuator Pos: 87");
163 //           i2cErrorLabel.setText("I2C: OK");
164 //           outsideTempLabel.setText("Outside Temp: 8.25 C");
165 //           insideTempLabel.setText("Inside Temp: 18.48 C");
166 //           humidityLabel.setText("Humidity: 55.45");
167 //           pressureLabel.setText("Pressure: 150 mBar");
168 //           leakLabel.setText("No leak detected");
169 //           headingLabel.setText("Heading: 19.0500");
170 //           latitudeLabel.setText("Latitude: 62.5274");
171 //           longitudeLabel.setText("Longitude: 6.2086");
```

```
172 //          seafloorDepthBoatLabel.setText("Beneath Boat: 9.71 m");
173 //          seafloorDepthRovLabel.setText("Beneath ROV: 7.84 m");
174 //          rovDepthLabel.setText("ROV Depth: 1.57 m");
175 //
176 //          try {
177 //              jMenuConnect.setIcon(new ImageIcon(ImageIO.read(new File("src/ntnusubsea/gui/Images/Calibrated.gif"))));
178 //              jMenuCalibrate.setIcon(new ImageIcon(ImageIO.read(new File("src/ntnusubsea/gui/Images/Calibrated.gif"))));
179 //              jMenuRovReady.setIcon(new ImageIcon(ImageIO.read(new File("src/ntnusubsea/gui/Images/Calibrated.gif"))));
180 //              jMenuLogger.setIcon(new ImageIcon(ImageIO.read(new File("src/ntnusubsea/gui/Images/Calibrated.gif"))));
181 //              jMenuVoltage.setIcon(new ImageIcon(ImageIO.read(new File("src/ntnusubsea/gui/Images/Calibrated.gif"))));
182 //              jMenuPing.setIcon(new ImageIcon(ImageIO.read(new File("src/ntnusubsea/gui/Images/Calibrated.gif"))));
183 //          } catch (IOException e) {
184 //          }
185       }
186   }
187
188   /**
189    * Calls the paintSheet method which updates the diplayed image.
190    *
191    * @param image The image to be displayed on the GUI
192    */
193   public void showImage(BufferedImage image) {
194       if (fullscreen.isVisible()) {
195           fullscreenVideoSheet.setSize(cameraPanel1.getSize());
196           fullscreenVideoSheet.paintSheet(ImageUtils.resize(image, fullscreenVideoSheet.getParent().getWidth(), fullscreenVideoSheet.getParent().getHeight()));
197       } else {
198           videoSheet.paintSheet(ImageUtils.resize(image, videoSheet.getParent().getWidth(), videoSheet.getParent().getHeight()));
199           videoSheet.setSize(cameraPanel.getSize());
200       }
201   }
202
203   /**
204    * This method is called from within the constructor to initialize the form.
205    * WARNING: Do NOT modify this code. The content of this method is always
206    * regenerated by the Form Editor.
207    */
208   @SuppressWarnings("unchecked")
209   // <editor-fold defaultstate="collapsed" desc="Generated Code">
210   private void initComponents() {
211       java.awt.GridBagConstraints gridBagConstraints;
212
213       fullscreen = new javax.swing.JFrame();
214       cameraPanel1 = new javax.swing.JPanel();
215       exitFullscreenButton = new javax.swing.JButton();
216       buttonGroup1 = new javax.swing.ButtonGroup();
217       helpframe = new javax.swing.JFrame();
218       jPanel1 = new javax.swing.JPanel();
219       helpFrameOKbutton = new javax.swing.JButton();
220       jLabel1 = new javax.swing.JLabel();
221       window = new javax.swing.JPanel();
222       background = new javax.swing.JPanel();
223       cameraPanel = new javax.swing.JPanel();
224       fullscreenButton = new javax.swing.JButton();
225       controlPanel = new javax.swing.JPanel();
226       depthPanel = new javax.swing.JPanel();
227       targetDistanceTextField = new javax.swing.JFormattedTextField();
228       depthHeader = new javax.swing.JLabel();
229       jSeparator4 = new javax.swing.JSeparator();
230       depthModeButton = new javax.swing.JRadioButton();
231       seafloorModeButton = new javax.swing.JRadioButton();
232       setpointLabel = new javax.swing.JLabel();
233       manualControlButton = new javax.swing.JToggleButton();
234       jLabel5 = new javax.swing.JLabel();
235       resetManualControlButton = new javax.swing.JButton();
236       actuatorControlPS = new javax.swing.JSlider();
237       actuatorControlSB = new javax.swing.JSlider();
238       actuatorPosLabel = new javax.swing.JLabel();
239       lockButton = new javax.swing.JToggleButton();
240       InputControllerButton = new javax.swing.JToggleButton();
241       lightPanel = new javax.swing.JPanel();
242       lightHeader = new javax.swing.JLabel();
243       jSeparator5 = new javax.swing.JSeparator();
244       lightSwitch = new javax.swing.JToggleButton();
245       lightSwitch_lbl = new javax.swing.JLabel();
246       filler2 = new javax.swing.Box.Filler(new java.awt.Dimension(0, 180), new java.awt.Dimension(0, 180), new java.awt.Dimension(32767, 180));
247       lightSlider = new javax.swing.JSlider();
248       lightSwitchBlueLED = new javax.swing.JToggleButton();
249       jLabel6 = new javax.swing.JLabel();
250       emergencyPanel = new javax.swing.JPanel();
251       emergencyHeader = new javax.swing.JLabel();
252       emergencyStopButton = new javax.swing.JButton();
253       jSeparator6 = new javax.swing.JSeparator();
254       filler1 = new javax.swing.Box.Filler(new java.awt.Dimension(0, 100), new java.awt.Dimension(0, 100), new java.awt.Dimension(32767, 100));
255       cameraControlPanel = new javax.swing.JPanel();
256       delayTextField = new javax.swing.JFormattedTextField();
257       cameraHeader = new javax.swing.JLabel();
258       jSeparator9 = new javax.swing.JSeparator();
```

```
259         photoModeButton = new javax.swing.JToggleButton();
260         jLabel2 = new javax.swing.JLabel();
261         jLabel3 = new javax.swing.JLabel();
262         cameraPitchSlider = new javax.swing.JSlider();
263         jLabel4 = new javax.swing.JLabel();
264         cameraPitchTextField = new javax.swing.JFormattedTextField();
265         cameraPitchLabel = new javax.swing.JLabel();
266         photoModeDelayLabel = new javax.swing.JLabel();
267         getPhotosButton = new javax.swing.JButton();
268         photoModeDelay_FB_Label = new javax.swing.JLabel();
269         clearImagesButton = new javax.swing.JButton();
270         imageNumberLabel = new javax.swing.JLabel();
271         jSeparator2 = new javax.swing.JSeparator();
272         jSeparator8 = new javax.swing.JSeparator();
273         jSeparator1 = new javax.swing.JSeparator();
274         infoPanel = new javax.swing.JPanel();
275         actuatorPanel1 = new javax.swing.JPanel();
276         actuatorHeader1 = new javax.swing.JLabel();
277         actuatorPSPosBar = new javax.swing.JProgressBar();
278         warningLabel1 = new javax.swing.JLabel();
279         actuatorPanel2 = new javax.swing.JPanel();
280         actuatorHeader2 = new javax.swing.JLabel();
281         actuatorSBPosBar = new javax.swing.JProgressBar();
282         warningLabel2 = new javax.swing.JLabel();
283         jPanel2 = new javax.swing.JPanel();
284         jPanel3 = new javax.swing.JPanel();
285         jLabel7 = new javax.swing.JLabel();
286         wingLabel = new javax.swing.JLabel();
287         pitchLabel = new javax.swing.JLabel();
288         rollLabel = new javax.swing.JLabel();
289         jLabel9 = new javax.swing.JLabel();
290         jLabel15 = new javax.swing.JLabel();
291         actuatorPSPosLabel = new javax.swing.JLabel();
292         actuatorSBPosLabel = new javax.swing.JLabel();
293         i2cErrorLabel = new javax.swing.JLabel();
294         jPanel4 = new javax.swing.JPanel();
295         seafloorDepthRovLabel = new javax.swing.JLabel();
296         rovDepthLabel = new javax.swing.JLabel();
297         seafloorDepthBoatLabel = new javax.swing.JLabel();
298         jLabel8 = new javax.swing.JLabel();
299         jLabel10 = new javax.swing.JLabel();
300         jLabel16 = new javax.swing.JLabel();
301         jPanel5 = new javax.swing.JPanel();
302         latitudeLabel = new javax.swing.JLabel();
303         longitudeLabel = new javax.swing.JLabel();
304         headingLabel = new javax.swing.JLabel();
305         jLabel11 = new javax.swing.JLabel();
306         jLabel12 = new javax.swing.JLabel();
307         jLabel17 = new javax.swing.JLabel();
308         jPanel7 = new javax.swing.JPanel();
309         leakLabel = new javax.swing.JLabel();
310         jLabel19 = new javax.swing.JLabel();
311         jLabel20 = new javax.swing.JLabel();
312         jLabel21 = new javax.swing.JLabel();
313         outsideTempLabel = new javax.swing.JLabel();
314         insideTempLabel = new javax.swing.JLabel();
315         humidityLabel = new javax.swing.JLabel();
316         pressureLabel = new javax.swing.JLabel();
317         actuatorSBPosLabel1 = new javax.swing.JLabel();
318         filler3 = new javax.swing.Box.Filler(new java.awt.Dimension(0, 0), new java.awt.Dimension(0, 0), new java.awt.Dimension(32767, 0));
319         jMenuBar = new javax.swing.JMenuBar();
320         jMenuTools = new javax.swing.JMenu();
321         jMenuEchosounder = new javax.swing.JMenuItem();
322         jMenuIOController = new javax.swing.JMenuItem();
323         jMenuOptions = new javax.swing.JMenuItem();
324         jMenuItemExit = new javax.swing.JMenuItem();
325         jMenuHelp = new javax.swing.JMenu();
326         jMenuAbout = new javax.swing.JMenuItem();
327         jMenuConnect = new javax.swing.JMenu();
328         jMenuItemConnect = new javax.swing.JMenuItem();
329         jMenuItemDisconnect = new javax.swing.JMenuItem();
330         jMenuCalibrate = new javax.swing.JMenu();
331         calibrateMenuItem = new javax.swing.JMenuItem();
332         jMenuRovReady = new javax.swing.JMenu();
333         jMenuLogger = new javax.swing.JMenu();
334         jMenuItemStartLogging = new javax.swing.JMenuItem();
335         jMenuItemStopLogging = new javax.swing.JMenuItem();
336         jMenu1 = new javax.swing.JMenu();
337         jMenuVoltage = new javax.swing.JMenu();
338         jMenu3 = new javax.swing.JMenu();
339         jMenuPing = new javax.swing.JMenu();
340 
341         fullscreen.setDefaultCloseOperation(javax.swing.WindowConstants.DISPOSE_ON_CLOSE);
342         fullscreen.setBackground(new java.awt.Color(39, 44, 50));
343         fullscreen.setFocusTraversalPolicyProvider(true);
344         fullscreen.setForeground(new java.awt.Color(39, 44, 50));
345         fullscreen.setLocationByPlatform(true);
```

```
346     fullscreen.setUndecorated(true);
347     fullscreen.addKeyListener(new java.awt.event.KeyAdapter() {
348         public void keyPressed(java.awt.event.KeyEvent evt) {
349             fullscreenKeyPressed(evt);
350         }
351     });
352 
353     cameraPanel1.setBackground(new java.awt.Color(39, 44, 50));
354     cameraPanel1.setForeground(new java.awt.Color(39, 44, 50));
355     cameraPanel1.setToolTipText("");
356     cameraPanel1.setMinimumSize(new java.awt.Dimension(450, 320));
357     cameraPanel1.setPreferredSize(new java.awt.Dimension(718, 580));
358 
359     exitFullscreenButton.setBackground(new java.awt.Color(0, 0, 0));
360     exitFullscreenButton.setIcon(new javax.swing.ImageIcon(getClass().getResource("/ntnusubsea/gui/Images/exitfullscreen1.gif"))); // NOI18N
361     exitFullscreenButton.setBorder(null);
362     exitFullscreenButton.setContentAreaFilled(false);
363     exitFullscreenButton.setCursor(new java.awt.Cursor(java.awt.Cursor.HAND_CURSOR));
364     exitFullscreenButton.addActionListener(new java.awt.event.ActionListener() {
365         public void actionPerformed(java.awt.event.ActionEvent evt) {
366             exitFullscreenButtonActionPerformed(evt);
367         }
368     });
369 
370     javax.swing.GroupLayout cameraPanel1Layout = new javax.swing.GroupLayout(cameraPanel1);
371     cameraPanel1.setLayout(cameraPanel1Layout);
372     cameraPanel1Layout.setHorizontalGroup(
373         cameraPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
374         .addGroup(javax.swing.GroupLayout.Alignment.TRAILING, cameraPanel1Layout.createSequentialGroup()
375             .addGap(0, 708, Short.MAX_VALUE)
376             .addComponent(exitFullscreenButton, javax.swing.GroupLayout.PREFERRED_SIZE, 30, javax.swing.GroupLayout.PREFERRED_SIZE))
377     );
378     cameraPanel1Layout.setVerticalGroup(
379         cameraPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
380         .addGroup(javax.swing.GroupLayout.Alignment.TRAILING, cameraPanel1Layout.createSequentialGroup()
381             .addGap(0, 572, Short.MAX_VALUE)
382             .addComponent(exitFullscreenButton, javax.swing.GroupLayout.PREFERRED_SIZE, 30, javax.swing.GroupLayout.PREFERRED_SIZE))
383     );
384 
385     javax.swing.GroupLayout fullscreenLayout = new javax.swing.GroupLayout(fullscreen.getContentPane());
386     fullscreen.getContentPane().setLayout(fullscreenLayout);
387     fullscreenLayout.setHorizontalGroup(
388         fullscreenLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
389         .addComponent(cameraPanel1, javax.swing.GroupLayout.Alignment.TRAILING, javax.swing.GroupLayout.DEFAULT_SIZE, 738, Short.MAX_VALUE)
390     );
391     fullscreenLayout.setVerticalGroup(
392         fullscreenLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
393         .addComponent(cameraPanel1, javax.swing.GroupLayout.Alignment.TRAILING, javax.swing.GroupLayout.DEFAULT_SIZE, 602, Short.MAX_VALUE)
394     );
395 
396     helpframe.setDefaultCloseOperation(javax.swing.WindowConstants.DISPOSE_ON_CLOSE);
397     helpframe.setTitle("About");
398     helpframe.setBackground(new java.awt.Color(39, 44, 50));
399     helpframe.setForeground(new java.awt.Color(39, 44, 50));
400     helpframe.setType(java.awt.Window.Type.POPUP);
401 
402     jPanel1.setBackground(new java.awt.Color(39, 44, 50));
403     jPanel1.setForeground(new java.awt.Color(39, 44, 50));
404     jPanel1.setMaximumSize(new java.awt.Dimension(395, 304));
405     jPanel1.setMinimumSize(new java.awt.Dimension(395, 304));
406     jPanel1.setPreferredSize(new java.awt.Dimension(395, 304));
407 
408     helpFrameOKbutton.setBackground(new java.awt.Color(39, 44, 50));
409     helpFrameOKbutton.setForeground(new java.awt.Color(255, 255, 255));
410     helpFrameOKbutton.setText("Close");
411     helpFrameOKbutton.setFocusPainted(false);
412     helpFrameOKbutton.addActionListener(new java.awt.event.ActionListener() {
413         public void actionPerformed(java.awt.event.ActionEvent evt) {
414             helpFrameOKbuttonActionPerformed(evt);
415         }
416     });
417 
418     jLabel1.setForeground(new java.awt.Color(255, 255, 255));
419     jLabel1.setText("<html>This code is for the bachelor thesis named \"Towed-ROV\". The purpose is to build a ROV which will be towed behind a surface ve
420     jLabel1.setToolTipText("");
421 
422     javax.swing.GroupLayout jPanel1Layout = new javax.swing.GroupLayout(jPanel1);
423     jPanel1.setLayout(jPanel1Layout);
424     jPanel1Layout.setHorizontalGroup(
425         jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
426         .addGroup(jPanel1Layout.createSequentialGroup()
427             .addContainerGap()
428             .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
429                 .addGroup(jPanel1Layout.createSequentialGroup()
430                     .addGap(0, 160, Short.MAX_VALUE)
431                     .addComponent(helpFrameOKbutton)
432                     .addGap(0, 161, Short.MAX_VALUE))
```

```
433                 .addComponent(jLabel1, javax.swing.GroupLayout.PREFERRED_SIZE, 0, Short.MAX_VALUE))
434             .addContainerGap())
435     );
436     jPanel1Layout.setVerticalGroup(
437         jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
438         .addGroup(jPanel1Layout.createSequentialGroup()
439             .addContainerGap()
440             .addComponent(jLabel1, javax.swing.GroupLayout.DEFAULT_SIZE, 254, Short.MAX_VALUE)
441             .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
442             .addComponent(helpFrameOKbutton)
443             .addContainerGap())
444     );
445
446     javax.swing.GroupLayout helpframeLayout = new javax.swing.GroupLayout(helpframe.getContentPane());
447     helpframe.getContentPane().setLayout(helpframeLayout);
448     helpframeLayout.setHorizontalGroup(
449         helpframeLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
450         .addComponent(jPanel1, javax.swing.GroupLayout.Alignment.TRAILING, javax.swing.GroupLayout.DEFAULT_SIZE, 407, Short.MAX_VALUE)
451     );
452     helpframeLayout.setVerticalGroup(
453         helpframeLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
454         .addComponent(jPanel1, javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRE
455     );
456
457     setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);
458     setTitle("Towed ROV");
459     setAutoRequestFocus(false);
460     setBackground(new java.awt.Color(39, 44, 50));
461     setCursor(new java.awt.Cursor(java.awt.Cursor.DEFAULT_CURSOR));
462     setForeground(new java.awt.Color(39, 44, 50));
463     setMaximumSize(new java.awt.Dimension(1460, 890));
464     setMinimumSize(new java.awt.Dimension(1460, 890));
465     setPreferredSize(new java.awt.Dimension(1460, 890));
466     setSize(new java.awt.Dimension(1445, 853));
467     getContentPane().setLayout(new java.awt.GridBagLayout());
468
469     window.setBackground(new java.awt.Color(39, 44, 50));
470     window.setForeground(new java.awt.Color(39, 44, 50));
471     window.setMinimumSize(new java.awt.Dimension(1445, 830));
472     window.setPreferredSize(new java.awt.Dimension(1445, 830));
473     window.setLayout(new java.awt.GridBagLayout());
474
475     background.setBackground(new java.awt.Color(39, 44, 50));
476     background.setForeground(new java.awt.Color(39, 44, 50));
477     background.setToolTipText("");
478     background.setAlignmentX(5.0F);
479     background.setMinimumSize(new java.awt.Dimension(1445, 830));
480     background.setPreferredSize(new java.awt.Dimension(1445, 830));
481
482     cameraPanel.setBackground(new java.awt.Color(39, 44, 50));
483     cameraPanel.setForeground(new java.awt.Color(39, 44, 50));
484     cameraPanel.setToolTipText("");
485     cameraPanel.setAlignmentX(0.8F);
486     cameraPanel.setAlignmentY(0.8F);
487     cameraPanel.setMaximumSize(new java.awt.Dimension(985, 605));
488     cameraPanel.setMinimumSize(new java.awt.Dimension(985, 605));
489     cameraPanel.setPreferredSize(new java.awt.Dimension(985, 605));
490
491     fullscreenButton.setBackground(new java.awt.Color(0, 0, 0));
492     fullscreenButton.setForeground(new java.awt.Color(255, 255, 255));
493     fullscreenButton.setIcon(new javax.swing.ImageIcon(getClass().getResource("/ntnusubsea/gui/Images/fullscreen1.gif"))); // NOI18N
494     fullscreenButton.setBorder(null);
495     fullscreenButton.setContentAreaFilled(false);
496     fullscreenButton.setCursor(new java.awt.Cursor(java.awt.Cursor.HAND_CURSOR));
497     fullscreenButton.setHideActionText(true);
498     fullscreenButton.setName(""); // NOI18N
499     fullscreenButton.addActionListener(new java.awt.event.ActionListener() {
500         public void actionPerformed(java.awt.event.ActionEvent evt) {
501             fullscreenButtonActionPerformed(evt);
502         }
503     });
504
505     javax.swing.GroupLayout cameraPanelLayout = new javax.swing.GroupLayout(cameraPanel);
506     cameraPanel.setLayout(cameraPanelLayout);
507     cameraPanelLayout.setHorizontalGroup(
508         cameraPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
509         .addGroup(cameraPanelLayout.createSequentialGroup()
510             .addContainerGap(955, Short.MAX_VALUE)
511             .addComponent(fullscreenButton, javax.swing.GroupLayout.PREFERRED_SIZE, 30, javax.swing.GroupLayout.PREFERRED_SIZE))
512     );
513     cameraPanelLayout.setVerticalGroup(
514         cameraPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
515         .addGroup(javax.swing.GroupLayout.Alignment.TRAILING, cameraPanelLayout.createSequentialGroup()
516             .addContainerGap(575, Short.MAX_VALUE)
517             .addComponent(fullscreenButton, javax.swing.GroupLayout.PREFERRED_SIZE, 30, javax.swing.GroupLayout.PREFERRED_SIZE))
518     );
519
```

```
520     controlPanel.setBackground(new java.awt.Color(39, 44, 50));
521     controlPanel.setForeground(new java.awt.Color(39, 44, 50));
522     controlPanel.setMinimumSize(new java.awt.Dimension(150, 140));
523     controlPanel.setPreferredSize(new java.awt.Dimension(768, 190));
524
525     depthPanel.setBackground(new java.awt.Color(39, 44, 50));
526     depthPanel.setMaximumSize(new java.awt.Dimension(260, 210));
527
528     targetDistanceTextField.setHorizontalAlignment(javax.swing.JTextField.CENTER);
529     targetDistanceTextField.setToolTipText("Depth (m)");
530     targetDistanceTextField.addActionListener(new java.awt.event.ActionListener() {
531         public void actionPerformed(java.awt.event.ActionEvent evt) {
532             targetDistanceTextFieldActionPerformed(evt);
533         }
534     });
535
536     depthHeader.setFont(new java.awt.Font("Tahoma", 0, 14)); // NOI18N
537     depthHeader.setForeground(new java.awt.Color(255, 255, 255));
538     depthHeader.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
539     depthHeader.setText("ROV Control");
540
541     jSeparator4.setBackground(new java.awt.Color(67, 72, 83));
542     jSeparator4.setForeground(new java.awt.Color(67, 72, 83));
543
544     depthModeButton.setBackground(new java.awt.Color(39, 44, 50));
545     buttonGroup1.add(depthModeButton);
546     depthModeButton.setForeground(new java.awt.Color(255, 255, 255));
547     depthModeButton.setSelected(true);
548     depthModeButton.setText("Depth");
549     depthModeButton.setCursor(new java.awt.Cursor(java.awt.Cursor.HAND_CURSOR));
550     depthModeButton.setFocusPainted(false);
551     depthModeButton.addActionListener(new java.awt.event.ActionListener() {
552         public void actionPerformed(java.awt.event.ActionEvent evt) {
553             depthModeButtonActionPerformed(evt);
554         }
555     });
556
557     seafloorModeButton.setBackground(new java.awt.Color(39, 44, 50));
558     buttonGroup1.add(seafloorModeButton);
559     seafloorModeButton.setForeground(new java.awt.Color(255, 255, 255));
560     seafloorModeButton.setText("Distance from seafloor");
561     seafloorModeButton.setCursor(new java.awt.Cursor(java.awt.Cursor.HAND_CURSOR));
562     seafloorModeButton.setFocusPainted(false);
563     seafloorModeButton.addActionListener(new java.awt.event.ActionListener() {
564         public void actionPerformed(java.awt.event.ActionEvent evt) {
565             seafloorModeButtonActionPerformed(evt);
566         }
567     });
568
569     setpointLabel.setBackground(new java.awt.Color(39, 44, 50));
570     setpointLabel.setForeground(new java.awt.Color(255, 255, 255));
571     setpointLabel.setHorizontalAlignment(javax.swing.SwingConstants.TRAILING);
572     setpointLabel.setText("Current setpoint: 0.0m");
573     setpointLabel.setToolTipText("");
574     setpointLabel.setHorizontalTextPosition(javax.swing.SwingConstants.LEADING);
575     setpointLabel.setOpaque(true);
576
577     manualControlButton.setBackground(new java.awt.Color(39, 44, 50));
578     buttonGroup1.add(manualControlButton);
579     manualControlButton.setForeground(new java.awt.Color(39, 44, 50));
580     manualControlButton.setIcon(new javax.swing.ImageIcon(getClass().getResource("/ntnusubsea/gui/Images/switch-off.gif"))); // NOI18N
581     manualControlButton.setBorder(null);
582     manualControlButton.setContentAreaFilled(false);
583     manualControlButton.setCursor(new java.awt.Cursor(java.awt.Cursor.HAND_CURSOR));
584     manualControlButton.setDisabledIcon(new javax.swing.ImageIcon(getClass().getResource("/ntnusubsea/gui/Images/switch-off.gif"))); // NOI18N
585     manualControlButton.setFocusPainted(false);
586     manualControlButton.setFocusable(false);
587     manualControlButton.setSelectedIcon(new javax.swing.ImageIcon(getClass().getResource("/ntnusubsea/gui/Images/switch-on.gif"))); // NOI18N
588     manualControlButton.addActionListener(new java.awt.event.ActionListener() {
589         public void actionPerformed(java.awt.event.ActionEvent evt) {
590             manualControlButtonActionPerformed(evt);
591         }
592     });
593
594     jLabel5.setForeground(new java.awt.Color(255, 255, 255));
595     jLabel5.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
596     jLabel5.setText("Manual Control:");
597     jLabel5.setPreferredSize(new java.awt.Dimension(85, 16));
598
599     resetManualControlButton.setBackground(new java.awt.Color(39, 44, 50));
600     resetManualControlButton.setFont(new java.awt.Font("Dialog", 1, 10)); // NOI18N
601     resetManualControlButton.setForeground(new java.awt.Color(255, 255, 255));
602     resetManualControlButton.setText("RESET");
603     resetManualControlButton.setCursor(new java.awt.Cursor(java.awt.Cursor.HAND_CURSOR));
604     resetManualControlButton.setFocusPainted(false);
605     resetManualControlButton.addActionListener(new java.awt.event.ActionListener() {
606         public void actionPerformed(java.awt.event.ActionEvent evt) {
```

```
607             resetManualControlButtonActionPerformed(evt);
608         }
609     });
610 
611     actuatorControlPS.setBackground(new java.awt.Color(39, 44, 50));
612     actuatorControlPS.setMaximum(254);
613     actuatorControlPS.setMinimum(1);
614     actuatorControlPS.setOrientation(javax.swing.JSlider.VERTICAL);
615     actuatorControlPS.setValue(127);
616     actuatorControlPS.setCursor(new java.awt.Cursor(java.awt.Cursor.HAND_CURSOR));
617     actuatorControlPS.setEnabled(false);
618     actuatorControlPS.addMouseListener(new java.awt.event.MouseAdapter() {
619         public void mouseReleased(java.awt.event.MouseEvent evt) {
620             actuatorControlPSMouseReleased(evt);
621         }
622     });
623 
624     actuatorControlSB.setBackground(new java.awt.Color(39, 44, 50));
625     actuatorControlSB.setMaximum(254);
626     actuatorControlSB.setMinimum(1);
627     actuatorControlSB.setOrientation(javax.swing.JSlider.VERTICAL);
628     actuatorControlSB.setValue(127);
629     actuatorControlSB.setCursor(new java.awt.Cursor(java.awt.Cursor.HAND_CURSOR));
630     actuatorControlSB.setEnabled(false);
631     actuatorControlSB.addMouseListener(new java.awt.event.MouseAdapter() {
632         public void mouseReleased(java.awt.event.MouseEvent evt) {
633             actuatorControlSBMouseReleased(evt);
634         }
635     });
636 
637     actuatorPosLabel.setForeground(new java.awt.Color(255, 255, 255));
638     actuatorPosLabel.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
639     actuatorPosLabel.setText("<html>PS: 127<br/><br/>SB: 127");
640 
641     lockButton.setBackground(new java.awt.Color(39, 44, 50));
642     lockButton.setFont(new java.awt.Font("Dialog", 1, 10)); // NOI18N
643     lockButton.setForeground(new java.awt.Color(255, 255, 255));
644     lockButton.setText("LOCK");
645     lockButton.setCursor(new java.awt.Cursor(java.awt.Cursor.HAND_CURSOR));
646     lockButton.setFocusPainted(false);
647     lockButton.setMaximumSize(new java.awt.Dimension(63, 30));
648     lockButton.setMinimumSize(new java.awt.Dimension(63, 30));
649     lockButton.addActionListener(new java.awt.event.ActionListener() {
650         public void actionPerformed(java.awt.event.ActionEvent evt) {
651             lockButtonActionPerformed(evt);
652         }
653     });
654 
655     InputControllerButton.setBackground(new java.awt.Color(39, 44, 50));
656     InputControllerButton.setFont(new java.awt.Font("Dialog", 1, 10)); // NOI18N
657     InputControllerButton.setForeground(new java.awt.Color(255, 255, 255));
658     InputControllerButton.setText("IC");
659     InputControllerButton.setCursor(new java.awt.Cursor(java.awt.Cursor.HAND_CURSOR));
660     InputControllerButton.setFocusPainted(false);
661     InputControllerButton.setMaximumSize(new java.awt.Dimension(63, 30));
662     InputControllerButton.setMinimumSize(new java.awt.Dimension(63, 30));
663     InputControllerButton.addActionListener(new java.awt.event.ActionListener() {
664         public void actionPerformed(java.awt.event.ActionEvent evt) {
665             InputControllerButtonActionPerformed(evt);
666         }
667     });
668 
669     javax.swing.GroupLayout depthPanelLayout = new javax.swing.GroupLayout(depthPanel);
670     depthPanel.setLayout(depthPanelLayout);
671     depthPanelLayout.setHorizontalGroup(
672         depthPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
673         .addComponent(jSeparator4)
674         .addComponent(depthHeader, javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
675         .addGroup(depthPanelLayout.createSequentialGroup()
676             .addContainerGap()
677             .addComponent(manualControlButton, javax.swing.GroupLayout.PREFERRED_SIZE, 88, javax.swing.GroupLayout.PREFERRED_SIZE)
678             .addGap(0, 0, 0)
679             .addGroup(depthPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING, false)
680                 .addGroup(depthPanelLayout.createSequentialGroup()
681                     .addComponent(actuatorControlPS, javax.swing.GroupLayout.PREFERRED_SIZE, 26, javax.swing.GroupLayout.PREFERRED_SIZE)
682                     .addGap(4, 4, 4)
683                     .addComponent(actuatorControlSB, javax.swing.GroupLayout.PREFERRED_SIZE, 26, javax.swing.GroupLayout.PREFERRED_SIZE)
684                     .addGap(4, 4, 4)
685                     .addComponent(actuatorPosLabel, javax.swing.GroupLayout.PREFERRED_SIZE, 50, javax.swing.GroupLayout.PREFERRED_SIZE))
686                 .addComponent(jLabel5, javax.swing.GroupLayout.PREFERRED_SIZE, 110, javax.swing.GroupLayout.PREFERRED_SIZE))
687             .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
688             .addGroup(depthPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
689                 .addComponent(lockButton, javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
690                 .addComponent(resetManualControlButton, javax.swing.GroupLayout.Alignment.TRAILING, javax.swing.GroupLayout.DEFAULT_SIZE, javax.swi
691                 .addComponent(InputControllerButton, javax.swing.GroupLayout.Alignment.TRAILING, javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.G
692             .addGap(18, 18, 18))
693         .addGroup(depthPanelLayout.createSequentialGroup()
```

```
            .addGap(43, 43, 43)
            .addGroup(depthPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                .addGroup(depthPanelLayout.createSequentialGroup()
                    .addGap(13, 13, 13)
                    .addComponent(setpointLabel, javax.swing.GroupLayout.PREFERRED_SIZE, 140, javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addGap(18, 18, 18)
                    .addComponent(targetDistanceTextField, javax.swing.GroupLayout.PREFERRED_SIZE, 34, javax.swing.GroupLayout.PREFERRED_SIZE))
                .addGroup(depthPanelLayout.createSequentialGroup()
                    .addComponent(depthModeButton)
                    .addGap(6, 6, 6)
                    .addComponent(seafloorModeButton)))
            .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
    );
    depthPanelLayout.setVerticalGroup(
        depthPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addGroup(depthPanelLayout.createSequentialGroup()
            .addComponent(depthHeader, javax.swing.GroupLayout.PREFERRED_SIZE, 27, javax.swing.GroupLayout.PREFERRED_SIZE)
            .addGap(6, 6, 6)
            .addComponent(jSeparator4, javax.swing.GroupLayout.PREFERRED_SIZE, 10, javax.swing.GroupLayout.PREFERRED_SIZE)
            .addGroup(depthPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                .addComponent(setpointLabel, javax.swing.GroupLayout.PREFERRED_SIZE, 24, javax.swing.GroupLayout.PREFERRED_SIZE)
                .addComponent(targetDistanceTextField, javax.swing.GroupLayout.PREFERRED_SIZE, 22, javax.swing.GroupLayout.PREFERRED_SIZE))
            .addGroup(depthPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                .addComponent(depthModeButton)
                .addGroup(depthPanelLayout.createSequentialGroup()
                    .addGap(2, 2, 2)
                    .addComponent(seafloorModeButton, javax.swing.GroupLayout.PREFERRED_SIZE, 24, javax.swing.GroupLayout.PREFERRED_SIZE)))
            .addGroup(depthPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                .addGroup(depthPanelLayout.createSequentialGroup()
                    .addGap(18, 18, 18)
                    .addComponent(jLabel5, javax.swing.GroupLayout.PREFERRED_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.P
                    .addGap(11, 11, 11)
                    .addGroup(depthPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                        .addComponent(actuatorControlPS, javax.swing.GroupLayout.PREFERRED_SIZE, 70, javax.swing.GroupLayout.PREFERRED_SIZE)
                        .addComponent(actuatorControlSB, javax.swing.GroupLayout.PREFERRED_SIZE, 70, javax.swing.GroupLayout.PREFERRED_SIZE)
                        .addGroup(depthPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING, false)
                            .addGroup(javax.swing.GroupLayout.Alignment.TRAILING, depthPanelLayout.createSequentialGroup()
                                .addGap(6, 6, 6)
                                .addComponent(actuatorPosLabel, javax.swing.GroupLayout.PREFERRED_SIZE, 60, javax.swing.GroupLayout.PREFERRED_SIZE))
                            .addGroup(depthPanelLayout.createSequentialGroup()
                                .addGap(12, 12, 12)
                                .addComponent(resetManualControlButton, javax.swing.GroupLayout.PREFERRED_SIZE, 15, javax.swing.GroupLayout.PREFERRED_S
                                .addGap(0, 0, 0)
                                .addComponent(lockButton, javax.swing.GroupLayout.PREFERRED_SIZE, 15, javax.swing.GroupLayout.PREFERRED_SIZE)
                                .addGap(0, 0, 0)
                                .addComponent(InputControllerButton, javax.swing.GroupLayout.PREFERRED_SIZE, 15, javax.swing.GroupLayout.PREFERRED_SIZE
                .addGroup(depthPanelLayout.createSequentialGroup()
                    .addGap(60, 60, 60)
                    .addComponent(manualControlButton, javax.swing.GroupLayout.PREFERRED_SIZE, 39, javax.swing.GroupLayout.PREFERRED_SIZE))))
    );

    lightPanel.setBackground(new java.awt.Color(39, 44, 50));
    lightPanel.setMaximumSize(new java.awt.Dimension(156, 213));

    lightHeader.setFont(new java.awt.Font("Tahoma", 0, 14)); // NOI18N
    lightHeader.setForeground(new java.awt.Color(255, 255, 255));
    lightHeader.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
    lightHeader.setText("Lights");

    jSeparator5.setBackground(new java.awt.Color(67, 72, 83));
    jSeparator5.setForeground(new java.awt.Color(67, 72, 83));

    lightSwitch.setBackground(new java.awt.Color(39, 44, 50));
    lightSwitch.setForeground(new java.awt.Color(39, 44, 50));
    lightSwitch.setIcon(new javax.swing.ImageIcon(getClass().getResource("/ntnusubsea/gui/Images/switch-off.gif"))); // NOI18N
    lightSwitch.setBorder(null);
    lightSwitch.setContentAreaFilled(false);
    lightSwitch.setCursor(new java.awt.Cursor(java.awt.Cursor.HAND_CURSOR));
    lightSwitch.setDisabledIcon(new javax.swing.ImageIcon(getClass().getResource("/ntnusubsea/gui/Images/switch-off.gif"))); // NOI18N
    lightSwitch.setFocusPainted(false);
    lightSwitch.setFocusable(false);
    lightSwitch.setSelectedIcon(new javax.swing.ImageIcon(getClass().getResource("/ntnusubsea/gui/Images/switch-on.gif"))); // NOI18N
    lightSwitch.addActionListener(new java.awt.event.ActionListener() {
        public void actionPerformed(java.awt.event.ActionEvent evt) {
            lightSwitchActionPerformed(evt);
        }
    });

    ImageIcon myimage = new javax.swing.ImageIcon(getClass().getResource("/ntnusubsea/gui/Images/light-bulb.png"));
    Image img = myimage.getImage();
    Image img2 = img.getScaledInstance(40, 40, Image.SCALE_SMOOTH);
    ImageIcon imgIcon = new ImageIcon(img2);
    lightSwitch_lbl.setIcon(imgIcon);
    lightSwitch_lbl.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
    lightSwitch_lbl.setEnabled(false);
    lightSwitch_lbl.setMaximumSize(new java.awt.Dimension(63, 100));
    lightSwitch_lbl.setMinimumSize(new java.awt.Dimension(63, 100));
```

```
781
782     lightSlider.setBackground(new java.awt.Color(39, 44, 50));
783     lightSlider.setMaximum(24);
784     lightSlider.setMinimum(19);
785     lightSlider.setCursor(new java.awt.Cursor(java.awt.Cursor.HAND_CURSOR));
786     lightSlider.setEnabled(false);
787     lightSlider.addMouseListener(new java.awt.event.MouseAdapter() {
788         public void mouseReleased(java.awt.event.MouseEvent evt) {
789             lightSliderMouseReleased(evt);
790         }
791     });
792
793     lightSwitchBlueLED.setBackground(new java.awt.Color(39, 44, 50));
794     lightSwitchBlueLED.setForeground(new java.awt.Color(39, 44, 50));
795     lightSwitchBlueLED.setIcon(new javax.swing.ImageIcon(getClass().getResource("/ntnusubsea/gui/Images/switch-off.gif"))); // NOI18N
796     lightSwitchBlueLED.setBorder(null);
797     lightSwitchBlueLED.setContentAreaFilled(false);
798     lightSwitchBlueLED.setCursor(new java.awt.Cursor(java.awt.Cursor.HAND_CURSOR));
799     lightSwitchBlueLED.setDisabledIcon(new javax.swing.ImageIcon(getClass().getResource("/ntnusubsea/gui/Images/switch-off.gif"))); // NOI18N
800     lightSwitchBlueLED.setFocusPainted(false);
801     lightSwitchBlueLED.setFocusable(false);
802     lightSwitchBlueLED.setSelectedIcon(new javax.swing.ImageIcon(getClass().getResource("/ntnusubsea/gui/Images/switch-on.gif"))); // NOI18N
803     lightSwitchBlueLED.addActionListener(new java.awt.event.ActionListener() {
804         public void actionPerformed(java.awt.event.ActionEvent evt) {
805             lightSwitchBlueLEDActionPerformed(evt);
806         }
807     });
808
809     jLabel6.setForeground(new java.awt.Color(255, 255, 255));
810     jLabel6.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
811     jLabel6.setText("Blue LEDs");
812
813     javax.swing.GroupLayout lightPanelLayout = new javax.swing.GroupLayout(lightPanel);
814     lightPanel.setLayout(lightPanelLayout);
815     lightPanelLayout.setHorizontalGroup(
816         lightPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
817         .addGroup(lightPanelLayout.createSequentialGroup()
818             .addGap(0, 0, 0)
819             .addComponent(filler2, javax.swing.GroupLayout.PREFERRED_SIZE, 0, javax.swing.GroupLayout.PREFERRED_SIZE)
820             .addGroup(lightPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
821                 .addGroup(lightPanelLayout.createSequentialGroup()
822                     .addGroup(lightPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING)
823                         .addGroup(lightPanelLayout.createSequentialGroup()
824                             .addGap(0, 0, Short.MAX_VALUE)
825                             .addComponent(lightSlider, javax.swing.GroupLayout.PREFERRED_SIZE, 100, javax.swing.GroupLayout.PREFERRED_SIZE))
826                         .addGroup(lightPanelLayout.createSequentialGroup()
827                             .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED, javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALU
828                             .addGroup(lightPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING)
829                                 .addComponent(lightSwitchBlueLED, javax.swing.GroupLayout.PREFERRED_SIZE, 100, javax.swing.GroupLayout.PREFERRED_SIZE
830                                 .addComponent(lightSwitch, javax.swing.GroupLayout.PREFERRED_SIZE, 100, javax.swing.GroupLayout.PREFERRED_SIZE)
831                                 .addComponent(jLabel6, javax.swing.GroupLayout.PREFERRED_SIZE, 100, javax.swing.GroupLayout.PREFERRED_SIZE))))
832                     .addGap(46, 46, 46))
833                 .addComponent(lightHeader, javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
834                 .addComponent(jSeparator5)
835                 .addGroup(lightPanelLayout.createSequentialGroup()
836                     .addGap(60, 60, 60)
837                     .addComponent(lightSwitch_lbl, javax.swing.GroupLayout.PREFERRED_SIZE, 72, javax.swing.GroupLayout.PREFERRED_SIZE)
838                     .addContainerGap(60, Short.MAX_VALUE))))
839     );
840     lightPanelLayout.setVerticalGroup(
841         lightPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
842         .addGroup(lightPanelLayout.createSequentialGroup()
843             .addComponent(lightHeader, javax.swing.GroupLayout.PREFERRED_SIZE, 27, javax.swing.GroupLayout.PREFERRED_SIZE)
844             .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
845             .addComponent(jSeparator5, javax.swing.GroupLayout.PREFERRED_SIZE, 10, javax.swing.GroupLayout.PREFERRED_SIZE)
846             .addGap(5, 5, 5)
847             .addComponent(lightSwitch_lbl, javax.swing.GroupLayout.PREFERRED_SIZE, 40, javax.swing.GroupLayout.PREFERRED_SIZE)
848             .addGap(0, 0, Short.MAX_VALUE)
849             .addComponent(lightSlider, javax.swing.GroupLayout.PREFERRED_SIZE, 17, javax.swing.GroupLayout.PREFERRED_SIZE)
850             .addGap(0, 0, 0)
851             .addComponent(lightSwitch, javax.swing.GroupLayout.PREFERRED_SIZE, 39, javax.swing.GroupLayout.PREFERRED_SIZE)
852             .addGap(10, 10, 10)
853             .addComponent(jLabel6)
854             .addGap(1, 1, 1)
855             .addComponent(lightSwitchBlueLED, javax.swing.GroupLayout.PREFERRED_SIZE, 39, javax.swing.GroupLayout.PREFERRED_SIZE))
856         .addComponent(filler2, javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
857     );
858
859     emergencyPanel.setBackground(new java.awt.Color(39, 44, 50));
860     emergencyPanel.setMaximumSize(new java.awt.Dimension(153, 213));
861
862     emergencyHeader.setBackground(new java.awt.Color(28, 28, 28));
863     emergencyHeader.setFont(new java.awt.Font("Tahoma", 0, 14)); // NOI18N
864     emergencyHeader.setForeground(new java.awt.Color(255, 255, 255));
865     emergencyHeader.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
866     emergencyHeader.setText("Emergency surfacing");
867
```

```
868     emergencyStopButton.setBackground(new java.awt.Color(39, 44, 50));
869     emergencyStopButton.setForeground(new java.awt.Color(28, 28, 28));
870     ImageIcon egm_myimage = new javax.swing.ImageIcon(getClass().getResource("/ntnusubsea/gui/Images/emg-stop.gif"));
871     Image egm_img = egm_myimage.getImage();
872     Image egm_img2 = egm_img.getScaledInstance(100, 100, Image.SCALE_SMOOTH);
873     ImageIcon egm_imgIcon = new ImageIcon(egm_img2);
874     emergencyStopButton.setIcon(egm_imgIcon);
875     emergencyStopButton.setBorder(null);
876     emergencyStopButton.setCursor(new java.awt.Cursor(java.awt.Cursor.HAND_CURSOR));
877     emergencyStopButton.setEnabled(false);
878     emergencyStopButton.addActionListener(new java.awt.event.ActionListener() {
879         public void actionPerformed(java.awt.event.ActionEvent evt) {
880             emergencyStopButtonActionPerformed(evt);
881         }
882     });
883
884     jSeparator6.setBackground(new java.awt.Color(67, 72, 83));
885     jSeparator6.setForeground(new java.awt.Color(67, 72, 83));
886
887     javax.swing.GroupLayout emergencyPanelLayout = new javax.swing.GroupLayout(emergencyPanel);
888     emergencyPanel.setLayout(emergencyPanelLayout);
889     emergencyPanelLayout.setHorizontalGroup(
890         emergencyPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
891         .addGroup(emergencyPanelLayout.createSequentialGroup()
892             .addGroup(emergencyPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
893                 .addComponent(jSeparator6)
894                 .addGroup(javax.swing.GroupLayout.Alignment.TRAILING, emergencyPanelLayout.createSequentialGroup()
895                     .addGap(47, 47, 47)
896                     .addComponent(emergencyStopButton, javax.swing.GroupLayout.PREFERRED_SIZE, 101, javax.swing.GroupLayout.PREFERRED_SIZE)
897                     .addGap(0, 0, Short.MAX_VALUE))
898                 .addGroup(javax.swing.GroupLayout.Alignment.TRAILING, emergencyPanelLayout.createSequentialGroup()
899                     .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
900                     .addComponent(emergencyHeader, javax.swing.GroupLayout.PREFERRED_SIZE, 173, javax.swing.GroupLayout.PREFERRED_SIZE)
901                     .addGap(18, 18, 18)))
902             .addComponent(filler1, javax.swing.GroupLayout.PREFERRED_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFE
903             .addGap(0, 0, 0))
904     );
905     emergencyPanelLayout.setVerticalGroup(
906         emergencyPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
907         .addGroup(emergencyPanelLayout.createSequentialGroup()
908             .addComponent(emergencyHeader, javax.swing.GroupLayout.PREFERRED_SIZE, 27, javax.swing.GroupLayout.PREFERRED_SIZE)
909             .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
910             .addComponent(jSeparator6, javax.swing.GroupLayout.PREFERRED_SIZE, 10, javax.swing.GroupLayout.PREFERRED_SIZE)
911             .addGap(35, 35, 35)
912             .addComponent(emergencyStopButton, javax.swing.GroupLayout.PREFERRED_SIZE, 100, javax.swing.GroupLayout.PREFERRED_SIZE)
913             .addGap(35, 35, 35))
914         .addGroup(emergencyPanelLayout.createSequentialGroup()
915             .addContainerGap()
916             .addComponent(filler1, javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
917     );
918
919     cameraControlPanel.setBackground(new java.awt.Color(39, 44, 50));
920     cameraControlPanel.setMaximumSize(new java.awt.Dimension(190, 213));
921     cameraControlPanel.setPreferredSize(new java.awt.Dimension(190, 213));
922
923     delayTextField.setHorizontalAlignment(javax.swing.JTextField.CENTER);
924     delayTextField.setToolTipText("Time between each frame (0-99). - Press enter to send command.");
925     delayTextField.setEnabled(false);
926     delayTextField.addActionListener(new java.awt.event.ActionListener() {
927         public void actionPerformed(java.awt.event.ActionEvent evt) {
928             delayTextFieldActionPerformed(evt);
929         }
930     });
931
932     cameraHeader.setFont(new java.awt.Font("Tahoma", 0, 14)); // NOI18N
933     cameraHeader.setForeground(new java.awt.Color(255, 255, 255));
934     cameraHeader.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
935     cameraHeader.setText("Camera Control");
936     cameraHeader.setHorizontalTextPosition(javax.swing.SwingConstants.CENTER);
937
938     jSeparator9.setBackground(new java.awt.Color(67, 72, 83));
939     jSeparator9.setForeground(new java.awt.Color(67, 72, 83));
940
941     photoModeButton.setBackground(new java.awt.Color(39, 44, 50));
942     photoModeButton.setForeground(new java.awt.Color(39, 44, 50));
943     photoModeButton.setIcon(new javax.swing.ImageIcon(getClass().getResource("/ntnusubsea/gui/Images/switch-off.gif"))); // NOI18N
944     photoModeButton.setBorder(null);
945     photoModeButton.setContentAreaFilled(false);
946     photoModeButton.setCursor(new java.awt.Cursor(java.awt.Cursor.HAND_CURSOR));
947     photoModeButton.setDisabledIcon(new javax.swing.ImageIcon(getClass().getResource("/ntnusubsea/gui/Images/switch-off.gif"))); // NOI18N
948     photoModeButton.setEnabled(false);
949     photoModeButton.setFocusPainted(false);
950     photoModeButton.setFocusable(false);
951     photoModeButton.setSelectedIcon(new javax.swing.ImageIcon(getClass().getResource("/ntnusubsea/gui/Images/switch-on.gif"))); // NOI18N
952     photoModeButton.addActionListener(new java.awt.event.ActionListener() {
953         public void actionPerformed(java.awt.event.ActionEvent evt) {
954             photoModeButtonActionPerformed(evt);
```

```
955         }
956     });
957 
958     jLabel2.setForeground(new java.awt.Color(255, 255, 255));
959     jLabel2.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
960     jLabel2.setText("Photo Mode");
961 
962     jLabel3.setForeground(new java.awt.Color(255, 255, 255));
963     jLabel3.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
964     jLabel3.setText("Delay:");
965 
966     cameraPitchSlider.setBackground(new java.awt.Color(39, 44, 50));
967     cameraPitchSlider.setMaximum(75);
968     cameraPitchSlider.setMinimum(50);
969     cameraPitchSlider.setValue(57);
970     cameraPitchSlider.setCursor(new java.awt.Cursor(java.awt.Cursor.HAND_CURSOR));
971     cameraPitchSlider.setEnabled(false);
972     cameraPitchSlider.addMouseListener(new java.awt.event.MouseAdapter() {
973         public void mouseReleased(java.awt.event.MouseEvent evt) {
974             cameraPitchSliderMouseReleased(evt);
975         }
976     });
977 
978     jLabel4.setForeground(new java.awt.Color(255, 255, 255));
979     jLabel4.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
980     jLabel4.setText("Camera Pitch:");
981 
982     cameraPitchTextField.setHorizontalAlignment(javax.swing.JTextField.CENTER);
983     cameraPitchTextField.setToolTipText("Camera Pitch (50-75). - Press enter to send command.");
984     cameraPitchTextField.setEnabled(false);
985     cameraPitchTextField.addActionListener(new java.awt.event.ActionListener() {
986         public void actionPerformed(java.awt.event.ActionEvent evt) {
987             cameraPitchTextFieldActionPerformed(evt);
988         }
989     });
990 
991     cameraPitchLabel.setForeground(new java.awt.Color(255, 255, 255));
992     cameraPitchLabel.setText("57");
993 
994     photoModeDelayLabel.setForeground(new java.awt.Color(255, 255, 255));
995     photoModeDelayLabel.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
996     photoModeDelayLabel.setText("0.00 s");
997 
998     getPhotosButton.setBackground(new java.awt.Color(39, 44, 50));
999     getPhotosButton.setFont(new java.awt.Font("Dialog", 1, 10)); // NOI18N
1000     getPhotosButton.setForeground(new java.awt.Color(255, 255, 255));
1001     getPhotosButton.setText("Get IMGs");
1002     getPhotosButton.setToolTipText("Retrieves the photos from the ROV and saves it to C:/TowedROV/ROV_Photos");
1003     getPhotosButton.setCursor(new java.awt.Cursor(java.awt.Cursor.HAND_CURSOR));
1004     getPhotosButton.setEnabled(false);
1005     getPhotosButton.setFocusPainted(false);
1006     getPhotosButton.addActionListener(new java.awt.event.ActionListener() {
1007         public void actionPerformed(java.awt.event.ActionEvent evt) {
1008             getPhotosButtonActionPerformed(evt);
1009         }
1010     });
1011 
1012     photoModeDelay_FB_Label.setForeground(new java.awt.Color(255, 255, 255));
1013     photoModeDelay_FB_Label.setText("1.00 s");
1014 
1015     clearImagesButton.setBackground(new java.awt.Color(39, 44, 50));
1016     clearImagesButton.setFont(new java.awt.Font("Dialog", 1, 10)); // NOI18N
1017     clearImagesButton.setForeground(new java.awt.Color(255, 255, 255));
1018     clearImagesButton.setToolTipText("Clears the image folder on the ROV RPi.");
1019     clearImagesButton.setCursor(new java.awt.Cursor(java.awt.Cursor.HAND_CURSOR));
1020     clearImagesButton.setEnabled(false);
1021     clearImagesButton.setFocusPainted(false);
1022     clearImagesButton.setLabel("CLR IMGs");
1023     clearImagesButton.addActionListener(new java.awt.event.ActionListener() {
1024         public void actionPerformed(java.awt.event.ActionEvent evt) {
1025             clearImagesButtonActionPerformed(evt);
1026         }
1027     });
1028 
1029     imageNumberLabel.setForeground(new java.awt.Color(255, 255, 255));
1030     imageNumberLabel.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
1031     imageNumberLabel.setText("0 / 1000");
1032 
1033     javax.swing.GroupLayout cameraControlPanelLayout = new javax.swing.GroupLayout(cameraControlPanel);
1034     cameraControlPanel.setLayout(cameraControlPanelLayout);
1035     cameraControlPanelLayout.setHorizontalGroup(
1036         cameraControlPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
1037         .addComponent(jSeparator9)
1038         .addGroup(cameraControlPanelLayout.createSequentialGroup()
1039             .addGap(69, 69, 69)
1040             .addComponent(jLabel2, javax.swing.GroupLayout.PREFERRED_SIZE, 100, javax.swing.GroupLayout.PREFERRED_SIZE)
1041             .addGap(0, 0, Short.MAX_VALUE))
```

```
1042            .addComponent(cameraHeader, javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
1043            .addGroup(cameraControlPanelLayout.createSequentialGroup()
1044                .addContainerGap(42, Short.MAX_VALUE)
1045                .addGroup(cameraControlPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
1046                    .addGroup(cameraControlPanelLayout.createSequentialGroup()
1047                        .addGroup(cameraControlPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING)
1048                            .addGroup(cameraControlPanelLayout.createSequentialGroup()
1049                                .addComponent(jLabel3, javax.swing.GroupLayout.PREFERRED_SIZE, 36, javax.swing.GroupLayout.PREFERRED_SIZE)
1050                                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
1051                                .addComponent(photoModeDelayLabel, javax.swing.GroupLayout.PREFERRED_SIZE, 43, javax.swing.GroupLayout.PREFERRED_SIZE)
1052                                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
1053                                .addComponent(delayTextField, javax.swing.GroupLayout.PREFERRED_SIZE, 34, javax.swing.GroupLayout.PREFERRED_SIZE))
1054                            .addComponent(jLabel4, javax.swing.GroupLayout.PREFERRED_SIZE, 112, javax.swing.GroupLayout.PREFERRED_SIZE))
1055                        .addGroup(cameraControlPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
1056                            .addGroup(cameraControlPanelLayout.createSequentialGroup()
1057                                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
1058                                .addComponent(cameraPitchLabel, javax.swing.GroupLayout.PREFERRED_SIZE, 35, javax.swing.GroupLayout.PREFERRED_SIZE)
1059                                .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
1060                            .addGroup(javax.swing.GroupLayout.Alignment.TRAILING, cameraControlPanelLayout.createSequentialGroup()
1061                                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED, 11, Short.MAX_VALUE)
1062                                .addComponent(photoModeDelay_FB_Label, javax.swing.GroupLayout.PREFERRED_SIZE, 56, javax.swing.GroupLayout.PREFERRED_S
1063                                .addContainerGap()))))
1064                    .addGroup(cameraControlPanelLayout.createSequentialGroup()
1065                        .addGroup(cameraControlPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING)
1066                            .addGroup(cameraControlPanelLayout.createSequentialGroup()
1067                                .addComponent(cameraPitchSlider, javax.swing.GroupLayout.PREFERRED_SIZE, 100, javax.swing.GroupLayout.PREFERRED_SIZE)
1068                                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
1069                                .addComponent(cameraPitchTextField, javax.swing.GroupLayout.PREFERRED_SIZE, 40, javax.swing.GroupLayout.PREFERRED_SIZE)
1070                                .addGap(6, 6, 6))
1071                            .addGroup(cameraControlPanelLayout.createSequentialGroup()
1072                                .addGroup(cameraControlPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING, false)
1073                                    .addComponent(photoModeButton, javax.swing.GroupLayout.PREFERRED_SIZE, 72, javax.swing.GroupLayout.PREFERRED_SIZE)
1074                                    .addComponent(imageNumberLabel, javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX
1075                                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
1076                                .addGroup(cameraControlPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING, false)
1077                                    .addComponent(clearImagesButton, javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_
1078                                    .addComponent(getPhotosButton, javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_V
1079                        .addContainerGap(41, Short.MAX_VALUE))))
1080        );
1081        cameraControlPanelLayout.setVerticalGroup(
1082            cameraControlPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
1083            .addGroup(cameraControlPanelLayout.createSequentialGroup()
1084                .addComponent(cameraHeader, javax.swing.GroupLayout.PREFERRED_SIZE, 27, javax.swing.GroupLayout.PREFERRED_SIZE)
1085                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
1086                .addComponent(jSeparator9, javax.swing.GroupLayout.PREFERRED_SIZE, 10, javax.swing.GroupLayout.PREFERRED_SIZE)
1087                .addGap(0, 0, 0)
1088                .addGroup(cameraControlPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
1089                    .addComponent(jLabel3, javax.swing.GroupLayout.PREFERRED_SIZE, 30, javax.swing.GroupLayout.PREFERRED_SIZE)
1090                    .addComponent(delayTextField, javax.swing.GroupLayout.PREFERRED_SIZE, 22, javax.swing.GroupLayout.PREFERRED_SIZE)
1091                    .addComponent(photoModeDelayLabel)
1092                    .addComponent(photoModeDelay_FB_Label))
1093                .addGap(0, 0, 0)
1094                .addGroup(cameraControlPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
1095                    .addComponent(jLabel4)
1096                    .addComponent(cameraPitchLabel))
1097                .addGap(2, 2, 2)
1098                .addGroup(cameraControlPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING)
1099                    .addComponent(cameraPitchTextField, javax.swing.GroupLayout.PREFERRED_SIZE, 22, javax.swing.GroupLayout.PREFERRED_SIZE)
1100                    .addComponent(cameraPitchSlider, javax.swing.GroupLayout.PREFERRED_SIZE, 17, javax.swing.GroupLayout.PREFERRED_SIZE))
1101                .addGap(10, 10, 10)
1102                .addComponent(jLabel2)
1103                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
1104                .addGroup(cameraControlPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING, false)
1105                    .addGroup(cameraControlPanelLayout.createSequentialGroup()
1106                        .addComponent(getPhotosButton, javax.swing.GroupLayout.PREFERRED_SIZE, 29, javax.swing.GroupLayout.PREFERRED_SIZE)
1107                        .addGap(0, 0, 0)
1108                        .addComponent(clearImagesButton, javax.swing.GroupLayout.PREFERRED_SIZE, 29, javax.swing.GroupLayout.PREFERRED_SIZE))
1109                    .addGroup(cameraControlPanelLayout.createSequentialGroup()
1110                        .addComponent(photoModeButton, javax.swing.GroupLayout.PREFERRED_SIZE, 39, javax.swing.GroupLayout.PREFERRED_SIZE)
1111                        .addGap(0, 0, Short.MAX_VALUE)
1112                        .addComponent(imageNumberLabel)))
1113                .addGap(246, 246, 246))
1114        );
1115
1116        delayTextField.getAccessibleContext().setAccessibleDescription("Time between each frame");
1117
1118        jSeparator2.setBackground(new java.awt.Color(67, 72, 83));
1119        jSeparator2.setForeground(new java.awt.Color(67, 72, 83));
1120        jSeparator2.setOrientation(javax.swing.SwingConstants.VERTICAL);
1121
1122        jSeparator8.setBackground(new java.awt.Color(67, 72, 83));
1123        jSeparator8.setForeground(new java.awt.Color(67, 72, 83));
1124        jSeparator8.setOrientation(javax.swing.SwingConstants.VERTICAL);
1125
1126        jSeparator1.setBackground(new java.awt.Color(67, 72, 83));
1127        jSeparator1.setForeground(new java.awt.Color(67, 72, 83));
1128        jSeparator1.setOrientation(javax.swing.SwingConstants.VERTICAL);
```

```
1129
1130        javax.swing.GroupLayout controlPanelLayout = new javax.swing.GroupLayout(controlPanel);
1131        controlPanel.setLayout(controlPanelLayout);
1132        controlPanelLayout.setHorizontalGroup(
1133            controlPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
1134            .addGroup(controlPanelLayout.createSequentialGroup()
1135                .addContainerGap()
1136                .addComponent(depthPanel, javax.swing.GroupLayout.PREFERRED_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PI
1137                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
1138                .addComponent(jSeparator1, javax.swing.GroupLayout.PREFERRED_SIZE, 2, javax.swing.GroupLayout.PREFERRED_SIZE)
1139                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
1140                .addComponent(lightPanel, javax.swing.GroupLayout.PREFERRED_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PR
1141                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
1142                .addComponent(jSeparator8, javax.swing.GroupLayout.PREFERRED_SIZE, 2, javax.swing.GroupLayout.PREFERRED_SIZE)
1143                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
1144                .addComponent(cameraControlPanel, javax.swing.GroupLayout.PREFERRED_SIZE, 240, javax.swing.GroupLayout.PREFERRED_SIZE)
1145                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
1146                .addComponent(jSeparator2, javax.swing.GroupLayout.PREFERRED_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PA
1147                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
1148                .addComponent(emergencyPanel, javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
1149                .addContainerGap())
1150        );
1151        controlPanelLayout.setVerticalGroup(
1152            controlPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
1153            .addComponent(jSeparator2, javax.swing.GroupLayout.Alignment.TRAILING)
1154            .addComponent(jSeparator8, javax.swing.GroupLayout.Alignment.TRAILING)
1155            .addComponent(jSeparator1, javax.swing.GroupLayout.Alignment.TRAILING)
1156            .addGroup(controlPanelLayout.createSequentialGroup()
1157                .addGroup(controlPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
1158                    .addComponent(cameraControlPanel, javax.swing.GroupLayout.PREFERRED_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.Grou
1159                    .addComponent(emergencyPanel, javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayou
1160                    .addComponent(depthPanel, javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PR
1161                .addGap(0, 1, Short.MAX_VALUE))
1162            .addComponent(lightPanel, javax.swing.GroupLayout.Alignment.TRAILING, javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEF
1163        );
1164
1165        infoPanel.setBackground(new java.awt.Color(39, 44, 50));
1166        infoPanel.setForeground(new java.awt.Color(39, 44, 50));
1167        infoPanel.setMaximumSize(new java.awt.Dimension(455, 509));
1168        infoPanel.setMinimumSize(new java.awt.Dimension(455, 509));
1169
1170        actuatorPanel1.setBackground(new java.awt.Color(42, 48, 57));
1171        actuatorPanel1.setBorder(new javax.swing.border.LineBorder(new java.awt.Color(45, 53, 62), 1, true));
1172        actuatorPanel1.setForeground(new java.awt.Color(255, 255, 255));
1173
1174        actuatorHeader1.setBackground(new java.awt.Color(42, 48, 57));
1175        actuatorHeader1.setFont(new java.awt.Font("Tahoma", 0, 14)); // NOI18N
1176        actuatorHeader1.setForeground(new java.awt.Color(255, 255, 255));
1177        actuatorHeader1.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
1178        actuatorHeader1.setText("PS Actuator Position");
1179
1180        actuatorPSPosBar.setBackground(new java.awt.Color(42, 48, 57));
1181        actuatorPSPosBar.setForeground(new java.awt.Color(97, 184, 114));
1182        actuatorPSPosBar.setMaximum(254);
1183        actuatorPSPosBar.setValue(85);
1184        actuatorPSPosBar.setBorder(null);
1185        actuatorPSPosBar.addChangeListener(new javax.swing.event.ChangeListener() {
1186            public void stateChanged(javax.swing.event.ChangeEvent evt) {
1187                actuatorPSPosBarStateChanged(evt);
1188            }
1189        });
1190
1191        warningLabel1.setBackground(new java.awt.Color(42, 48, 57));
1192        warningLabel1.setFont(new java.awt.Font("Tahoma", 1, 12)); // NOI18N
1193        warningLabel1.setForeground(new java.awt.Color(255, 255, 255));
1194        warningLabel1.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
1195        warningLabel1.setHorizontalTextPosition(javax.swing.SwingConstants.CENTER);
1196        warningLabel1.setOpaque(true);
1197
1198        javax.swing.GroupLayout actuatorPanel1Layout = new javax.swing.GroupLayout(actuatorPanel1);
1199        actuatorPanel1.setLayout(actuatorPanel1Layout);
1200        actuatorPanel1Layout.setHorizontalGroup(
1201            actuatorPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
1202            .addComponent(actuatorHeader1, javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
1203            .addComponent(warningLabel1, javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
1204            .addComponent(actuatorPSPosBar, javax.swing.GroupLayout.Alignment.TRAILING, javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLay
1205        );
1206        actuatorPanel1Layout.setVerticalGroup(
1207            actuatorPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
1208            .addGroup(actuatorPanel1Layout.createSequentialGroup()
1209                .addComponent(actuatorHeader1, javax.swing.GroupLayout.PREFERRED_SIZE, 29, javax.swing.GroupLayout.PREFERRED_SIZE)
1210                .addGap(0, 0, 0)
1211                .addComponent(actuatorPSPosBar, javax.swing.GroupLayout.PREFERRED_SIZE, 30, javax.swing.GroupLayout.PREFERRED_SIZE)
1212                .addGap(0, 0, 0)
1213                .addComponent(warningLabel1, javax.swing.GroupLayout.PREFERRED_SIZE, 30, javax.swing.GroupLayout.PREFERRED_SIZE))
1214        );
1215
```

```
1216    actuatorPanel2.setBackground(new java.awt.Color(42, 48, 57));
1217    actuatorPanel2.setBorder(new javax.swing.border.LineBorder(new java.awt.Color(45, 53, 62), 1, true));
1218    actuatorPanel2.setForeground(new java.awt.Color(255, 255, 255));
1219
1220    actuatorHeader2.setBackground(new java.awt.Color(42, 48, 57));
1221    actuatorHeader2.setFont(new java.awt.Font("Tahoma", 0, 14)); // NOI18N
1222    actuatorHeader2.setForeground(new java.awt.Color(255, 255, 255));
1223    actuatorHeader2.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
1224    actuatorHeader2.setText("SB Actuator Position");
1225
1226    actuatorSBPosBar.setBackground(new java.awt.Color(42, 48, 57));
1227    actuatorSBPosBar.setForeground(new java.awt.Color(97, 184, 114));
1228    actuatorSBPosBar.setMaximum(254);
1229    actuatorSBPosBar.setToolTipText("");
1230    actuatorSBPosBar.setValue(85);
1231    actuatorSBPosBar.setBorder(null);
1232    actuatorSBPosBar.addChangeListener(new javax.swing.event.ChangeListener() {
1233        public void stateChanged(javax.swing.event.ChangeEvent evt) {
1234            actuatorSBPosBarStateChanged(evt);
1235        }
1236    });
1237
1238    warningLabel2.setBackground(new java.awt.Color(42, 48, 57));
1239    warningLabel2.setFont(new java.awt.Font("Tahoma", 1, 12)); // NOI18N
1240    warningLabel2.setForeground(new java.awt.Color(255, 255, 255));
1241    warningLabel2.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
1242    warningLabel2.setHorizontalTextPosition(javax.swing.SwingConstants.CENTER);
1243
1244    javax.swing.GroupLayout actuatorPanel2Layout = new javax.swing.GroupLayout(actuatorPanel2);
1245    actuatorPanel2.setLayout(actuatorPanel2Layout);
1246    actuatorPanel2Layout.setHorizontalGroup(
1247        actuatorPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
1248        .addComponent(actuatorHeader2, javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
1249        .addComponent(actuatorSBPosBar, javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
1250        .addComponent(warningLabel2, javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
1251    );
1252    actuatorPanel2Layout.setVerticalGroup(
1253        actuatorPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
1254        .addGroup(actuatorPanel2Layout.createSequentialGroup()
1255            .addComponent(actuatorHeader2, javax.swing.GroupLayout.PREFERRED_SIZE, 29, javax.swing.GroupLayout.PREFERRED_SIZE)
1256            .addGap(0, 0, 0)
1257            .addComponent(actuatorSBPosBar, javax.swing.GroupLayout.PREFERRED_SIZE, 30, javax.swing.GroupLayout.PREFERRED_SIZE)
1258            .addGap(0, 0, 0)
1259            .addComponent(warningLabel2, javax.swing.GroupLayout.PREFERRED_SIZE, 30, javax.swing.GroupLayout.PREFERRED_SIZE))
1260    );
1261
1262    jPanel2.setBackground(new java.awt.Color(39, 44, 50));
1263
1264    jPanel3.setBackground(new java.awt.Color(39, 44, 50));
1265    jPanel3.setBorder(new javax.swing.border.LineBorder(new java.awt.Color(45, 53, 62), 1, true));
1266
1267    jLabel7.setBackground(new java.awt.Color(42, 48, 57));
1268    jLabel7.setFont(new java.awt.Font("Tahoma", 1, 14)); // NOI18N
1269    jLabel7.setForeground(new java.awt.Color(255, 255, 255));
1270    jLabel7.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
1271    jLabel7.setText("ROV");
1272    jLabel7.setHorizontalTextPosition(javax.swing.SwingConstants.CENTER);
1273    jLabel7.setOpaque(true);
1274
1275    wingLabel.setBackground(new java.awt.Color(39, 46, 54));
1276    wingLabel.setFont(new java.awt.Font("Tahoma", 0, 14)); // NOI18N
1277    wingLabel.setForeground(new java.awt.Color(255, 255, 255));
1278    wingLabel.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
1279    wingLabel.setText("Wing Angle: 0.1");
1280    wingLabel.setHorizontalTextPosition(javax.swing.SwingConstants.CENTER);
1281    wingLabel.setOpaque(true);
1282
1283    pitchLabel.setBackground(new java.awt.Color(39, 46, 54));
1284    pitchLabel.setFont(new java.awt.Font("Tahoma", 0, 14)); // NOI18N
1285    pitchLabel.setForeground(new java.awt.Color(255, 255, 255));
1286    pitchLabel.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
1287    pitchLabel.setText("Pitch Angle: 0.1");
1288    pitchLabel.setHorizontalTextPosition(javax.swing.SwingConstants.CENTER);
1289    pitchLabel.setOpaque(true);
1290
1291    rollLabel.setBackground(new java.awt.Color(39, 46, 54));
1292    rollLabel.setFont(new java.awt.Font("Tahoma", 0, 14)); // NOI18N
1293    rollLabel.setForeground(new java.awt.Color(255, 255, 255));
1294    rollLabel.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
1295    rollLabel.setText("Roll Angle: 0.1");
1296    rollLabel.setHorizontalTextPosition(javax.swing.SwingConstants.CENTER);
1297    rollLabel.setMinimumSize(new java.awt.Dimension(140, 110));
1298    rollLabel.setOpaque(true);
1299
1300    jLabel9.setBackground(new java.awt.Color(39, 46, 54));
1301    jLabel9.setOpaque(true);
1302
```

```
1303        jLabel15.setBackground(new java.awt.Color(39, 46, 54));
1304        jLabel15.setOpaque(true);
1305
1306        actuatorPSPosLabel.setBackground(new java.awt.Color(39, 46, 54));
1307        actuatorPSPosLabel.setFont(new java.awt.Font("Tahoma", 0, 14)); // NOI18N
1308        actuatorPSPosLabel.setForeground(new java.awt.Color(255, 255, 255));
1309        actuatorPSPosLabel.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
1310        actuatorPSPosLabel.setText("PS Actuator Pos: 0.1");
1311        actuatorPSPosLabel.setHorizontalTextPosition(javax.swing.SwingConstants.CENTER);
1312        actuatorPSPosLabel.setOpaque(true);
1313
1314        actuatorSBPosLabel.setBackground(new java.awt.Color(39, 46, 54));
1315        actuatorSBPosLabel.setFont(new java.awt.Font("Tahoma", 0, 14)); // NOI18N
1316        actuatorSBPosLabel.setForeground(new java.awt.Color(255, 255, 255));
1317        actuatorSBPosLabel.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
1318        actuatorSBPosLabel.setText("SB Actuator Pos: 0.1");
1319        actuatorSBPosLabel.setHorizontalTextPosition(javax.swing.SwingConstants.CENTER);
1320        actuatorSBPosLabel.setOpaque(true);
1321
1322        i2cErrorLabel.setBackground(new java.awt.Color(39, 46, 54));
1323        i2cErrorLabel.setFont(new java.awt.Font("Tahoma", 0, 14)); // NOI18N
1324        i2cErrorLabel.setForeground(new java.awt.Color(255, 255, 255));
1325        i2cErrorLabel.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
1326        i2cErrorLabel.setText("I2C: OK");
1327        i2cErrorLabel.setToolTipText("");
1328        i2cErrorLabel.setHorizontalTextPosition(javax.swing.SwingConstants.CENTER);
1329        i2cErrorLabel.setOpaque(true);
1330
1331        javax.swing.GroupLayout jPanel3Layout = new javax.swing.GroupLayout(jPanel3);
1332        jPanel3.setLayout(jPanel3Layout);
1333        jPanel3Layout.setHorizontalGroup(
1334            jPanel3Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
1335            .addComponent(jLabel9, javax.swing.GroupLayout.Alignment.TRAILING, javax.swing.GroupLayout.PREFERRED_SIZE, 164, javax.swing.GroupLayou
1336            .addGroup(javax.swing.GroupLayout.Alignment.TRAILING, jPanel3Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
1337                .addGroup(jPanel3Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING, false)
1338                    .addComponent(actuatorSBPosLabel, javax.swing.GroupLayout.PREFERRED_SIZE, 170, javax.swing.GroupLayout.PREFERRED_SIZE)
1339                    .addComponent(actuatorPSPosLabel, javax.swing.GroupLayout.PREFERRED_SIZE, 170, javax.swing.GroupLayout.PREFERRED_SIZE)
1340                    .addComponent(i2cErrorLabel, javax.swing.GroupLayout.Alignment.LEADING, javax.swing.GroupLayout.PREFERRED_SIZE, 170, javax.swing.G
1341                .addComponent(wingLabel, javax.swing.GroupLayout.PREFERRED_SIZE, 170, javax.swing.GroupLayout.PREFERRED_SIZE)
1342                .addComponent(jLabel7, javax.swing.GroupLayout.PREFERRED_SIZE, 170, javax.swing.GroupLayout.PREFERRED_SIZE)
1343                .addGroup(jPanel3Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
1344                    .addComponent(rollLabel, javax.swing.GroupLayout.PREFERRED_SIZE, 170, javax.swing.GroupLayout.PREFERRED_SIZE)
1345                    .addComponent(pitchLabel, javax.swing.GroupLayout.Alignment.TRAILING, javax.swing.GroupLayout.PREFERRED_SIZE, 170, javax.swing.Grou
1346                .addComponent(jLabel15, javax.swing.GroupLayout.PREFERRED_SIZE, 170, javax.swing.GroupLayout.PREFERRED_SIZE))
1347        );
1348        jPanel3Layout.setVerticalGroup(
1349            jPanel3Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
1350            .addGroup(jPanel3Layout.createSequentialGroup()
1351                .addGap(0, 0, 0)
1352                .addComponent(jLabel7, javax.swing.GroupLayout.PREFERRED_SIZE, 40, javax.swing.GroupLayout.PREFERRED_SIZE)
1353                .addGap(0, 0, 0)
1354                .addComponent(jLabel9, javax.swing.GroupLayout.PREFERRED_SIZE, 10, javax.swing.GroupLayout.PREFERRED_SIZE)
1355                .addGap(0, 0, 0)
1356                .addComponent(rollLabel, javax.swing.GroupLayout.PREFERRED_SIZE, 32, javax.swing.GroupLayout.PREFERRED_SIZE)
1357                .addGap(0, 0, 0)
1358                .addComponent(pitchLabel, javax.swing.GroupLayout.PREFERRED_SIZE, 32, javax.swing.GroupLayout.PREFERRED_SIZE)
1359                .addGap(0, 0, 0)
1360                .addComponent(wingLabel, javax.swing.GroupLayout.PREFERRED_SIZE, 32, javax.swing.GroupLayout.PREFERRED_SIZE)
1361                .addGap(0, 0, 0)
1362                .addComponent(actuatorPSPosLabel, javax.swing.GroupLayout.PREFERRED_SIZE, 32, javax.swing.GroupLayout.PREFERRED_SIZE)
1363                .addGap(0, 0, 0)
1364                .addComponent(actuatorSBPosLabel, javax.swing.GroupLayout.PREFERRED_SIZE, 32, javax.swing.GroupLayout.PREFERRED_SIZE)
1365                .addGap(0, 0, 0)
1366                .addComponent(i2cErrorLabel, javax.swing.GroupLayout.PREFERRED_SIZE, 32, javax.swing.GroupLayout.PREFERRED_SIZE)
1367                .addGap(0, 0, 0)
1368                .addComponent(jLabel15, javax.swing.GroupLayout.PREFERRED_SIZE, 10, javax.swing.GroupLayout.PREFERRED_SIZE)
1369                .addGap(0, 0, Short.MAX_VALUE))
1370        );
1371
1372        jPanel4.setBackground(new java.awt.Color(39, 44, 50));
1373        jPanel4.setBorder(new javax.swing.border.LineBorder(new java.awt.Color(45, 53, 62), 1, true));
1374
1375        seafloorDepthRovLabel.setBackground(new java.awt.Color(39, 46, 54));
1376        seafloorDepthRovLabel.setFont(new java.awt.Font("Tahoma", 0, 14)); // NOI18N
1377        seafloorDepthRovLabel.setForeground(new java.awt.Color(255, 255, 255));
1378        seafloorDepthRovLabel.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
1379        seafloorDepthRovLabel.setText("Beneath ROV: 0.1m");
1380        seafloorDepthRovLabel.setHorizontalTextPosition(javax.swing.SwingConstants.CENTER);
1381        seafloorDepthRovLabel.setOpaque(true);
1382
1383        rovDepthLabel.setBackground(new java.awt.Color(39, 46, 54));
1384        rovDepthLabel.setFont(new java.awt.Font("Tahoma", 0, 14)); // NOI18N
1385        rovDepthLabel.setForeground(new java.awt.Color(255, 255, 255));
1386        rovDepthLabel.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
1387        rovDepthLabel.setText("ROV Depth: 0.1m");
1388        rovDepthLabel.setHorizontalTextPosition(javax.swing.SwingConstants.CENTER);
1389        rovDepthLabel.setOpaque(true);
```

```
1390
1391    seafloorDepthBoatLabel.setBackground(new java.awt.Color(39, 46, 54));
1392    seafloorDepthBoatLabel.setFont(new java.awt.Font("Tahoma", 0, 14)); // NOI18N
1393    seafloorDepthBoatLabel.setForeground(new java.awt.Color(255, 255, 255));
1394    seafloorDepthBoatLabel.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
1395    seafloorDepthBoatLabel.setText("Beneath Boat: 0.1m");
1396    seafloorDepthBoatLabel.setHorizontalTextPosition(javax.swing.SwingConstants.CENTER);
1397    seafloorDepthBoatLabel.setOpaque(true);
1398
1399    jLabel8.setBackground(new java.awt.Color(42, 48, 57));
1400    jLabel8.setFont(new java.awt.Font("Tahoma", 1, 14)); // NOI18N
1401    jLabel8.setForeground(new java.awt.Color(255, 255, 255));
1402    jLabel8.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
1403    jLabel8.setText("Depth");
1404    jLabel8.setOpaque(true);
1405
1406    jLabel10.setBackground(new java.awt.Color(39, 46, 54));
1407    jLabel10.setOpaque(true);
1408
1409    jLabel16.setBackground(new java.awt.Color(39, 46, 54));
1410    jLabel16.setOpaque(true);
1411
1412    javax.swing.GroupLayout jPanel4Layout = new javax.swing.GroupLayout(jPanel4);
1413    jPanel4.setLayout(jPanel4Layout);
1414    jPanel4Layout.setHorizontalGroup(
1415        jPanel4Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
1416        .addGroup(jPanel4Layout.createSequentialGroup()
1417            .addGap(0, 0, 0)
1418            .addGroup(jPanel4Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
1419                .addGroup(jPanel4Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING, false)
1420                    .addComponent(jLabel8, javax.swing.GroupLayout.PREFERRED_SIZE, 170, javax.swing.GroupLayout.PREFERRED_SIZE)
1421                    .addComponent(seafloorDepthRovLabel, javax.swing.GroupLayout.PREFERRED_SIZE, 170, javax.swing.GroupLayout.PREFERRED_SIZE)
1422                    .addComponent(seafloorDepthBoatLabel, javax.swing.GroupLayout.PREFERRED_SIZE, 170, javax.swing.GroupLayout.PREFERRED_SIZE)
1423                    .addComponent(rovDepthLabel, javax.swing.GroupLayout.Alignment.LEADING, javax.swing.GroupLayout.PREFERRED_SIZE, 170, javax.swin
1424                .addComponent(jLabel10, javax.swing.GroupLayout.PREFERRED_SIZE, 170, javax.swing.GroupLayout.PREFERRED_SIZE)
1425                .addComponent(jLabel16, javax.swing.GroupLayout.PREFERRED_SIZE, 170, javax.swing.GroupLayout.PREFERRED_SIZE))
1426            .addGap(0, 0, Short.MAX_VALUE))
1427    );
1428    jPanel4Layout.setVerticalGroup(
1429        jPanel4Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
1430        .addGroup(javax.swing.GroupLayout.Alignment.TRAILING, jPanel4Layout.createSequentialGroup()
1431            .addGap(0, 0, 0)
1432            .addComponent(jLabel8, javax.swing.GroupLayout.PREFERRED_SIZE, 40, javax.swing.GroupLayout.PREFERRED_SIZE)
1433            .addGap(0, 0, 0)
1434            .addComponent(jLabel10, javax.swing.GroupLayout.PREFERRED_SIZE, 10, javax.swing.GroupLayout.PREFERRED_SIZE)
1435            .addGap(0, 0, 0)
1436            .addComponent(seafloorDepthBoatLabel, javax.swing.GroupLayout.PREFERRED_SIZE, 32, javax.swing.GroupLayout.PREFERRED_SIZE)
1437            .addGap(0, 0, 0)
1438            .addComponent(seafloorDepthRovLabel, javax.swing.GroupLayout.PREFERRED_SIZE, 32, javax.swing.GroupLayout.PREFERRED_SIZE)
1439            .addGap(0, 0, 0)
1440            .addComponent(rovDepthLabel, javax.swing.GroupLayout.PREFERRED_SIZE, 32, javax.swing.GroupLayout.PREFERRED_SIZE)
1441            .addGap(0, 0, Short.MAX_VALUE)
1442            .addComponent(jLabel16, javax.swing.GroupLayout.PREFERRED_SIZE, 10, javax.swing.GroupLayout.PREFERRED_SIZE)
1443            .addGap(0, 0, 0))
1444    );
1445
1446    jPanel5.setBackground(new java.awt.Color(39, 44, 50));
1447    jPanel5.setBorder(new javax.swing.border.LineBorder(new java.awt.Color(45, 53, 62), 1, true));
1448
1449    latitudeLabel.setBackground(new java.awt.Color(39, 46, 54));
1450    latitudeLabel.setFont(new java.awt.Font("Tahoma", 0, 14)); // NOI18N
1451    latitudeLabel.setForeground(new java.awt.Color(255, 255, 255));
1452    latitudeLabel.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
1453    latitudeLabel.setText("Latitude: 0.1");
1454    latitudeLabel.setHorizontalTextPosition(javax.swing.SwingConstants.CENTER);
1455    latitudeLabel.setOpaque(true);
1456
1457    longitudeLabel.setBackground(new java.awt.Color(39, 46, 54));
1458    longitudeLabel.setFont(new java.awt.Font("Tahoma", 0, 14)); // NOI18N
1459    longitudeLabel.setForeground(new java.awt.Color(255, 255, 255));
1460    longitudeLabel.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
1461    longitudeLabel.setText("Longitude: 0.1");
1462    longitudeLabel.setHorizontalTextPosition(javax.swing.SwingConstants.CENTER);
1463    longitudeLabel.setOpaque(true);
1464
1465    headingLabel.setBackground(new java.awt.Color(39, 46, 54));
1466    headingLabel.setFont(new java.awt.Font("Tahoma", 0, 14)); // NOI18N
1467    headingLabel.setForeground(new java.awt.Color(255, 255, 255));
1468    headingLabel.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
1469    headingLabel.setText("Heading: 0.1");
1470    headingLabel.setHorizontalTextPosition(javax.swing.SwingConstants.CENTER);
1471    headingLabel.setOpaque(true);
1472
1473    jLabel11.setBackground(new java.awt.Color(42, 48, 57));
1474    jLabel11.setFont(new java.awt.Font("Tahoma", 1, 14)); // NOI18N
1475    jLabel11.setForeground(new java.awt.Color(255, 255, 255));
1476    jLabel11.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
```

```
1477        jLabel11.setText("Position");
1478        jLabel11.setOpaque(true);
1479
1480        jLabel12.setBackground(new java.awt.Color(39, 46, 54));
1481        jLabel12.setOpaque(true);
1482
1483        jLabel17.setBackground(new java.awt.Color(39, 46, 54));
1484        jLabel17.setOpaque(true);
1485
1486        javax.swing.GroupLayout jPanel5Layout = new javax.swing.GroupLayout(jPanel5);
1487        jPanel5.setLayout(jPanel5Layout);
1488        jPanel5Layout.setHorizontalGroup(
1489            jPanel5Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
1490            .addGroup(jPanel5Layout.createSequentialGroup()
1491                .addGap(0, 0, 0)
1492                .addGroup(jPanel5Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
1493                    .addGroup(jPanel5Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING, false)
1494                        .addComponent(jLabel11, javax.swing.GroupLayout.PREFERRED_SIZE, 170, javax.swing.GroupLayout.PREFERRED_SIZE)
1495                        .addComponent(latitudeLabel, javax.swing.GroupLayout.PREFERRED_SIZE, 170, javax.swing.GroupLayout.PREFERRED_SIZE)
1496                        .addComponent(headingLabel, javax.swing.GroupLayout.PREFERRED_SIZE, 170, javax.swing.GroupLayout.PREFERRED_SIZE)
1497                        .addComponent(longitudeLabel, javax.swing.GroupLayout.Alignment.LEADING, javax.swing.GroupLayout.PREFERRED_SIZE, 170, javax.swin
1498                    .addComponent(jLabel12, javax.swing.GroupLayout.PREFERRED_SIZE, 170, javax.swing.GroupLayout.PREFERRED_SIZE)
1499                    .addComponent(jLabel17, javax.swing.GroupLayout.PREFERRED_SIZE, 170, javax.swing.GroupLayout.PREFERRED_SIZE))
1500                .addGap(0, 0, Short.MAX_VALUE))
1501        );
1502        jPanel5Layout.setVerticalGroup(
1503            jPanel5Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
1504            .addGroup(javax.swing.GroupLayout.Alignment.TRAILING, jPanel5Layout.createSequentialGroup()
1505                .addGap(0, 0, Short.MAX_VALUE)
1506                .addComponent(jLabel11, javax.swing.GroupLayout.PREFERRED_SIZE, 40, javax.swing.GroupLayout.PREFERRED_SIZE)
1507                .addGap(0, 0, 0)
1508                .addComponent(jLabel12, javax.swing.GroupLayout.PREFERRED_SIZE, 10, javax.swing.GroupLayout.PREFERRED_SIZE)
1509                .addGap(0, 0, 0)
1510                .addComponent(headingLabel, javax.swing.GroupLayout.PREFERRED_SIZE, 32, javax.swing.GroupLayout.PREFERRED_SIZE)
1511                .addGap(0, 0, 0)
1512                .addComponent(latitudeLabel, javax.swing.GroupLayout.PREFERRED_SIZE, 32, javax.swing.GroupLayout.PREFERRED_SIZE)
1513                .addGap(0, 0, 0)
1514                .addComponent(longitudeLabel, javax.swing.GroupLayout.PREFERRED_SIZE, 32, javax.swing.GroupLayout.PREFERRED_SIZE)
1515                .addGap(0, 0, 0)
1516                .addComponent(jLabel17, javax.swing.GroupLayout.PREFERRED_SIZE, 10, javax.swing.GroupLayout.PREFERRED_SIZE))
1517        );
1518
1519        jPanel7.setBackground(new java.awt.Color(39, 44, 50));
1520        jPanel7.setBorder(new javax.swing.border.LineBorder(new java.awt.Color(45, 53, 62), 1, true));
1521
1522        leakLabel.setBackground(new java.awt.Color(39, 46, 54));
1523        leakLabel.setFont(new java.awt.Font("Tahoma", 0, 14)); // NOI18N
1524        leakLabel.setForeground(new java.awt.Color(255, 255, 255));
1525        leakLabel.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
1526        leakLabel.setText("No leak detected");
1527        leakLabel.setHorizontalTextPosition(javax.swing.SwingConstants.CENTER);
1528        leakLabel.setOpaque(true);
1529
1530        jLabel19.setBackground(new java.awt.Color(42, 48, 57));
1531        jLabel19.setFont(new java.awt.Font("Tahoma", 1, 14)); // NOI18N
1532        jLabel19.setForeground(new java.awt.Color(255, 255, 255));
1533        jLabel19.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
1534        jLabel19.setText("Camera Housing");
1535        jLabel19.setOpaque(true);
1536
1537        jLabel20.setBackground(new java.awt.Color(39, 46, 54));
1538        jLabel20.setOpaque(true);
1539
1540        jLabel21.setBackground(new java.awt.Color(39, 46, 54));
1541        jLabel21.setOpaque(true);
1542
1543        outsideTempLabel.setBackground(new java.awt.Color(39, 46, 54));
1544        outsideTempLabel.setFont(new java.awt.Font("Tahoma", 0, 14)); // NOI18N
1545        outsideTempLabel.setForeground(new java.awt.Color(255, 255, 255));
1546        outsideTempLabel.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
1547        outsideTempLabel.setText("Outside Temp: 0.1");
1548        outsideTempLabel.setHorizontalTextPosition(javax.swing.SwingConstants.CENTER);
1549        outsideTempLabel.setMinimumSize(new java.awt.Dimension(140, 110));
1550        outsideTempLabel.setOpaque(true);
1551
1552        insideTempLabel.setBackground(new java.awt.Color(39, 46, 54));
1553        insideTempLabel.setFont(new java.awt.Font("Tahoma", 0, 14)); // NOI18N
1554        insideTempLabel.setForeground(new java.awt.Color(255, 255, 255));
1555        insideTempLabel.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
1556        insideTempLabel.setText("Inside Temp: 0.1");
1557        insideTempLabel.setHorizontalTextPosition(javax.swing.SwingConstants.CENTER);
1558        insideTempLabel.setOpaque(true);
1559
1560        humidityLabel.setBackground(new java.awt.Color(39, 46, 54));
1561        humidityLabel.setFont(new java.awt.Font("Tahoma", 0, 14)); // NOI18N
1562        humidityLabel.setForeground(new java.awt.Color(255, 255, 255));
1563        humidityLabel.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
```

```
1564        humidityLabel.setText("Humidity: 0.1");
1565        humidityLabel.setHorizontalTextPosition(javax.swing.SwingConstants.CENTER);
1566        humidityLabel.setOpaque(true);
1567
1568        pressureLabel.setBackground(new java.awt.Color(39, 46, 54));
1569        pressureLabel.setFont(new java.awt.Font("Tahoma", 0, 14)); // NOI18N
1570        pressureLabel.setForeground(new java.awt.Color(255, 255, 255));
1571        pressureLabel.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
1572        pressureLabel.setText("Pressure: 0.1");
1573        pressureLabel.setHorizontalTextPosition(javax.swing.SwingConstants.CENTER);
1574        pressureLabel.setOpaque(true);
1575
1576        actuatorSBPosLabel1.setBackground(new java.awt.Color(39, 46, 54));
1577        actuatorSBPosLabel1.setFont(new java.awt.Font("Tahoma", 0, 14)); // NOI18N
1578        actuatorSBPosLabel1.setForeground(new java.awt.Color(255, 255, 255));
1579        actuatorSBPosLabel1.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
1580        actuatorSBPosLabel1.setHorizontalTextPosition(javax.swing.SwingConstants.CENTER);
1581        actuatorSBPosLabel1.setOpaque(true);
1582
1583        javax.swing.GroupLayout jPanel7Layout = new javax.swing.GroupLayout(jPanel7);
1584        jPanel7.setLayout(jPanel7Layout);
1585        jPanel7Layout.setHorizontalGroup(
1586            jPanel7Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
1587            .addComponent(jLabel21, javax.swing.GroupLayout.PREFERRED_SIZE, 170, javax.swing.GroupLayout.PREFERRED_SIZE)
1588            .addGroup(jPanel7Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING, false)
1589                .addComponent(jLabel19, javax.swing.GroupLayout.PREFERRED_SIZE, 170, javax.swing.GroupLayout.PREFERRED_SIZE)
1590                .addComponent(leakLabel, javax.swing.GroupLayout.Alignment.LEADING, javax.swing.GroupLayout.PREFERRED_SIZE, 170, javax.swing.GroupL
1591            .addComponent(jLabel20, javax.swing.GroupLayout.PREFERRED_SIZE, 170, javax.swing.GroupLayout.PREFERRED_SIZE)
1592            .addComponent(pressureLabel, javax.swing.GroupLayout.PREFERRED_SIZE, 170, javax.swing.GroupLayout.PREFERRED_SIZE)
1593            .addComponent(humidityLabel, javax.swing.GroupLayout.PREFERRED_SIZE, 170, javax.swing.GroupLayout.PREFERRED_SIZE)
1594            .addGroup(jPanel7Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
1595                .addComponent(outsideTempLabel, javax.swing.GroupLayout.PREFERRED_SIZE, 170, javax.swing.GroupLayout.PREFERRED_SIZE)
1596                .addComponent(insideTempLabel, javax.swing.GroupLayout.Alignment.TRAILING, javax.swing.GroupLayout.PREFERRED_SIZE, 170, javax.swing.
1597            .addComponent(actuatorSBPosLabel1, javax.swing.GroupLayout.PREFERRED_SIZE, 170, javax.swing.GroupLayout.PREFERRED_SIZE)
1598        );
1599        jPanel7Layout.setVerticalGroup(
1600            jPanel7Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
1601            .addGroup(javax.swing.GroupLayout.Alignment.TRAILING, jPanel7Layout.createSequentialGroup()
1602                .addGap(0, 0, 0)
1603                .addComponent(jLabel19, javax.swing.GroupLayout.PREFERRED_SIZE, 40, javax.swing.GroupLayout.PREFERRED_SIZE)
1604                .addGap(0, 0, 0)
1605                .addComponent(jLabel20, javax.swing.GroupLayout.PREFERRED_SIZE, 10, javax.swing.GroupLayout.PREFERRED_SIZE)
1606                .addGap(0, 0, 0)
1607                .addComponent(outsideTempLabel, javax.swing.GroupLayout.PREFERRED_SIZE, 32, javax.swing.GroupLayout.PREFERRED_SIZE)
1608                .addGap(0, 0, 0)
1609                .addComponent(insideTempLabel, javax.swing.GroupLayout.PREFERRED_SIZE, 32, javax.swing.GroupLayout.PREFERRED_SIZE)
1610                .addGap(0, 0, 0)
1611                .addComponent(humidityLabel, javax.swing.GroupLayout.PREFERRED_SIZE, 32, javax.swing.GroupLayout.PREFERRED_SIZE)
1612                .addGap(0, 0, 0)
1613                .addComponent(pressureLabel, javax.swing.GroupLayout.PREFERRED_SIZE, 32, javax.swing.GroupLayout.PREFERRED_SIZE)
1614                .addGap(0, 0, 0)
1615                .addComponent(leakLabel, javax.swing.GroupLayout.PREFERRED_SIZE, 32, javax.swing.GroupLayout.PREFERRED_SIZE)
1616                .addGap(0, 0, 0)
1617                .addComponent(actuatorSBPosLabel1, javax.swing.GroupLayout.PREFERRED_SIZE, 32, javax.swing.GroupLayout.PREFERRED_SIZE)
1618                .addGap(0, 0, 0)
1619                .addComponent(jLabel21, javax.swing.GroupLayout.PREFERRED_SIZE, 10, javax.swing.GroupLayout.PREFERRED_SIZE))
1620        );
1621
1622        javax.swing.GroupLayout jPanel2Layout = new javax.swing.GroupLayout(jPanel2);
1623        jPanel2.setLayout(jPanel2Layout);
1624        jPanel2Layout.setHorizontalGroup(
1625            jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
1626            .addGroup(jPanel2Layout.createSequentialGroup()
1627                .addContainerGap(30, Short.MAX_VALUE)
1628                .addGroup(jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
1629                    .addComponent(jPanel5, javax.swing.GroupLayout.PREFERRED_SIZE, 170, javax.swing.GroupLayout.PREFERRED_SIZE)
1630                    .addComponent(jPanel3, javax.swing.GroupLayout.PREFERRED_SIZE, 170, javax.swing.GroupLayout.PREFERRED_SIZE))
1631                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED, 40, Short.MAX_VALUE)
1632                .addGroup(jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING, false)
1633                    .addComponent(jPanel4, javax.swing.GroupLayout.PREFERRED_SIZE, 170, javax.swing.GroupLayout.PREFERRED_SIZE)
1634                    .addComponent(jPanel7, javax.swing.GroupLayout.PREFERRED_SIZE, 170, javax.swing.GroupLayout.PREFERRED_SIZE))
1635                .addGap(26, 26, 26))
1636        );
1637        jPanel2Layout.setVerticalGroup(
1638            jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
1639            .addGroup(jPanel2Layout.createSequentialGroup()
1640                .addGap(60, 60, 60)
1641                .addGroup(jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING, false)
1642                    .addComponent(jPanel3, javax.swing.GroupLayout.PREFERRED_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PR
1643                    .addComponent(jPanel7, javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
1644                .addGap(50, 50, 50)
1645                .addGroup(jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
1646                    .addComponent(jPanel5, javax.swing.GroupLayout.PREFERRED_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PR
1647                    .addComponent(jPanel4, javax.swing.GroupLayout.PREFERRED_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PR
1648                .addContainerGap(64, Short.MAX_VALUE))
1649        );
1650
```

```
1651    javax.swing.GroupLayout infoPanelLayout = new javax.swing.GroupLayout(infoPanel);
1652    infoPanel.setLayout(infoPanelLayout);
1653    infoPanelLayout.setHorizontalGroup(
1654        infoPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
1655        .addComponent(actuatorPanel1, javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
1656        .addComponent(actuatorPanel2, javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
1657        .addGroup(infoPanelLayout.createSequentialGroup()
1658            .addContainerGap()
1659            .addComponent(jPanel2, javax.swing.GroupLayout.PREFERRED_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREF
1660            .addContainerGap())
1661    );
1662    infoPanelLayout.setVerticalGroup(
1663        infoPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
1664        .addGroup(infoPanelLayout.createSequentialGroup()
1665            .addContainerGap()
1666            .addComponent(jPanel2, javax.swing.GroupLayout.PREFERRED_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREF
1667            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
1668            .addComponent(actuatorPanel1, javax.swing.GroupLayout.PREFERRED_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayou
1669            .addGap(15, 15, 15)
1670            .addComponent(actuatorPanel2, javax.swing.GroupLayout.PREFERRED_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayou
1671            .addGap(0, 0, Short.MAX_VALUE))
1672    );
1673
1674    javax.swing.GroupLayout backgroundLayout = new javax.swing.GroupLayout(background);
1675    background.setLayout(backgroundLayout);
1676    backgroundLayout.setHorizontalGroup(
1677        backgroundLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
1678        .addGroup(backgroundLayout.createSequentialGroup()
1679            .addGroup(backgroundLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING, false)
1680                .addComponent(controlPanel, javax.swing.GroupLayout.DEFAULT_SIZE, 985, Short.MAX_VALUE)
1681                .addComponent(cameraPanel, javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.P
1682            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
1683            .addComponent(infoPanel, javax.swing.GroupLayout.PREFERRED_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PRI
1684            .addContainerGap())
1685    );
1686    backgroundLayout.setVerticalGroup(
1687        backgroundLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
1688        .addGroup(backgroundLayout.createSequentialGroup()
1689            .addGroup(backgroundLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING, false)
1690                .addGroup(javax.swing.GroupLayout.Alignment.LEADING, backgroundLayout.createSequentialGroup()
1691                    .addContainerGap()
1692                    .addComponent(infoPanel, javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
1693                .addGroup(javax.swing.GroupLayout.Alignment.LEADING, backgroundLayout.createSequentialGroup()
1694                    .addComponent(cameraPanel, javax.swing.GroupLayout.PREFERRED_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLay
1695                    .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
1696                    .addComponent(controlPanel, javax.swing.GroupLayout.PREFERRED_SIZE, 214, javax.swing.GroupLayout.PREFERRED_SIZE)))
1697            .addGap(170, 170, 170))
1698    );
1699
1700    gridBagConstraints = new java.awt.GridBagConstraints();
1701    gridBagConstraints.gridx = 0;
1702    gridBagConstraints.gridy = 0;
1703    gridBagConstraints.ipadx = 856;
1704    gridBagConstraints.ipady = 318;
1705    gridBagConstraints.anchor = java.awt.GridBagConstraints.NORTHWEST;
1706    gridBagConstraints.insets = new java.awt.Insets(0, 0, 6, 0);
1707    window.add(background, gridBagConstraints);
1708
1709    gridBagConstraints = new java.awt.GridBagConstraints();
1710    gridBagConstraints.gridx = 0;
1711    gridBagConstraints.gridy = 0;
1712    gridBagConstraints.anchor = java.awt.GridBagConstraints.NORTHWEST;
1713    getContentPane().add(window, gridBagConstraints);
1714    getContentPane().add(filler3, new java.awt.GridBagConstraints());
1715
1716    jMenuBar.setForeground(new java.awt.Color(39, 44, 50));
1717
1718    jMenuTools.setText("Tools");
1719    jMenuTools.setCursor(new java.awt.Cursor(java.awt.Cursor.HAND_CURSOR));
1720
1721    jMenuEchosounder.setText("Echo sounder");
1722    jMenuEchosounder.addActionListener(new java.awt.event.ActionListener() {
1723        public void actionPerformed(java.awt.event.ActionEvent evt) {
1724            jMenuEchosounderActionPerformed(evt);
1725        }
1726    });
1727    jMenuTools.add(jMenuEchosounder);
1728
1729    jMenuIOController.setText("I/O Controller");
1730    jMenuIOController.addActionListener(new java.awt.event.ActionListener() {
1731        public void actionPerformed(java.awt.event.ActionEvent evt) {
1732            jMenuIOControllerActionPerformed(evt);
1733        }
1734    });
1735    jMenuTools.add(jMenuIOController);
1736
1737    jMenuOptions.setText("Options");
```

```
1738        jMenuOptions.addActionListener(new java.awt.event.ActionListener() {
1739            public void actionPerformed(java.awt.event.ActionEvent evt) {
1740                jMenuOptionsActionPerformed(evt);
1741            }
1742        });
1743        jMenuTools.add(jMenuOptions);
1744
1745        jMenuItemExit.setText("Exit");
1746        jMenuItemExit.addActionListener(new java.awt.event.ActionListener() {
1747            public void actionPerformed(java.awt.event.ActionEvent evt) {
1748                jMenuItemExitActionPerformed(evt);
1749            }
1750        });
1751        jMenuTools.add(jMenuItemExit);
1752
1753        jMenuBar.add(jMenuTools);
1754
1755        jMenuHelp.setText("Help");
1756        jMenuHelp.setToolTipText("");
1757        jMenuHelp.setCursor(new java.awt.Cursor(java.awt.Cursor.HAND_CURSOR));
1758
1759        jMenuAbout.setText("About");
1760        jMenuAbout.addActionListener(new java.awt.event.ActionListener() {
1761            public void actionPerformed(java.awt.event.ActionEvent evt) {
1762                jMenuAboutActionPerformed(evt);
1763            }
1764        });
1765        jMenuHelp.add(jMenuAbout);
1766
1767        jMenuBar.add(jMenuHelp);
1768
1769        jMenuConnect.setIcon(new javax.swing.ImageIcon(getClass().getResource("/ntnusubsea/gui/Images/NotCalibrated.gif"))); // NOI18N
1770        jMenuConnect.setText("Connect");
1771        jMenuConnect.setCursor(new java.awt.Cursor(java.awt.Cursor.HAND_CURSOR));
1772        jMenuConnect.setFocusPainted(true);
1773
1774        jMenuItemConnect.setText("Connect");
1775        jMenuItemConnect.addActionListener(new java.awt.event.ActionListener() {
1776            public void actionPerformed(java.awt.event.ActionEvent evt) {
1777                jMenuItemConnectActionPerformed(evt);
1778            }
1779        });
1780        jMenuConnect.add(jMenuItemConnect);
1781
1782        jMenuItemDisconnect.setText("Disconnect");
1783        jMenuItemDisconnect.setEnabled(false);
1784        jMenuItemDisconnect.addActionListener(new java.awt.event.ActionListener() {
1785            public void actionPerformed(java.awt.event.ActionEvent evt) {
1786                jMenuItemDisconnectActionPerformed(evt);
1787            }
1788        });
1789        jMenuConnect.add(jMenuItemDisconnect);
1790
1791        jMenuBar.add(jMenuConnect);
1792
1793        jMenuCalibrate.setIcon(new javax.swing.ImageIcon(getClass().getResource("/ntnusubsea/gui/Images/NotCalibrated.gif"))); // NOI18N
1794        jMenuCalibrate.setText("Not Calibrated!");
1795        jMenuCalibrate.setCursor(new java.awt.Cursor(java.awt.Cursor.HAND_CURSOR));
1796        jMenuCalibrate.setDisabledIcon(null);
1797
1798        calibrateMenuItem.setText("Calibrate");
1799        calibrateMenuItem.addActionListener(new java.awt.event.ActionListener() {
1800            public void actionPerformed(java.awt.event.ActionEvent evt) {
1801                calibrateMenuItemActionPerformed(evt);
1802            }
1803        });
1804        jMenuCalibrate.add(calibrateMenuItem);
1805
1806        jMenuBar.add(jMenuCalibrate);
1807
1808        jMenuRovReady.setIcon(new javax.swing.ImageIcon(getClass().getResource("/ntnusubsea/gui/Images/NotCalibrated.gif"))); // NOI18N
1809        jMenuRovReady.setText("ROV not ready");
1810        jMenuRovReady.setBorderPainted(false);
1811        jMenuRovReady.setContentAreaFilled(false);
1812        jMenuRovReady.setCursor(new java.awt.Cursor(java.awt.Cursor.DEFAULT_CURSOR));
1813        jMenuRovReady.setFocusable(false);
1814        jMenuBar.add(jMenuRovReady);
1815
1816        jMenuLogger.setIcon(new javax.swing.ImageIcon(getClass().getResource("/ntnusubsea/gui/Images/NotCalibrated.gif"))); // NOI18N
1817        jMenuLogger.setText("Not logging");
1818        jMenuLogger.setActionCommand("jMenuLogger");
1819        jMenuLogger.setCursor(new java.awt.Cursor(java.awt.Cursor.HAND_CURSOR));
1820        jMenuLogger.addActionListener(new java.awt.event.ActionListener() {
1821            public void actionPerformed(java.awt.event.ActionEvent evt) {
1822                jMenuLoggerActionPerformed(evt);
1823            }
1824        });
```

```
1825
1826        jMenuItemStartLogging.setText("Start logging");
1827        jMenuItemStartLogging.setCursor(new java.awt.Cursor(java.awt.Cursor.HAND_CURSOR));
1828        jMenuItemStartLogging.addActionListener(new java.awt.event.ActionListener() {
1829            public void actionPerformed(java.awt.event.ActionEvent evt) {
1830                jMenuItemStartLoggingActionPerformed(evt);
1831            }
1832        });
1833        jMenuLogger.add(jMenuItemStartLogging);
1834
1835        jMenuItemStopLogging.setText("Stop logging");
1836        jMenuItemStopLogging.setCursor(new java.awt.Cursor(java.awt.Cursor.HAND_CURSOR));
1837        jMenuItemStopLogging.setEnabled(false);
1838        jMenuItemStopLogging.addActionListener(new java.awt.event.ActionListener() {
1839            public void actionPerformed(java.awt.event.ActionEvent evt) {
1840                jMenuItemStopLoggingActionPerformed(evt);
1841            }
1842        });
1843        jMenuLogger.add(jMenuItemStopLogging);
1844
1845        jMenuBar.add(jMenuLogger);
1846
1847        jMenu1.setText("                ");
1848        jMenu1.setBorderPainted(false);
1849        jMenu1.setContentAreaFilled(false);
1850        jMenu1.setEnabled(false);
1851        jMenu1.setFocusable(false);
1852        jMenu1.setPreferredSize(new java.awt.Dimension(600, 21));
1853        jMenuBar.add(jMenu1);
1854
1855        jMenuVoltage.setIcon(new javax.swing.ImageIcon(getClass().getResource("/ntnusubsea/gui/Images/NotCalibrated.gif"))); // NOI18N
1856        jMenuVoltage.setText("Voltage: 0.0 V");
1857        jMenuVoltage.setBorderPainted(false);
1858        jMenuVoltage.setContentAreaFilled(false);
1859        jMenuVoltage.setFocusable(false);
1860        jMenuBar.add(jMenuVoltage);
1861
1862        jMenu3.setBorderPainted(false);
1863        jMenu3.setContentAreaFilled(false);
1864        jMenu3.setEnabled(false);
1865        jMenu3.setFocusable(false);
1866        jMenu3.setPreferredSize(new java.awt.Dimension(20, 5));
1867        jMenuBar.add(jMenu3);
1868
1869        jMenuPing.setIcon(new javax.swing.ImageIcon(getClass().getResource("/ntnusubsea/gui/Images/NotCalibrated.gif"))); // NOI18N
1870        jMenuPing.setText("Ping (ROV): Not connected");
1871        jMenuBar.add(jMenuPing);
1872
1873        setJMenuBar(jMenuBar);
1874
1875        pack();
1876    }// </editor-fold>
1877
1878    private void fullscreenButtonActionPerformed(java.awt.event.ActionEvent evt) {
1879        Rectangle maximumWindowBounds = GraphicsEnvironment.getLocalGraphicsEnvironment().getMaximumWindowBounds();
1880        int width = (int) maximumWindowBounds.getWidth();
1881        int height = (int) maximumWindowBounds.getHeight();
1882        fullscreen.setLocationRelativeTo(this);
1883        this.setVisible(false);
1884        fullscreen.setExtendedState(MAXIMIZED_BOTH);
1885        fullscreen.setUndecorated(true);
1886        fullscreen.setVisible(true);
1887        exitFullscreenButton.setBounds(width - exitFullscreenButton.getWidth(), height - exitFullscreenButton.getHeight() - 25, 30, 30);
1888    }
1889
1890    private void exitFullscreenButtonActionPerformed(java.awt.event.ActionEvent evt) {
1891        fullscreen.setVisible(false);
1892        fullscreen.dispose();
1893        this.setVisible(true);
1894    }
1895
1896    private void fullscreenKeyPressed(java.awt.event.KeyEvent evt) {
1897        int key = evt.getKeyCode();
1898        if (key == KeyEvent.VK_ESCAPE) {
1899            fullscreen.dispose();
1900        }
1901        System.out.println(key);
1902    }
1903
1904    private void actuatorPSPosBarStateChanged(javax.swing.event.ChangeEvent evt) {
1905        int actuatorTime1 = actuatorPSPosBar.getValue();
1906        if (actuatorTime1 <= 254) {
1907 //           actuatorPSPosBar.setForeground(new Color(actuatorTime1, 255, 0));
1908            actuatorPSPosBar.setForeground(new Color(77, 192, 99));
1909            warningLabel1.setText("");
1910            warningLabel1.setBackground(new Color(42, 48, 57));
1911        } else {
```

```
1912            actuatorPSPosBar.setForeground(new Color(255, 255 + (256 - actuatorTime1), 0));
1913            warningLabel1.setText("Warning!");
1914            warningLabel1.setBackground(Color.red);
1915        }
1916    }
1917
1918    private void seafloorModeButtonActionPerformed(java.awt.event.ActionEvent evt) {
1919
1920        double d = data.getRovDepth();
1921        targetDistanceTextField.setText(String.valueOf(d));
1922        actuatorControlPS.setValue(127);
1923        actuatorControlSB.setValue(127);
1924        actuatorPosLabel.setText("<html>PS: " + String.valueOf(actuatorControlPS.getValue()) + "<br/><br/>SB: " + String.valueOf(actuatorControlSB.getValue()
1925        actuatorControlPS.setEnabled(false);
1926        actuatorControlSB.setEnabled(false);
1927        if (this.targetMode != 1) {
1928            this.targetMode = 1;
1929            System.out.println("Mode 1 - Distance from seafloor");
1930            data.setManualMode(false);
1931            try {
1932                this.client_ROV.sendCommand("cmd_targetMode:" + String.valueOf(this.targetMode));
1933                this.client_ROV.sendCommand("cmd_targetDistance:" + String.valueOf(d));
1934            } catch (IOException ex) {
1935                System.out.println("IOException: " + ex.getMessage());
1936            }
1937        } else {
1938            System.out.println("Already in seafloor mode.");
1939        }
1940    }
1941
1942    private void lightSwitchActionPerformed(java.awt.event.ActionEvent evt) {
1943        if (lightSwitch.isSelected()) {
1944            try {
1945                int value = lightSlider.getValue();
1946                client_Camera.sendCommand("setLed:" + String.valueOf(value));
1947                //lightSlider.setValue(40);
1948            } catch (IOException ex) {
1949                Logger.getLogger(ROVFrame.class.getName()).log(Level.SEVERE, null, ex);
1950            }
1951        } else {
1952            try {
1953                client_Camera.sendCommand("setLed:0");
1954                //lightSlider.setValue(19);
1955            } catch (IOException ex) {
1956                Logger.getLogger(ROVFrame.class.getName()).log(Level.SEVERE, null, ex);
1957            }
1958        }
1959    }
1960
1961    private void emergencyStopButtonActionPerformed(java.awt.event.ActionEvent evt) {
1962 //       previousSetpoint = setpoint;
1963 //       setpoint = 0.000;
1964        if (!data.isEmergencyMode()) {
1965            data.setEmergencyMode(true);
1966        }
1967        targetDistanceTextField.setText("0.00");
1968        setpointLabel.setText("EMERGENCY STOP: " + targetDistanceTextField.getText() + "m");
1969        setpointLabel.setBackground(new Color(255, 0, 0));
1970        manualControlButton.doClick();
1971        try {
1972            this.client_ROV.sendCommand("cmd_targetMode:2");
1973            this.client_ROV.sendCommand("cmd_actuatorPS:254");
1974            this.client_ROV.sendCommand("cmd_actuatorSB:254");
1975        } catch (IOException ex) {
1976            System.out.println("IOException in emergencyStopButtonActionPerformed: " + ex.getMessage());
1977        }
1978    }
1979
1980    private void jMenuItemConnectActionPerformed(java.awt.event.ActionEvent evt) {
1981 //       String ip = (String) JOptionPane.showInputDialog(this, "Enter IP", "Connection", JOptionPane.PLAIN_MESSAGE, null, null, data.getIP_Rov());
1982        try {
1983
1984            this.clientThreadExecutor = Executors.newScheduledThreadPool(4);
1985            clientThreadExecutor.scheduleAtFixedRate(client_Pinger,
1986                    0, 1000, TimeUnit.MILLISECONDS);
1987            clientThreadExecutor.scheduleAtFixedRate(client_ROV,
1988                    0, 100, TimeUnit.MILLISECONDS);
1989            clientThreadExecutor.scheduleAtFixedRate(client_Camera,
1990                    0, 100, TimeUnit.MILLISECONDS);
1991            clientThreadExecutor.scheduleAtFixedRate(udpServer,
1992                    0, 20, TimeUnit.MILLISECONDS);
1993            Thread.sleep(500);
1994
1995            if (client_ROV.isConnected() && client_Camera.isConnected()) {
1996                // ROV RPi:
1997                lightSwitch_lbl.setEnabled(true);
1998                emergencyStopButton.setEnabled(true);
```

```
1999            lightSwitchBlueLED.setEnabled(true);
2000            targetDistanceTextField.setEnabled(true);
2001            depthModeButton.setEnabled(true);
2002            seafloorModeButton.setEnabled(true);
2003            InputControllerButton.setEnabled(true);
2004            manualControlButton.setEnabled(true);
2005            resetManualControlButton.setEnabled(true);
2006            lockButton.setEnabled(true);
2007            io.enableIO();
2008            // Camera RPi:
2009            lightSwitch.setEnabled(true);
2010            lightSlider.setEnabled(true);
2011            getPhotosButton.setEnabled(true);
2012            clearImagesButton.setEnabled(true);
2013            photoModeButton.setEnabled(true);
2014            cameraPitchSlider.setEnabled(true);
2015            cameraPitchTextField.setEnabled(true);
2016            delayTextField.setEnabled(true);
2017
2018            this.client_ROV.sendCommand("cmd_pid_p:" + data.getKp());
2019            Thread.sleep(10);
2020            this.client_ROV.sendCommand("cmd_pid_i:" + data.getKi());
2021            Thread.sleep(10);
2022            this.client_ROV.sendCommand("cmd_pid_d:" + data.getKd());
2023            Thread.sleep(10);
2024            this.client_ROV.sendCommand("cmd_offsetDepthBeneathROV:" + data.getOffsetDepthBeneathROV());
2025            Thread.sleep(10);
2026            this.client_ROV.sendCommand("cmd_offsetROVdepth:" + data.getOffsetROVdepth());
2027            jMenuConnect.setText("Connected 2/2");
2028            jMenuConnect.setIcon(new ImageIcon(ImageIO.read(getClass().getResource("/ntnusubsea/gui/Images/Calibrated.gif"))));
2029            jMenuItemDisconnect.setEnabled(true);
2030            jMenuItemConnect.setEnabled(false);
2031            JOptionPane.showMessageDialog(this,
2032                    "Successfully connected to the ROV RPi and the camera RPi.",
2033                    "Connected",
2034                    JOptionPane.PLAIN_MESSAGE);
2035        } else if (client_ROV.isConnected() && !client_Camera.isConnected()) {
2036            // ROV RPi:
2037            emergencyStopButton.setEnabled(true);
2038            lightSwitchBlueLED.setEnabled(true);
2039            targetDistanceTextField.setEnabled(true);
2040            depthModeButton.setEnabled(true);
2041            seafloorModeButton.setEnabled(true);
2042            InputControllerButton.setEnabled(true);
2043            manualControlButton.setEnabled(true);
2044            resetManualControlButton.setEnabled(true);
2045            lockButton.setEnabled(true);
2046            io.enableIO();
2047
2048            this.client_ROV.sendCommand("cmd_pid_p:" + data.getKp());
2049            Thread.sleep(10);
2050            this.client_ROV.sendCommand("cmd_pid_i:" + data.getKi());
2051            Thread.sleep(10);
2052            this.client_ROV.sendCommand("cmd_pid_d:" + data.getKd());
2053            Thread.sleep(10);
2054            this.client_ROV.sendCommand("cmd_offsetDepthBeneathROV:" + data.getOffsetDepthBeneathROV());
2055            Thread.sleep(10);
2056            this.client_ROV.sendCommand("cmd_offsetROVdepth:" + data.getOffsetROVdepth());
2057            jMenuConnect.setText("Connected 1/2");
2058            jMenuConnect.setIcon(new ImageIcon(ImageIO.read(getClass().getResource("/ntnusubsea/gui/Images/NotCalibrated.gif"))));
2059            jMenuItemDisconnect.setEnabled(true);
2060            jMenuItemConnect.setEnabled(false);
2061            JOptionPane.showMessageDialog(this,
2062                    "Could only connect to the ROV RPi, but not the camera RPi...",
2063                    "Connected 1/2",
2064                    JOptionPane.PLAIN_MESSAGE);
2065        } else if (!client_ROV.isConnected() && client_Camera.isConnected()) {
2066            // Camera RPi:
2067            lightSwitch_lbl.setEnabled(true);
2068            lightSwitch.setEnabled(true);
2069            lightSlider.setEnabled(true);
2070            getPhotosButton.setEnabled(true);
2071            clearImagesButton.setEnabled(true);
2072            photoModeButton.setEnabled(true);
2073            cameraPitchSlider.setEnabled(true);
2074            cameraPitchTextField.setEnabled(true);
2075            delayTextField.setEnabled(true);
2076
2077            jMenuConnect.setText("Connected 1/2");
2078            jMenuConnect.setIcon(new ImageIcon(ImageIO.read(getClass().getResource("/ntnusubsea/gui/Images/NotCalibrated.gif"))));
2079            jMenuItemDisconnect.setEnabled(true);
2080            jMenuItemConnect.setEnabled(false);
2081            JOptionPane.showMessageDialog(this,
2082                    "Could only connect to the camera RPi, but not the ROV RPi...",
2083                    "Connected 1/2",
2084                    JOptionPane.PLAIN_MESSAGE);
2085        } else {
```

```
2086            JOptionPane.showMessageDialog(this,
2087                    "Error: Could not connect to either the ROV RPi nor the camera RPi.",
2088                    "Error: Could not connect",
2089                    JOptionPane.PLAIN_MESSAGE);
2090            client_Pinger.disconnect();
2091            client_ROV.disconnect();
2092            client_Camera.disconnect();
2093
2094            if (clientThreadExecutor != null) {
2095                clientThreadExecutor.shutdown();
2096                clientThreadExecutor = null;
2097            }
2098        }
2099
2100    } catch (Exception ex) {
2101        jMenuConnect.setText("Connect");
2102        try {
2103            jMenuConnect.setIcon(new ImageIcon(ImageIO.read(getClass().getResource("/ntnusubsea/gui/Images/NotCalibrated.gif"))));
2104        } catch (IOException ex1) {
2105            System.out.println("IOException: " + ex.getMessage());
2106        }
2107        JOptionPane.showMessageDialog(this,
2108                "Connection failed.",
2109                "Conncetion error",
2110                JOptionPane.ERROR_MESSAGE);
2111    }
2112 }
2113
2114 private void jMenuItemDisconnectActionPerformed(java.awt.event.ActionEvent evt) {
2115
2116     try {
2117         client_Pinger.disconnect();
2118         client_ROV.disconnect();
2119         client_Camera.disconnect();
2120         jMenuPing.setText("Ping (ROV): Not connected");
2121
2122         if (clientThreadExecutor != null) {
2123             clientThreadExecutor.shutdown();
2124         }
2125         // ROV RPi:
2126         emergencyStopButton.setEnabled(false);
2127         lightSwitchBlueLED.setEnabled(false);
2128         targetDistanceTextField.setEnabled(false);
2129         depthModeButton.setEnabled(false);
2130         seafloorModeButton.setEnabled(false);
2131         manualControlButton.setEnabled(false);
2132         InputControllerButton.setEnabled(false);
2133         resetManualControlButton.setEnabled(false);
2134         lockButton.setEnabled(false);
2135         io.disableIO();
2136         // Camera RPi:
2137         lightSwitch_lbl.setEnabled(false);
2138         lightSwitch.setEnabled(false);
2139         lightSlider.setEnabled(false);
2140         getPhotosButton.setEnabled(false);
2141         clearImagesButton.setEnabled(false);
2142         photoModeButton.setEnabled(false);
2143         cameraPitchSlider.setEnabled(false);
2144         cameraPitchTextField.setEnabled(false);
2145         delayTextField.setEnabled(false);
2146
2147         jMenuItemDisconnect.setEnabled(false);
2148         jMenuItemConnect.setEnabled(true);
2149         jMenuConnect.setText("Connect");
2150         jMenuConnect.setIcon(new ImageIcon(ImageIO.read(getClass().getResource("/ntnusubsea/gui/Images/NotCalibrated.gif"))));
2151         JOptionPane.showMessageDialog(this,
2152                 "Successfully disconnected from the ROV RPi and the camera RPi.",
2153                 "Disconnected",
2154                 JOptionPane.PLAIN_MESSAGE);
2155         videoImage = ImageIO.read(getClass().getResource("ntnusubsea/gui/Images/TowedROV.jpg"));
2156         data.setVideoImage(videoImage);
2157     } catch (IOException ex) {
2158         JOptionPane.showMessageDialog(this,
2159                 "Failed to disconnect.",
2160                 "Disconnect error",
2161                 JOptionPane.ERROR_MESSAGE);
2162     }
2163 }
2164
2165 private void jMenuItemExitActionPerformed(java.awt.event.ActionEvent evt) {
2166     System.exit(0);
2167 }
2168
2169
2170 private void jMenuEchosounderActionPerformed(java.awt.event.ActionEvent evt) {
2171     echoSounder.setVisible(true);
2172 }
```

```
2173
2174    private void helpFrameOKbuttonActionPerformed(java.awt.event.ActionEvent evt) {
2175        helpframe.dispose();
2176    }
2177
2178    private void jMenuAboutActionPerformed(java.awt.event.ActionEvent evt) {
2179        helpframe.setVisible(true);
2180        helpframe.setSize(helpframe.getPreferredSize());
2181        helpframe.pack();
2182        helpframe.setLocationRelativeTo(null);
2183        //helpframe.setLocation(this.getLocation().x, this.getLocation().y);
2184    }
2185
2186    private void actuatorSBPosBarStateChanged(javax.swing.event.ChangeEvent evt) {
2187        int actuatorTime2 = actuatorSBPosBar.getValue();
2188        if (actuatorTime2 <= 254) {
2189            actuatorSBPosBar.setForeground(new Color(77, 192, 99));
2190            warningLabel2.setText("");
2191            warningLabel2.setBackground(new Color(42, 48, 57));
2192        } else {
2193            actuatorSBPosBar.setForeground(new Color(255, 255 + (256 - actuatorTime2), 0));
2194            warningLabel2.setText("Warning!");
2195            warningLabel2.setBackground(Color.red);
2196        }
2197    }
2198
2199    private void jMenuOptionsActionPerformed(java.awt.event.ActionEvent evt) {
2200        options.setVisible(true);
2201    }
2202
2203    private void depthModeButtonActionPerformed(java.awt.event.ActionEvent evt) {
2204        double d = data.getRovDepth();
2205        targetDistanceTextField.setText(String.valueOf(d));
2206        actuatorControlPS.setValue(127);
2207        actuatorControlSB.setValue(127);
2208        actuatorPosLabel.setText("<html>PS: " + String.valueOf(actuatorControlPS.getValue()) + "<br/><br/>SB: " + String.valueOf(actuatorControlSB.getValue()
2209        actuatorControlPS.setEnabled(false);
2210        actuatorControlSB.setEnabled(false);
2211        if (this.targetMode != 0) {
2212            this.targetMode = 0;
2213            data.setManualMode(false);
2214            System.out.println("Mode 0 - Depth");
2215            try {
2216                this.client_ROV.sendCommand("cmd_targetMode:" + String.valueOf(this.targetMode));
2217                this.client_ROV.sendCommand("cmd_targetDistance:" + String.valueOf(d));
2218            } catch (IOException ex) {
2219                System.out.println("IOException: " + ex.getMessage());
2220            }
2221        } else {
2222            System.out.println("Already in depth mode.");
2223        }
2224    }
2225
2226    private void jMenuIOControllerActionPerformed(java.awt.event.ActionEvent evt) {
2227        io.setVisible(true);
2228    }
2229
2230    private void cameraPitchTextFieldActionPerformed(java.awt.event.ActionEvent evt)
2231    {
2232
2233        try {
2234            cameraPitchTextField.commitEdit();
2235            if (cameraPitchTextField.getText() != null && isInteger(cameraPitchTextField.getText())) {
2236                this.cameraPitchValue = Integer.parseInt(cameraPitchTextField.getText());
2237            }
2238            if (this.cameraPitchValue > 100) {
2239                this.cameraPitchValue = 100;
2240                System.out.println("Camera Pitch input too high! Set to max (100)");
2241            } else if (this.cameraPitchValue < 0) {
2242                this.cameraPitchValue = 0;
2243                System.out.println("Camera Pitch input too low! Set to min (0)");
2244            }
2245
2246            if (this.cameraPitchValue <= 75 && this.cameraPitchValue >= 50) {
2247                cameraPitchLabel.setBackground(new Color(28, 28, 28));
2248                cameraPitchSlider.setValue(this.cameraPitchValue);
2249                cameraPitchLabel.setText(String.valueOf(this.cameraPitchValue));
2250                data.setCameraPitchValue(this.cameraPitchValue);
2251                System.out.println("Camera Pitch set to " + this.cameraPitchValue);
2252                //this.client_Camera.sendCommand("setPitch:" + String.valueOf(this.cameraPitchValue));
2253                // Send this to the python TcpController program running on the Camera RPi
2254
2255            } else {
2256                cameraPitchTextField.setValue(null);
2257                JOptionPane.showMessageDialog(this,
2258                        "Input is invalid. Valid integer values are 50-75.",
2259                        "Input error",
```

```
                JOptionPane.ERROR_MESSAGE);
        }

    } catch (ParseException ex) {
        System.out.println(ex.getMessage());

    } catch (Exception ex) {
        System.out.println("Error: " + ex.getMessage());
    }
}

private void cameraPitchSliderMouseReleased(java.awt.event.MouseEvent evt)
{
    try {
        this.cameraPitchValue = cameraPitchSlider.getValue();
        cameraPitchLabel.setText(Integer.toString(this.cameraPitchValue));
        cameraPitchTextField.setText(Integer.toString(this.cameraPitchValue));
        data.setCameraPitchValue(cameraPitchValue);
        System.out.println("Camera Pitch set to " + cameraPitchSlider.getValue());
        this.client_Camera.sendCommand("setPitch:" + String.valueOf(this.cameraPitchValue));
        // Send this to the java program running on the Camera RPi
    } catch (Exception ex) {
        Logger.getLogger(ROVFrame.class.getName()).log(Level.SEVERE, null, ex);
    }
}

private void delayTextFieldActionPerformed(java.awt.event.ActionEvent evt)
{
    try {
        if (delayTextField.getText() != null && isNumeric(delayTextField.getText())) {
            this.photoModeDelay = Double.parseDouble(delayTextField.getText());
            if (this.photoModeDelay > 99) {
                this.photoModeDelay = 99;
                System.out.println("Photo Mode Delay input too high! Set to max (99)");
            } else if (this.photoModeDelay < 0) {
                this.photoModeDelay = 0;
                System.out.println("Photo Mode Delay input too low! Set to min (0)");
            }
            photoModeDelayLabel.setText(String.valueOf(this.photoModeDelay) + " s");
            data.setPhotoModeDelay(this.photoModeDelay);
            System.out.println("Photo Mode Delay set to " + String.valueOf(this.photoModeDelay));

            this.udpServer.sendDelayCommand();
        } else {
            System.out.println("Invalid delay entered.");
        }
    } catch (NumberFormatException ex) {
        Logger.getLogger(ROVFrame.class.getName()).log(Level.SEVERE, null, ex);
    }
}

private void photoModeButtonActionPerformed(java.awt.event.ActionEvent evt)
{
    if (photoModeButton.isSelected()) {
        try {
            data.setPhotoMode(true);
        } catch (Exception ex) {
            Logger.getLogger(ROVFrame.class.getName()).log(Level.SEVERE, null, ex);
        }
    } else {
        try {
            data.setPhotoMode(false);
        } catch (Exception ex) {
            Logger.getLogger(ROVFrame.class.getName()).log(Level.SEVERE, null, ex);
        }
    }
}

private void getPhotosButtonActionPerformed(java.awt.event.ActionEvent evt)
{
//      GetImagesFrame imgframe = new GetImagesFrame();
//      imgframe.setVisible(true);
//      imgframe.setLocation(this.getLocation().x, this.getLocation().y);
    Object[] options
            = {
                "OK", "Cancel"
            };
    int choice = JOptionPane.showOptionDialog(this,
            "Warning! This might take a few minutes and will freeze the GUI.\nPress OK to continue, or Cancel to abort.",
            "Warning!",
            JOptionPane.YES_NO_OPTION, //int optionType
            JOptionPane.INFORMATION_MESSAGE, //int messageType
            null, //Icon icon,
            options, //Object[] options,
            options[0]);//Object initialValue
    if (choice == 0) {
        FtpClient ftp = null;
```

```
2347        File dir = null;
2348        try {
2349            try {
2350                dir = new File("C:\\TowedROV\\ROV_Photos\\");
2351                if (!dir.exists() || !dir.isDirectory()) {
2352                    System.out.println("No directory found, creating a new one at C://TowedROV/ROV_Photos/");
2353                    dir.mkdir();
2354                }
2355            } catch (Exception e) {
2356                System.out.println("No directory found, creating a new one at C://TowedROV/ROV_Photos/");
2357                dir.mkdir();
2358            }
2359
2360            // Test FTP Client (WORKING)
2361            ftp = new FtpClient(this.client_Camera.getIP());
2362            Thread ftpThread = new Thread(ftp);
2363            ftpThread.start();
2364            ftp.open();
2365            //int count = 1;
2366            Collection<String> list = ftp.getFileList("ftp/images");
2367            for (String s : list) {
2368                ftp.downloadFile("ftp/images/" + s, dir.getPath() + "\\" + s, dir.getPath());
2369                //imgframe.progressLabel.setText(String.valueOf(count) + " / " + String.valueOf(list.size()));
2370                System.out.println(s);
2371                //count++;
2372            }
2373
2374        } catch (Exception ex) {
2375            System.out.println(ex.getMessage());
2376        } finally {
2377            if (ftp != null) {
2378                ftp.disconnect();
2379            }
2380        }
2381    } else if (choice == 0) {
2382        System.out.println("Cancelled getting the images.");
2383    } else {
2384        System.out.println("No option chosen: Cancelled getting the images.");
2385    }
2386 }
2387
2388 private void lightSliderMouseReleased(java.awt.event.MouseEvent evt)
2389 {
2390    try {
2391        if (lightSwitch.isSelected()) {
2392            //data.setCameraPitchValue(cameraPitchValue);
2393            int value = lightSlider.getValue();
2394            System.out.println("ROV Lights set to " + value);
2395            this.client_Camera.sendCommand("setLed:" + String.valueOf(value));
2396            // Send this to the python TcpController program running on the Camera RPi
2397        }
2398    } catch (Exception ex) {
2399        Logger.getLogger(ROVFrame.class.getName()).log(Level.SEVERE, null, ex);
2400    }
2401 }
2402
2403 private void manualControlButtonActionPerformed(java.awt.event.ActionEvent evt)
2404 {
2405    if (this.targetMode != 2) {
2406        actuatorControlPS.setEnabled(true);
2407        actuatorControlSB.setEnabled(true);
2408        data.setManualMode(true);
2409        this.targetMode = 2;
2410        System.out.println("Mode 2 - Manual wing control");
2411        try {
2412            this.client_ROV.sendCommand("cmd_targetMode:" + String.valueOf(this.targetMode));
2413        } catch (IOException ex) {
2414            System.out.println("IOException: " + ex.getMessage());
2415        }
2416    } else {
2417        depthModeButton.doClick();
2418    }
2419 }
2420
2421 private void resetManualControlButtonActionPerformed(java.awt.event.ActionEvent evt)
2422 {
2423    actuatorControlPS.setValue(127);
2424    actuatorControlSB.setValue(127);
2425    actuatorPosLabel.setText("<html>PS: " + String.valueOf(actuatorControlPS.getValue()) + "<br/><br/>SB: " + String.valueOf(actuatorControlSB.getValue()
2426    if (manualControlButton.isSelected()) {
2427        try {
2428            this.client_ROV.sendCommand("cmd_actuatorPS:127");
2429            this.client_ROV.sendCommand("cmd_actuatorSB:127");
2430        } catch (IOException ex) {
2431            System.out.println("IOException: " + ex.getMessage());
2432        }
2433    } else {
```

```
2434            //System.out.println("Not in manual control mode.");
2435        }
2436    }
2437
2438    private void actuatorControlPSMouseReleased(java.awt.event.MouseEvent evt)
2439    {
2440        if (manualControlButton.isSelected()) {
2441            if (lockButton.isSelected()) {
2442                actuatorControlSB.setValue(actuatorControlPS.getValue());
2443                actuatorPosLabel.setText("<html>PS: " + String.valueOf(actuatorControlPS.getValue()) + "<br/><br/>SB: " + String.valueOf(actuatorControlSB.getVa
2444                try {
2445                    this.client_ROV.sendCommand("cmd_actuatorPS:" + String.valueOf(actuatorControlPS.getValue()));
2446                    this.client_ROV.sendCommand("cmd_actuatorSB:" + String.valueOf(actuatorControlSB.getValue()));
2447                } catch (IOException ex) {
2448                    System.out.println("IOException: " + ex.getMessage());
2449                }
2450            } else {
2451                actuatorPosLabel.setText("<html>PS: " + String.valueOf(actuatorControlPS.getValue()) + "<br/><br/>SB: " + String.valueOf(actuatorControlSB.getVa
2452                try {
2453                    this.client_ROV.sendCommand("cmd_actuatorPS:" + String.valueOf(actuatorControlPS.getValue()));
2454                } catch (IOException ex) {
2455                    System.out.println("IOException: " + ex.getMessage());
2456                }
2457            }
2458
2459        } else {
2460            //System.out.println("Not in manual control mode.");
2461        }
2462
2463    }
2464
2465    private void actuatorControlSBMouseReleased(java.awt.event.MouseEvent evt)
2466    {
2467        if (manualControlButton.isSelected()) {
2468            if (lockButton.isSelected()) {
2469                actuatorControlPS.setValue(actuatorControlSB.getValue());
2470                actuatorPosLabel.setText("<html>PS: " + String.valueOf(actuatorControlPS.getValue()) + "<br/><br/>SB: " + String.valueOf(actuatorControlSB.getVa
2471                try {
2472                    this.client_ROV.sendCommand("cmd_actuatorPS:" + String.valueOf(actuatorControlPS.getValue()));
2473                    this.client_ROV.sendCommand("cmd_actuatorSB:" + String.valueOf(actuatorControlSB.getValue()));
2474                } catch (IOException ex) {
2475                    System.out.println("IOException: " + ex.getMessage());
2476                }
2477            } else {
2478
2479                actuatorPosLabel.setText("<html>PS: " + String.valueOf(actuatorControlPS.getValue()) + "<br/><br/>SB: " + String.valueOf(actuatorControlSB.getVa
2480                try {
2481                    this.client_ROV.sendCommand("cmd_actuatorSB:" + String.valueOf(actuatorControlSB.getValue()));
2482                } catch (IOException ex) {
2483                    System.out.println("IOException: " + ex.getMessage());
2484                }
2485            }
2486        } else {
2487            //System.out.println("Not in manual control mode.");
2488        }
2489    }
2490
2491    private void targetDistanceTextFieldActionPerformed(java.awt.event.ActionEvent evt)
2492    {
2493        data.setEmergencyMode(false);
2494        try {
2495            targetDistanceTextField.commitEdit();
2496            if (targetDistanceTextField.getText() != null && isNumeric(targetDistanceTextField.getText())) {
2497                double newSetpoint;
2498                try {
2499                    newSetpoint = Double.parseDouble(targetDistanceTextField.getText());
2500                } catch (ClassCastException ex) {
2501                    Long newSetpointLong = Long.parseLong(targetDistanceTextField.getText());
2502                    newSetpoint = newSetpointLong.doubleValue();
2503                }
2504
2505                if (newSetpoint <= 50 && newSetpoint >= 0) {
2506                    setpointLabel.setBackground(new Color(39, 44, 50));
2507 //                   previousSetpoint = setpoint;
2508 //                   setpoint = newSetpoint;
2509 //                   depthInputTextField.setValue(null);
2510                    setpointLabel.setText("Current setpoint: " + newSetpoint + "m");
2511                    System.out.println("targetDistance set to " + String.valueOf(newSetpoint));
2512                    this.client_ROV.sendCommand("cmd_targetDistance:" + String.valueOf(newSetpoint));
2513                } else {
2514                    targetDistanceTextField.setValue(null);
2515                    targetDistanceTextField.setText("");
2516                    JOptionPane.showMessageDialog(this,
2517                            "Input is invalid. (Max depth 50m)",
2518                            "Input error",
2519                            JOptionPane.ERROR_MESSAGE);
2520                }
```

```
2521
2522            } else {
2523                System.out.println("Invalid input entered.");
2524            }
2525        } catch (IOException ex) {
2526            System.out.println("IOException: " + ex.getMessage());
2527        } catch (ParseException ex) {
2528            System.out.println("ParseException: " + ex.getMessage());
2529        }
2530    }
2531
2532    private void lockButtonActionPerformed(java.awt.event.ActionEvent evt)
2533    {
2534        if (actuatorControlPS.getValue() != 127) {
2535            actuatorControlPS.setValue(127);
2536            actuatorPosLabel.setText("<html>PS: " + String.valueOf(actuatorControlPS.getValue()) + "<br/><br/>SB: " + String.valueOf(actuatorControlSB.getValu
2537            try {
2538                this.client_ROV.sendCommand("cmd_actuatorPS:" + String.valueOf(actuatorControlPS.getValue()));
2539            } catch (IOException ex) {
2540                System.out.println("IOException: " + ex.getMessage());
2541            }
2542        }
2543        if (actuatorControlSB.getValue() != 127) {
2544            actuatorControlSB.setValue(127);
2545            actuatorPosLabel.setText("<html>PS: " + String.valueOf(actuatorControlPS.getValue()) + "<br/><br/>SB: " + String.valueOf(actuatorControlSB.getValu
2546            try {
2547                this.client_ROV.sendCommand("cmd_actuatorSB:" + String.valueOf(actuatorControlSB.getValue()));
2548            } catch (IOException ex) {
2549                System.out.println("IOException: " + ex.getMessage());
2550            }
2551        }
2552    }
2553
2554    private void clearImagesButtonActionPerformed(java.awt.event.ActionEvent evt)
2555    {
2556        try {
2557            this.client_Camera.sendCommand("clearImages");
2558            this.udpServer.sendResetIMGcommand();
2559            int tempNum = data.getImageNumber();
2560            data.setImageNumber(0);
2561            imageNumberLabel.setText("0 / 1000");
2562            JOptionPane.showMessageDialog(this,
2563                    "Successfully cleared " + String.valueOf(tempNum) + " of " + String.valueOf(tempNum) + " images on the RPi.",
2564                    "Cleared Images",
2565                    JOptionPane.PLAIN_MESSAGE);
2566            data.setImagesCleared(true);
2567        } catch (IOException ioex) {
2568            System.out.println("IOException: " + ioex.getMessage());
2569        } catch (Exception ex) {
2570            System.out.println("Exception: " + ex.getMessage());
2571        }
2572    }
2573
2574    private void calibrateMenuItemActionPerformed(java.awt.event.ActionEvent evt)
2575    {
2576        try {
2577            // TODO add your handling code here:
2578            // Kjør kalibrering!
2579            jMenuCalibrate.setText("Calibrated!");
2580            jMenuCalibrate.setIcon(new ImageIcon(ImageIO.read(getClass().getResource("/ntnusubsea/gui/Images/Calibrated.gif"))));
2581        } catch (IOException ex) {
2582            System.out.println("IOException when calibrating: " + ex.getMessage());
2583
2584        }
2585    }
2586
2587    private void lightSwitchBlueLEDActionPerformed(java.awt.event.ActionEvent evt)
2588    {
2589        if (lightSwitchBlueLED.isSelected()) {
2590            try {
2591                client_ROV.sendCommand("cmd_BlueLED:1");
2592            } catch (IOException ex) {
2593                System.out.println("Error while turning the blue LEDs on: " + ex.getMessage());
2594            }
2595        } else {
2596            try {
2597                client_ROV.sendCommand("cmd_BlueLED:0");
2598            } catch (IOException ex) {
2599                System.out.println("Error while turning the blue LEDs off: " + ex.getMessage());
2600            }
2601        }
2602    }
2603
2604    private void InputControllerButtonActionPerformed(java.awt.event.ActionEvent evt)
2605    {
2606        if (data.isControllerEnabled()) {
2607            data.setControllerEnabled(false);
```

```
2608        } else {
2609            data.setControllerEnabled(true);
2610        }
2611    }
2612
2613    private void jMenuLoggerActionPerformed(java.awt.event.ActionEvent evt)
2614    {
2615
2616    }
2617
2618    private void jMenuItemStartLoggingActionPerformed(java.awt.event.ActionEvent evt)
2619    {
2620        this.data.setStartLogging(true);
2621        encoder = new VideoEncoder(this.data);
2622        this.data.addObserver(encoder);
2623        this.encoderThreadExecutor = Executors.newScheduledThreadPool(1);
2624        encoderThreadExecutor.scheduleAtFixedRate(encoder,
2625                0, 40, TimeUnit.MILLISECONDS);
2626
2627        Runtime.getRuntime()
2628                .addShutdownHook(new Thread(new Runnable() {
2629                    @Override
2630                    public void run() {
2631                        if (encoder != null) {
2632                            encoder.finishVideo();
2633                        }
2634                        if (encoderThreadExecutor != null) {
2635                            encoderThreadExecutor.shutdown();
2636                        }
2637                    }
2638                },
2639                        "Shutdown-thread"));
2640        jMenuLogger.setText("Logging!");
2641        jMenuItemStartLogging.setEnabled(false);
2642        jMenuItemStopLogging.setEnabled(true);
2643        try {
2644            jMenuLogger.setIcon(new ImageIcon(ImageIO.read(getClass().getResource("/ntnusubsea/gui/Images/Calibrated.gif"))));
2645        } catch (IOException ex) {
2646            System.out.println("Error setting icon: " + ex.getMessage());
2647        }
2648    }
2649
2650    private void jMenuItemStopLoggingActionPerformed(java.awt.event.ActionEvent evt)
2651    {
2652        this.data.setStartLogging(false);
2653        this.lgh.closeLog();
2654        encoderThreadExecutor.shutdown();
2655        encoder.finishVideo();
2656        encoderThreadExecutor = null;
2657        encoder = null;
2658        jMenuLogger.setText("Not logging");
2659        jMenuItemStopLogging.setEnabled(false);
2660        jMenuItemStartLogging.setEnabled(true);
2661
2662        try {
2663            jMenuLogger.setIcon(new ImageIcon(ImageIO.read(getClass().getResource("/ntnusubsea/gui/Images/NotCalibrated.gif"))));
2664        } catch (IOException ex) {
2665            System.out.println("Error setting icon: " + ex.getMessage());
2666        }
2667    }
2668
2669    Action exitFullscreenAction = new AbstractAction() {
2670        public void actionPerformed(ActionEvent e) {
2671            exitFullscreenButton.doClick();
2672        }
2673    };
2674
2675 //    Action sendInputAction = new AbstractAction()
2676 //    {
2677 //        public void actionPerformed(ActionEvent e)
2678 //        {
2679 //            if (depthInputTextField.isFocusOwner())
2680 //            {
2681 //                //sendButton.doClick();
2682 //            }
2683 //
2684 //        }
2685 //    };
2686     /**
2687      * Checks if the given string is numeric
2688      *
2689      * @param string the given string to check
2690      * @return true if the given string is numeric, false if not
2691      */
2692     public static boolean isNumeric(String string) {
2693         boolean numeric = true;
2694         try {
```

```
2695            Double num = Double.parseDouble(string);
2696        } catch (NumberFormatException e) {
2697            numeric = false;
2698        }
2699        return numeric;
2700    }
2701
2702    /**
2703     * Checks if the given string is integer
2704     *
2705     * @param str the given string to check
2706     * @return true if the given string is integer, false if not
2707     */
2708    public static boolean isInteger(String str) {
2709        if (str == null) {
2710            return false;
2711        }
2712        if (str.isEmpty()) {
2713            return false;
2714        }
2715        int i = 0;
2716        if (str.charAt(0) == '-') {
2717            if (1 == str.length()) {
2718                return false;
2719            }
2720            i = 1;
2721        }
2722        for (; i < str.length(); i++) {
2723            char c = str.charAt(i);
2724            if (c < '0' || c > '9') {
2725                return false;
2726            }
2727        }
2728        return true;
2729    }
2730
2731    // Variables declaration - do not modify
2732    private javax.swing.JToggleButton InputControllerButton;
2733    private javax.swing.JSlider actuatorControlPS;
2734    private javax.swing.JSlider actuatorControlSB;
2735    private javax.swing.JLabel actuatorHeader1;
2736    private javax.swing.JLabel actuatorHeader2;
2737    private javax.swing.JProgressBar actuatorPSPosBar;
2738    private javax.swing.JLabel actuatorPSPosLabel;
2739    private javax.swing.JPanel actuatorPanel1;
2740    private javax.swing.JPanel actuatorPanel2;
2741    private javax.swing.JLabel actuatorPosLabel;
2742    private javax.swing.JProgressBar actuatorSBPosBar;
2743    private javax.swing.JLabel actuatorSBPosLabel;
2744    private javax.swing.JLabel actuatorSBPosLabel1;
2745    private javax.swing.JPanel background;
2746    private javax.swing.ButtonGroup buttonGroup1;
2747    private javax.swing.JMenuItem calibrateMenuItem;
2748    private javax.swing.JPanel cameraControlPanel;
2749    private javax.swing.JLabel cameraHeader;
2750    private javax.swing.JPanel cameraPanel;
2751    private javax.swing.JPanel cameraPanel1;
2752    private javax.swing.JLabel cameraPitchLabel;
2753    private javax.swing.JSlider cameraPitchSlider;
2754    private javax.swing.JFormattedTextField cameraPitchTextField;
2755    private javax.swing.JButton clearImagesButton;
2756    private javax.swing.JPanel controlPanel;
2757    private javax.swing.JFormattedTextField delayTextField;
2758    private javax.swing.JLabel depthHeader;
2759    private javax.swing.JRadioButton depthModeButton;
2760    private javax.swing.JPanel depthPanel;
2761    private javax.swing.JLabel emergencyHeader;
2762    private javax.swing.JPanel emergencyPanel;
2763    private javax.swing.JButton emergencyStopButton;
2764    private javax.swing.JButton exitFullscreenButton;
2765    private javax.swing.Box.Filler filler1;
2766    private javax.swing.Box.Filler filler2;
2767    private javax.swing.Box.Filler filler3;
2768    private javax.swing.JFrame fullscreen;
2769    private javax.swing.JButton fullscreenButton;
2770    private javax.swing.JButton getPhotosButton;
2771    private javax.swing.JLabel headingLabel;
2772    private javax.swing.JButton helpFrameOKbutton;
2773    private javax.swing.JFrame helpframe;
2774    private javax.swing.JLabel humidityLabel;
2775    private javax.swing.JLabel i2cErrorLabel;
2776    private javax.swing.JLabel imageNumberLabel;
2777    private javax.swing.JPanel infoPanel;
2778    private javax.swing.JLabel insideTempLabel;
2779    private javax.swing.JLabel jLabel1;
2780    private javax.swing.JLabel jLabel10;
2781    private javax.swing.JLabel jLabel11;
```

```
2782    private javax.swing.JLabel jLabel12;
2783    private javax.swing.JLabel jLabel15;
2784    private javax.swing.JLabel jLabel16;
2785    private javax.swing.JLabel jLabel17;
2786    private javax.swing.JLabel jLabel19;
2787    private javax.swing.JLabel jLabel2;
2788    private javax.swing.JLabel jLabel20;
2789    private javax.swing.JLabel jLabel21;
2790    private javax.swing.JLabel jLabel3;
2791    private javax.swing.JLabel jLabel4;
2792    private javax.swing.JLabel jLabel5;
2793    private javax.swing.JLabel jLabel6;
2794    private javax.swing.JLabel jLabel7;
2795    private javax.swing.JLabel jLabel8;
2796    private javax.swing.JLabel jLabel9;
2797    private javax.swing.JMenu jMenu1;
2798    private javax.swing.JMenu jMenu3;
2799    private javax.swing.JMenuItem jMenuAbout;
2800    private javax.swing.JMenuBar jMenuBar;
2801    private javax.swing.JMenu jMenuCalibrate;
2802    private javax.swing.JMenu jMenuConnect;
2803    private javax.swing.JMenuItem jMenuEchosounder;
2804    private javax.swing.JMenu jMenuHelp;
2805    private javax.swing.JMenuItem jMenuIOController;
2806    private javax.swing.JMenuItem jMenuItemConnect;
2807    private javax.swing.JMenuItem jMenuItemDisconnect;
2808    private javax.swing.JMenuItem jMenuItemExit;
2809    private javax.swing.JMenuItem jMenuItemStartLogging;
2810    private javax.swing.JMenuItem jMenuItemStopLogging;
2811    private javax.swing.JMenu jMenuLogger;
2812    private javax.swing.JMenuItem jMenuOptions;
2813    private javax.swing.JMenu jMenuPing;
2814    private javax.swing.JMenu jMenuRovReady;
2815    private javax.swing.JMenu jMenuTools;
2816    private javax.swing.JMenu jMenuVoltage;
2817    private javax.swing.JPanel jPanel1;
2818    private javax.swing.JPanel jPanel2;
2819    private javax.swing.JPanel jPanel3;
2820    private javax.swing.JPanel jPanel4;
2821    private javax.swing.JPanel jPanel5;
2822    private javax.swing.JPanel jPanel7;
2823    private javax.swing.JSeparator jSeparator1;
2824    private javax.swing.JSeparator jSeparator2;
2825    private javax.swing.JSeparator jSeparator4;
2826    private javax.swing.JSeparator jSeparator5;
2827    private javax.swing.JSeparator jSeparator6;
2828    private javax.swing.JSeparator jSeparator8;
2829    private javax.swing.JSeparator jSeparator9;
2830    private javax.swing.JLabel latitudeLabel;
2831    private javax.swing.JLabel leakLabel;
2832    private javax.swing.JLabel lightHeader;
2833    private javax.swing.JPanel lightPanel;
2834    private javax.swing.JSlider lightSlider;
2835    private javax.swing.JToggleButton lightSwitch;
2836    private javax.swing.JToggleButton lightSwitchBlueLED;
2837    private javax.swing.JLabel lightSwitch_lbl;
2838    private javax.swing.JToggleButton lockButton;
2839    private javax.swing.JLabel longitudeLabel;
2840    private javax.swing.JToggleButton manualControlButton;
2841    private javax.swing.JLabel outsideTempLabel;
2842    private javax.swing.JToggleButton photoModeButton;
2843    private javax.swing.JLabel photoModeDelayLabel;
2844    private javax.swing.JLabel photoModeDelay_FB_Label;
2845    private javax.swing.JLabel pitchLabel;
2846    private javax.swing.JLabel pressureLabel;
2847    private javax.swing.JButton resetManualControlButton;
2848    private javax.swing.JLabel rollLabel;
2849    private javax.swing.JLabel rovDepthLabel;
2850    private javax.swing.JLabel seafloorDepthBoatLabel;
2851    private javax.swing.JLabel seafloorDepthRovLabel;
2852    private javax.swing.JRadioButton seafloorModeButton;
2853    private javax.swing.JLabel setpointLabel;
2854    private javax.swing.JFormattedTextField targetDistanceTextField;
2855    private javax.swing.JLabel warningLabel1;
2856    private javax.swing.JLabel warningLabel2;
2857    private javax.swing.JPanel window;
2858    private javax.swing.JLabel wingLabel;
2859    // End of variables declaration
2860
2861    /**
2862     * Runs the ROV frame thread.
2863     */
2864    @Override
2865    public void run() {
2866        try {
2867            videoImage = ImageIO.read(getClass().getResource("/ntnusubsea/gui/Images/TowedROV.jpg"));
2868            data.setVideoImage(videoImage);
```

```
2869        } catch (IOException ex) {
2870            Logger.getLogger(NTNUSubseaGUI.class.getName()).log(Level.SEVERE, null, ex);
2871        }
2872        this.showImage(videoImage);
2873        this.setVisible(true); //To change body of generated methods, choose Tools | Templates.
2874        this.showImage(videoImage);
2875
2876 //      try {
2877 //          Thread.sleep(10);
2878 //      } catch (InterruptedException ex) {
2879 //          Logger.getLogger(NTNUSubseaGUI.class.getName()).log(Level.SEVERE, null, ex);
2880 //      }
2881    }
2882
2883    /**
2884     * Updates the GUI by observing the shared resource Data class.
2885     *
2886     * @param o
2887     * @param arg
2888     */
2889    @Override
2890    public void update(Observable o, Object arg) {
2891        //actuatorDutyCycleBar1.setValue(data.getBarValue());
2892        //System.out.println(data.getPitchAngle());
2893        if (data.getVideoImage() != null) {
2894            this.showImage(data.getVideoImage());
2895        }
2896        if (this.sounderThread == null) {
2897            this.sounderThread = new Thread(this.sounder);
2898        }
2899        if (data.isEmergencyMode() && !this.sounderThread.isAlive()) {
2900
2901            targetDistanceTextField.setText("0.00");
2902            setpointLabel.setText("EMERGENCY STOP: " + targetDistanceTextField.getText() + "m");
2903            setpointLabel.setBackground(new Color(255, 0, 0));
2904            manualControlButton.doClick();
2905            try {
2906                this.client_ROV.sendCommand("cmd_targetMode:2");
2907                this.client_ROV.sendCommand("cmd_actuatorPS:254");
2908                this.client_ROV.sendCommand("cmd_actuatorSB:254");
2909            } catch (IOException ex) {
2910                System.out.println("IOException in emergencyStopButtonActionPerformed: " + ex.getMessage());
2911            }
2912
2913            if (!this.sounder.isAlive()) {
2914                this.sounderThread = new Thread(this.sounder);
2915                this.sounderThread.setName("Sounder");
2916            }
2917            this.sounderThread.start();
2918        }
2919
2920        rollLabel.setText("Roll Angle: " + data.getRollAngle());
2921        pitchLabel.setText("Pitch Angle: " + data.getPitchAngle());
2922        wingLabel.setText("Wing Angle: " + data.getWingAngle());
2923
2924        seafloorDepthBoatLabel.setText("Beneath Boat: " + String.valueOf(df2.format(data.getDepthBeneathBoat())) + "m");
2925        seafloorDepthRovLabel.setText("Beneath ROV: " + data.getDepthBeneathRov() + "m");
2926        rovDepthLabel.setText("ROV Depth: " + data.getDepth() + "m");
2927
2928        headingLabel.setText("Heading: " + data.getGPSAngle());
2929        longitudeLabel.setText("Longitude: " + data.getLongitude());
2930        latitudeLabel.setText("Latitude: " + data.getLatitude());
2931
2932        actuatorPSPosLabel.setText("PS Position: " + data.getFb_actuatorPSPos());
2933        actuatorSBPosLabel.setText("SB Position: " + data.getFb_actuatorSBPos());
2934        actuatorPSPosBar.setValue(data.getFb_actuatorPSPos());
2935        actuatorSBPosBar.setValue(data.getFb_actuatorSBPos());
2936
2937        if (data.isI2cError()) {
2938            i2cErrorLabel.setText("I²C: ERROR!");
2939            i2cErrorLabel.setBackground(Color.red);
2940            data.setEmergencyMode(true);
2941        } else {
2942            i2cErrorLabel.setText("I²C: OK");
2943        }
2944
2945        if (data.getLeakStatus()) {
2946            leakLabel.setText("LEAK DETECTED!");
2947            leakLabel.setBackground(Color.red);
2948            data.setEmergencyMode(true);
2949        } else {
2950            leakLabel.setText("No leak detected");
2951            leakLabel.setBackground(new Color(39, 46, 54));
2952        }
2953        outsideTempLabel.setText("Outside Temp: " + data.getOutsideTemp() + " C");
2954        insideTempLabel.setText("Inside Temp: " + data.getInsideTemp() + "C");
2955        humidityLabel.setText("Rel. Humidity: " + data.getHumidity());
```

```
2956        pressureLabel.setText("Pressure: " + (data.getPressure() + " mBar"));
2957        rovDepthLabel.setText("ROV Depth: " + data.getRovDepth() + "m");
2958
2959        if (data.isRovReady()) {
2960            try {
2961                jMenuRovReady.setText("ROV Ready!");
2962                jMenuRovReady.setIcon(new ImageIcon(ImageIO.read(getClass().getResource("/ntnusubsea/gui/Images/Calibrated.gif"))));
2963            } catch (IOException ex) {
2964                System.out.println("Error: " + ex.getMessage());
2965            }
2966        } else {
2967            try {
2968                jMenuRovReady.setIcon(new ImageIcon(ImageIO.read(getClass().getResource("/ntnusubsea/gui/Images/NotCalibrated.gif"))));
2969            } catch (IOException ex) {
2970                System.out.println("Error: " + ex.getMessage());
2971            }
2972        }
2973
2974        if (data.getVoltage() < 28.00 && data.getVoltage() > 25.00) {
2975            try {
2976                jMenuVoltage.setIcon(new ImageIcon(ImageIO.read(getClass().getResource("/ntnusubsea/gui/Images/NotCalibrated.gif"))));
2977            } catch (IOException ex) {
2978                System.out.println("Error: " + ex.getMessage());
2979            }
2980            data.setEmergencyMode(true);
2981        } else if (data.getVoltage() > 28.00) {
2982            jMenuVoltage.setText("Voltage: " + data.getVoltage() + " V");
2983            try {
2984                jMenuVoltage.setIcon(new ImageIcon(ImageIO.read(getClass().getResource("/ntnusubsea/gui/Images/Calibrated.gif"))));
2985            } catch (IOException ex) {
2986                System.out.println("Error: " + ex.getMessage());
2987            }
2988        }
2989
2990        if (this.client_Pinger.isConnected() && (data.getRovPing() == 0.00)) {
2991            data.setEmergencyMode(true);
2992            jMenuPing.setText("Ping (ROV): Lost connection...");
2993            try {
2994                jMenuPing.setIcon(new ImageIcon(ImageIO.read(getClass().getResource("/ntnusubsea/gui/Images/NotCalibrated.gif"))));
2995            } catch (IOException ex) {
2996                System.out.println("Error: " + ex.getMessage());
2997            }
2998        } else if (this.client_Pinger.isConnected() && (data.getRovPing() != 999.99)) {
2999            jMenuPing.setText("Ping (ROV): " + String.valueOf(data.getRovPing()) + " ms");
3000            try {
3001                jMenuPing.setIcon(new ImageIcon(ImageIO.read(getClass().getResource("/ntnusubsea/gui/Images/Calibrated.gif"))));
3002            } catch (IOException ex) {
3003                System.out.println("Error: " + ex.getMessage());
3004            }
3005        }
3006
3007        photoModeDelay_FB_Label.setText(String.valueOf(df2.format(data.getPhotoModeDelay_FB())) + " s");
3008        imageNumberLabel.setText(data.getImageNumber() + " / 1000");
3009
3010 //       actuatorControlPS.setValue(data.getFb_actuatorPSPos);
3011 //       actuatorControlSB.setValue(data.getFb_actuatorSBPos);
3012        actuatorPosLabel.setText("<html>PS: " + String.valueOf(actuatorControlPS.getValue()) + "<br/><br/>SB: " + String.valueOf(actuatorControlSB.getValue()
3013
3014    }
3015 }
3016
```

D:\Dokumenter\Skule\04 - NTNU\Bachelor\Github\TowedROV_GUI\src\basestation_rov\SerialDataHandler.java

```
1  /*
2   * This code is for the bachelor thesis named "Towed-ROV".
3   * The purpose is to build a ROV which will be towed behind a surface vessel
4   * and act as a multi-sensor platform, were it shall be easy to place new
5   * sensors. There will also be a video stream from the ROV.
6   *
7   * The system consists of two Raspberry Pis in the ROV that is connected to
8   * several Arduino micro controllers. These micro controllers are connected to
9   * feedback from the actuators, the echo sounder and extra optional sensors.
10  * The external computer which is on the surface vessel is connected to a GPS,
11  * echo sounder over USB, and the ROV over ethernet. It will present and
12  * log data in addition to handle user commands for controlling the ROV.
13  */
14 package basestation_rov;
15
16 import ntnusubsea.gui.Data;
17 import java.util.HashMap;
18 import java.util.Map.Entry;
19 import jssc.SerialPort;
20 import jssc.SerialPortList;
21
22 /**
23  * Responsible for finding and storing the com ports connected.
24  */
25 public class SerialDataHandler {
26
27     private HashMap<String, String> portNamesList = new HashMap<>();
28     String comPort = "";
29     SerialPort serialPort;
30     Data data;
31     String start_char = "<";
32     String end_char = ">";
33     String sep_char = ":";
34     int comCheck = 0;
35
36     /**
37      * The constructor of the SerialDataHandler
38      *
39      * @param data the shared resource Data class
40      */
41     public SerialDataHandler(Data data) {
42         this.data = data;
43     }
44
45     /**
46      * Initiates the com ports. Not finished
47      */
48     public void initiateComPorts() {
49
50     }
51
52     /**
53      * Saves the usable com ports found.
54      */
```

```
55  private void saveUsableComPorts() {
56      for (Entry e : portNamesList.entrySet()) {
57          String comPortKey = (String) e.getKey();
58          String comPortValue = (String) e.getValue();
59          if (!comPortValue.contains("Unknown")) {
60              data.comPortList.put(comPortKey, comPortValue);
61              comCheck++;
62          }
63      }
64      if (comCheck < 3) {
65          //Not all comports was found
66          System.out.println("ERROR: Not all com ports was found, trying again...");
67          findComPorts();
68      }
69  }
70
71  /**
72   * Finds the com ports available.
73   */
74  public void findComPorts() {
75      int baudrate = 0;
76      int searchRuns = 0;
77      while (searchRuns != 2) {
78          String[] portNames = getAvailableComPorts();
79          for (int i = 0; i < portNames.length; i++) {
80              if (portNames[i].contains("COM")) {
81                  portNamesList.put(portNames[i], "Unknown");
82              }
83          }
84
85          if (searchRuns == 0) {
86              baudrate = 115200;
87          }
88          if (searchRuns == 1) {
89              baudrate = 4800;
90          }
91
92          for (Entry e : portNamesList.entrySet()) {
93              String comPortKey = (String) e.getKey();
94              String comPortValue = (String) e.getValue();
95
96              if (comPortValue.contains("Unknown")) {
97                  serialPort = new SerialPort(comPortKey);
98              }
99              try {
100                 serialPort.openPort();
101                 serialPort.setParams(baudrate, 8, 1, 0);
102                 String buffer = "";
103                 Thread.sleep(5000);
104                 buffer = serialPort.readString();
105
106                 if (buffer != null) {
107                     if (buffer.contains("<") && buffer.contains(">")) {
108                         buffer = buffer.substring(buffer.indexOf(start_char) + 1);
109                         buffer = buffer.substring(0, buffer.indexOf(end_char));
110                         // buffer = buffer.replace("?", "");
111                         String[] data = buffer.split(sep_char);
112
```

```
113                     for (int i = 0; i < data.length; i = i + 2) {
114                         if (data[i].contains("Roll")) {
115                             String key = (String) e.getKey();
116                             portNamesList.put(key, "IMU");
117                         }
118                         if (data[i].contains("GPS")) {
119                             String key = (String) e.getKey();
120                             portNamesList.put(key, "GPS");
121                         }
122                         if (data[i].contains("EchoSounder")) {
123                             String key = (String) e.getKey();
124                             portNamesList.put(key, "EchoSounder");
125                         }
126                         if (data[i].contains("ROVDummy") || data[i].contains("Test")) {
127                             String key = (String) e.getKey();
128                             portNamesList.put(key, "ROVDummy");
129                         }
130                     }
131                 }
132             }
133             serialPort.closePort();
134         } catch (Exception ex) {
135             System.out.println("Error: " + ex);
136             portNamesList.put(comPortKey, "Unreadable");
137             try {
138                 serialPort.closePort();
139 
140             } catch (Exception exe) {
141                 System.out.println("Error: Failed to close port " + exe);
142             }
143         }
144     }
145     saveUsableComPorts();
146     searchRuns++;
147   }
148 }
149 
150 /**
151  * Returns the available com ports
152  *
153  * @return the available com ports
154  */
155 private String[] getAvailableComPorts() {
156     // getting serial ports list into the array
157     String[] portNames = SerialPortList.getPortNames();
158 
159     if (portNames.length == 0) {
160         System.out.println("No com ports is connected...");
161         try {
162             System.in.read();
163         } catch (Exception e) {
164             // TODO Auto-generated catch block
165             e.printStackTrace();
166         }
167     }
168 //      for (int i = 0; i < portNames.length; i++)
169 //      {
170 //          System.out.println(portNames[i]);
```

```
171 //        }
172         return portNames;
173     }
174 }
175
```

D:\Dokumenter\Skule\04 - NTNU\Bachelor\Github\TowedROV_GUI\src\ntnusubsea\gui\Sounder.java

```
1  /*
2   * This code is for the bachelor thesis named "Towed-ROV".
3   * The purpose is to build a ROV which will be towed behind a surface vessel
4   * and act as a multi-sensor platform, were it shall be easy to place new
5   * sensors. There will also be a video stream from the ROV.
6   *
7   * The system consists of two Raspberry Pis in the ROV that is connected to
8   * several Arduino micro controllers. These micro controllers are connected to
9   * feedback from the actuators, the echo sounder and extra optional sensors.
10  * The external computer which is on the surface vessel is connected to a GPS,
11  * echo sounder over USB, and the ROV over ethernet. It will present and
12  * log data in addition to handle user commands for controlling the ROV.
13  */
14 package ntnusubsea.gui;
15
16 import java.io.FileInputStream;
17 import java.io.InputStream;
18 import sun.audio.AudioPlayer;
19 import sun.audio.AudioStream;
20
21 /**
22  * Responsible for playing an audio file representing an alarm. This class is
23  * needed for the GUI not to freeze while loading and playing the audio file.
24  */
25 public class Sounder implements Runnable {
26
27     private boolean isAlive = false;
28
29     /**
30      * The constructor of the Sounder class.
31      */
32     public Sounder() {
33         this.isAlive = true;
34     }
35
36     /**
37      * Runs the Sounder thread. Plays the audio file.
38      */
39     @Override
40     public void run() {
41         try {
42             String gongFile = "src/ntnusubsea/gui/audio/Emergency_Warning_06.wav";
43             InputStream in = new FileInputStream(gongFile);
44             AudioStream audioStream = new AudioStream(in);
45             AudioPlayer.player.start(audioStream);
46             Thread.sleep(15000);
47
48         } catch (Exception e) {
49             System.out.println("Error while trying to play an audio file: " + e.getMessage());
50         }
51         this.setAlive(false);
52     }
53
54     /**
55      * Sets the state of the thread.
```

```
56      *
57      * @param bool the status to set
58      */
59     public void setAlive(boolean bool) {
60         this.isAlive = bool;
61     }
62
63     /**
64      * Returns the alive status of the thread
65      *
66      * @return the thread alive status
67      */
68     public boolean isAlive() {
69         return this.isAlive;
70     }
71
72 }
73
```

D:\Dokumenter\Skule\04 - NTNU\Bachelor\Github\TowedROV_GUI\src\basestation_rov\calibrationClasses\StartupCalibration.java

```
1  /*
2   * This code is for the bachelor thesis named "Towed-ROV".
3   * The purpose is to build a ROV which will be towed behind a surface vessel
4   * and act as a multi-sensor platform, were it shall be easy to place new
5   * sensors. There will also be a video stream from the ROV.
6   *
7   * The system consists of two Raspberry Pis in the ROV that is connected to
8   * several Arduino micro controllers. These micro controllers are connected to
9   * feedback from the actuators, the echo sounder and extra optional sensors.
10  * The external computer which is on the surface vessel is connected to a GPS,
11  * echo sounder over USB, and the ROV over ethernet. It will present and
12  * log data in addition to handle user commands for controlling the ROV.
13  */
14 package basestation_rov.calibrationClasses;
15
16 import ntnusubsea.gui.*;
17
18 /**
19  * Responsible for calibrating the necessary equipment at start-up. Not
20  * finished.
21  */
22 public class StartupCalibration {
23
24     private static Thread actuatorCalibrationThread;
25     Data data = null;
26     TCPClient client_ROV = null;
27     TCPClient client_Camera = null;
28     long currentTime = 0;
29     long lastTimePS = 0;
30     long lastTimeSB = 0;
31
32     /**
33      * The constructor of the StartupCalibration class.
34      *
35      * @param data the shared resource Data class
36      * @param client_ROV the ROV TCP client
37      */
38     public StartupCalibration(Data data, TCPClient client_ROV) {
39         this.data = data;
40         this.client_ROV = client_ROV;
41         this.client_Camera = client_Camera;
42
43     }
44
45     /**
46      * Calibrates all the necessary equipment.
47      *
48      * @return a message saying the calibration was completed.
49      */
50     public String doStartupCalibration() {
51         calibrateActuators();
52         //testLights();
53
54         return "Calibration complete...";
```

```
55    }
56
57    /**
58     * Sets up the com ports
59     */
60    public void setupComPorts() {
61
62    }
63
64    /**
65     * Calibrates the actuators
66     */
67    public void calibrateActuators() {
68        actuatorCalibrationThread = new Thread(new ActuatorCalibration(data, client_ROV));
69        actuatorCalibrationThread.start();
70        actuatorCalibrationThread.setName("actuatorCalibrationThread");
71    }
72
73 //   Test lights
74 //   public void testLights()
75 //   {
76 //       boolean testingLights = true;
77 //       long lastTime = System.nanoTime();
78 //       int lightIntensity = 1;
79 //       while (testingLights && lightIntensity < 100)
80 //       {
81 //           if (System.nanoTime() - lastTime >= 250000000)
82 //           {
83 //               dh.cmd_lightIntensity = lightIntensity + 1;
84 //               lastTime = System.nanoTime();
85 //           }
86 //       }
87 //       try
88 //       {
89 //           Thread.sleep(5000);
90 //           dh.cmd_lightIntensity = 0;
91 //       } catch (Exception e)
92 //       {
93 //       }
94 //   }
95 //
96 //Calibrate depth sensor here:
97 }
98
```

D:\Dokumenter\Skule\04 - NTNU\Bachelor\Github\TowedROV_GUI\src\ntnusubsea\gui\TCPClient.java

```
1  /*
2   * This code is for the bachelor thesis named "Towed-ROV".
3   * The purpose is to build a ROV which will be towed behind a surface vessel
4   * and act as a multi-sensor platform, were it shall be easy to place new
5   * sensors. There will also be a video stream from the ROV.
6   *
7   * The system consists of two Raspberry Pis in the ROV that is connected to
8   * several Arduino micro controllers. These micro controllers are connected to
9   * feedback from the actuators, the echo sounder and extra optional sensors.
10  * The external computer which is on the surface vessel is connected to a GPS,
11  * echo sounder over USB, and the ROV over ethernet. It will present and
12  * log data in addition to handle user commands for controlling the ROV.
13  */
14 package ntnusubsea.gui;
15
16 import java.io.*;
17 import java.net.*;
18 import java.util.HashMap;
19 import java.util.Map;
20
21 /**
22  * Client class that handles the connection to the server, retrieves the video
23  * stream and sends commands to the server
24  */
25 public class TCPClient implements Runnable {
26
27     boolean connectionResetError = false;
28     private boolean connected = false;
29     private static String sentence;
30     private static String serverResponse;
31     private Socket clientSocket;
32     private InputStream inputStream;
33     private OutputStream outputStream;
34     private int port;
35     private String IP;
36     private Data data;
37
38     String start_char = "<";
39     String end_char = ">";
40     String sep_char = ":";
41
42     BufferedReader inFromServer;
43     PrintWriter outToServer;
44
45     /**
46      * The constructor of the TCPClient.
47      *
48      * @param IP the given IP to connect to
49      * @param port the given port to connect to
50      * @param data the shared resource Data class
51      */
52     public TCPClient(String IP, int port, Data data) {
53         this.data = data;
54         this.port = port;
55         this.IP = IP;
56     }
57
58     /**
59      * Runs the TCPClient thread. Connects to the TCP server, and reconnects if
```

```
60      * the connection is lost.
61      */
62     @Override
63     public void run() {
64         while (!this.connected) {
65             try {
66                 this.connect(this.IP, this.port);
67             } catch (Exception e1) {
68                 //System.out.println("Could not connect to server (" + this.IP + ":" + this.port + "): " + e1.getMessage());
69                 long sec = 5000;
70                 //System.out.println("Trying to reconnect in " + sec + " ms...");
71                 try {
72                     Thread.sleep(sec);
73                 } catch (Exception e2) {
74                 }
75             }
76         }
77 
78         boolean finished = false;
79 
80         while (!finished) {
81             try {
82                 //Run
83 
84                 if (this.connectionResetError && !isConnected()) {
85                     int sec = 5000;
86                     System.out.println("Trying to reconnect (" + this.IP + ":" + this.port + ") in " + sec + " sec...");
87                     Thread.sleep(sec);
88                     this.connect(this.IP, this.port);
89                 }
90             } catch (SocketTimeoutException ex) {
91                 System.out.println("Error: Read timed out");
92                 this.connectionResetError = true;
93                 this.connected = false;
94             } catch (SocketException ex) {
95                 System.out.println("An error occured: Connection reset");
96                 this.connectionResetError = true;
97                 this.connected = false;
98             } catch (Exception e) {
99                 System.out.println("An error occured: " + e);
100                 this.connectionResetError = true;
101                 this.connected = false;
102             }
103         }
104 
105     }
106 
107     /**
108      * Sends a command to the server.
109      *
110      * @param cmd
111      * @throws IOException Throws IOException if client is disconnected
112      */
113     public synchronized void sendCommand(String cmd) throws IOException {
114         try {
115             if (isConnected()) {
116 
117                 String commandString = "<" + cmd + ">";
118                 outToServer.println(commandString);
119 //                System.out.println("Cmd sent: " + commandString);
120 
121                 if (cmd.contains("actuator")) {
```

```
122             System.out.println("Actuator Cmd sent: " + commandString);
123         }
124
125         outToServer.flush();
126
127         String serverResponse = inFromServer.readLine();
128         if (serverResponse.contains("not ready")) {
129             System.out.println("Server not ready!");
130         } else {
131             //System.out.println("Server response: " + serverResponse);
132             if (cmd.equals("fb_allData") || cmd.equals("getData")) {
133                 HashMap<String, String> newDataList = new HashMap<>();
134                 String key = "";
135                 String value = "";
136                 if (serverResponse.contains("<") && serverResponse.contains(">")) {
137                     serverResponse = serverResponse.substring(serverResponse.indexOf(start_char) + 1);
138                     serverResponse = serverResponse.substring(0, serverResponse.indexOf(end_char));
139                     serverResponse = serverResponse.replace("?", "");
140                     if (serverResponse.contains(":")) {
141                         String[] dataArray = serverResponse.split(sep_char);
142                         for (int i = 0; i < dataArray.length; i += 2) {
143                             newDataList.put(dataArray[i], dataArray[i + 1]);
144                         }
145                     } else {
146                         System.out.println(serverResponse);
147                     }
148
149                 } else {
150                     System.out.println("The data string which was received was not complete...");
151                 }
152
153                 this.handleDataFromRemote(newDataList);
154             }
155         }
156
157     } else {
158         System.out.println("Command not sent: Not connected to server");
159     }
160
161 } catch (SocketTimeoutException ex) {
162     System.out.println("Error: Read timed out");
163     this.connectionResetError = true;
164     this.connected = false;
165 } catch (SocketException ex) {
166     System.out.println("An error occured: Connection reset");
167     this.connectionResetError = true;
168     this.connected = false;
169 } catch (Exception e) {
170     System.out.println("An error occured: " + e);
171     this.connectionResetError = true;
172     this.connected = false;
173 }
174
175 }
176
177 /**
178  * Returns the connection status of the socket
179  *
180  * @return The connection status of the socket
181  */
182 public boolean isConnected() {
183     return connected;
```

```
184     }
185
186     /**
187      * Connects the client to the server through a socket and saves the IP and
188      * port to the global variables IP and port
189      *
190      * @param IP IP of the server to connect to
191      * @param port
192      * @throws IOException Throws an IOException when the connection is
193      * unsuccessful
194      */
195     public void connect(String IP, int port) throws IOException {
196         clientSocket = new Socket(IP, port);
197 //          this.inputStream = clientSocket.getInputStream();
198 //          this.outputStream = clientSocket.getOutputStream();
199         outToServer = new PrintWriter(
200                 clientSocket.getOutputStream(), true);
201         inFromServer = new BufferedReader(new InputStreamReader(
202                 clientSocket.getInputStream()));
203         System.out.println("Success! Connected to server " + this.IP + ":" + this.port);
204         this.connected = true;
205         this.connectionResetError = false;
206     }
207
208     /**
209      * Closes the socket if the client is currently connected
210      *
211      * @throws IOException Throws IOException if there is a problem with the
212      * connection
213      */
214     public void disconnect() throws IOException {
215         if (clientSocket != null) {
216             clientSocket.close();
217         }
218         connected = false;
219     }
220
221     /**
222      * Sends the given data string to the TCP server.
223      *
224      * @param sentence the given string
225      * @return the server response
226      */
227     public synchronized String sendData(String sentence) {
228         try {
229             outToServer.println(sentence);
230             //System.out.println("Data is sent...");
231             outToServer.flush();
232             serverResponse = inFromServer.readLine();
233
234         } catch (Exception e) {
235         }
236
237         return serverResponse;
238
239     }
240
241     /**
242      * Sends a ping command to the TCP server.
243      *
244      * @return the ping of the connection
245      */
```

```
246   public String ping() {
247       double elapsedTimer = 0;
248       double elapsedTimerNano = 0;
249       long lastTime = 0;
250
251       String ping = "<Ping:null>";
252       lastTime = System.nanoTime();
253       String serverResponse = sendData("ping");
254       if (serverResponse.equals("<ping:true>")) {
255           elapsedTimerNano = (System.nanoTime() - lastTime);
256           elapsedTimer = elapsedTimerNano / 1000000;
257           System.out.println("<Ping: " + elapsedTimer + ">");
258
259           elapsedTimer = 0;
260       } else {
261
262           ping = "<ping:null>";
263       }
264
265       return ping;
266   }
267
268   /**
269    * Returns the port of the server
270    *
271    * @return the port of the server
272    */
273   public int getPort() {
274       return port;
275   }
276
277   /**
278    * Sets the port of the server
279    *
280    * @param port the port of the server
281    */
282   public void setPort(int port) {
283       this.port = port;
284   }
285
286   /**
287    * Returns the IP of the server
288    *
289    * @return the IP of the server
290    */
291   public String getIP() {
292       return IP;
293   }
294
295   /**
296    * Sets the IP of the server
297    *
298    * @param IP the IP of the server
299    */
300   public void setIP(String IP) {
301       this.IP = IP;
302   }
303
304   /**
305    * Compare keys to control values coming in from remote, and puts the
306    * correct value to correct variable in the shared resource Data class.
307    *
```

```
308     * @param newDataList the data list
309     */
310    public void handleDataFromRemote(HashMap<String, String> newDataList) {
311        for (Map.Entry e : newDataList.entrySet()) {
312            String key = (String) e.getKey();
313            String value = (String) e.getValue();
314
315            switch (key) {
316                // From ROV RPi:
317                case "Fb_actuatorPSPos":
318                    data.setFb_actuatorPSPos(Integer.parseInt(value));
319                    break;
320                case "Fb_actuatorSBPos":
321                    data.setFb_actuatorSBPos(Integer.parseInt(value));
322                    break;
323                case "Fb_rollAngle":
324                    data.setRollAngle(Double.parseDouble(value));
325                    break;
326                case "Fb_pitchAngle":
327                    data.setPitchAngle(Double.parseDouble(value));
328                    break;
329 //                 case "Fb_depthToSeabedEcho":
330 //                     data.setDepthBeneathRov(Double.parseDouble(value));
331 //                     break;
332                case "Fb_depthBelowTransduser":
333                    data.setDepthBeneathRov(Double.parseDouble(value));
334                    break;
335                case "Fb_depthBeneathROV":
336                    data.setDepthBeneathRov(Double.parseDouble(value));
337                    break;
338                case "Fb_tempElBoxFront":
339                    data.setFb_tempElBoxFront(Double.parseDouble(value));
340                    break;
341                case "Fb_tempElBoxRear":
342                    data.setFb_tempElBoxRear(Double.parseDouble(value));
343                    break;
344                case "Fb_ROVReady":
345                    data.setRovReady(Boolean.parseBoolean(value));
346                    break;
347                case "ERROR_I2C":
348                    data.setI2cError(Boolean.parseBoolean(value));
349                    break;
350
351                // From Camera RPi:
352                case "leakAlarm":
353                    if (value.equals("1")) {
354                        data.setLeakStatus(true);
355                    } else if (equals("0")) {
356                        data.setLeakStatus(false);
357                    }
358                    break;
359                case "depth":
360                    data.setRovDepth(Double.parseDouble(value));
361                    break;
362                case "pressure":
363                    data.setPressure(Double.parseDouble(value));
364                    break;
365                case "outsideTemp":
366                    data.setOutsideTemp(Double.parseDouble(value));
367                    break;
368                case "insideTemp":
369                    data.setInsideTemp(Double.parseDouble(value));
```

```
370                 break;
371             case "humidity":
372                 data.setHumidity(Double.parseDouble(value));
373                 break;
374         }
375     }
376 }
377 }
378
```

D:\Dokumenter\Skule\04 - NTNU\Bachelor\Github\TowedROV_GUI\src\ntnusubsea\gui\TCPpinger.java

```
1 /*
2  * This code is for the bachelor thesis named "Towed-ROV".
3  * The purpose is to build a ROV which will be towed behind a surface vessel
4  * and act as a multi-sensor platform, were it shall be easy to place new
5  * sensors. There will also be a video stream from the ROV.
6  *
7  * The system consists of two Raspberry Pis in the ROV that is connected to
8  * several Arduino micro controllers. These micro controllers are connected to
9  * feedback from the actuators, the echo sounder and extra optional sensors.
10  * The external computer which is on the surface vessel is connected to a GPS,
11  * echo sounder over USB, and the ROV over ethernet. It will present and
12  * log data in addition to handle user commands for controlling the ROV.
13  */
14 package ntnusubsea.gui;
15
16 import java.io.*;
17 import java.net.*;
18 import java.util.logging.Level;
19 import java.util.logging.Logger;
20
21 /**
22  * Client class that handles the connection to the server, retrieves the video
23  * stream and sends commands to the server
24  *
25  */
26 public class TCPpinger implements Runnable {
27
28     boolean connectionResetError = false;
29     private boolean connected = false;
30     private static String sentence;
31     private static String serverResponse;
32     private Socket clientSocket;
33     private InputStream inputStream;
34     private OutputStream outputStream;
35     private int port;
36     private String IP;
37     private Data data;
38
39     BufferedReader inFromServer;
40     PrintWriter outToServer;
41
42     /**
43      * The constructor of the TCPClient.
44      *
45      * @param IP the given IP to connect to
46      * @param port the given port to connect to
47      * @param data the shared resource Data class
48      */
49     public TCPpinger(String IP, int port, Data data) {
50         this.data = data;
51         this.port = port;
52         this.IP = IP;
53     }
54
55     /**
```

```
56   * Runs the TCPpinger thread. Connects to the TCP server, and reconnects if
57   * the connection is lost.
58   */
59  @Override
60  public void run() {
61      if (!this.isConnected())
62      {
63          try {
64              this.connect(this.IP, this.port);
65          } catch (IOException ex) {
66              Logger.getLogger(TCPpinger.class.getName()).log(Level.SEVERE, null, ex);
67          }
68      }
69      if (this.isConnected()) {
70          data.setRovPing(this.getPing());
71          //System.out.println("Ping (ROV): " + data.getRovPing());
72      }
73  }
74
75  /**
76   * Connects the client to the server through a socket and saves the IP and
77   * port to the global variables IP and port
78   *
79   * @param IP IP of the server to connect to
80   * @param port
81   * @throws IOException Throws an IOException when the connection is
82   * unsuccessful
83   */
84  public void connect(String IP, int port) throws IOException {
85      clientSocket = new Socket(IP, port);
86      clientSocket.setSoTimeout(3000);
87      outToServer = new PrintWriter(
88              clientSocket.getOutputStream(), true);
89      inFromServer = new BufferedReader(new InputStreamReader(
90              clientSocket.getInputStream()));
91      System.out.println("Success! Connected to server " + this.IP + ":" + this.port);
92      this.connected = true;
93      this.connectionResetError = false;
94  }
95
96  /**
97   * Closes the socket if the client is currently connected
98   *
99   * @throws IOException Throws IOException if there is a problem with the
100   * connection
101   */
102  public void disconnect() throws IOException {
103      if (clientSocket != null) {
104          clientSocket.close();
105      }
106      connected = false;
107  }
108
109  /**
110   * Sends a ping command to the TCP server.
111   *
112   * @return the ping of the connection
113   */
```

```
114  public Double getPing() {
115      double pingValue = 0.00;
116      double elapsedTimer = 0;
117      double elapsedTimerNano = 0;
118      long lastTime = 0;
119
120      String ping = "<Ping:null>";
121      lastTime = System.nanoTime();
122      String serverResponse = sendData("ping");
123      if (serverResponse != null && serverResponse.equals("<ping:true>")) {
124          elapsedTimerNano = (System.nanoTime() - lastTime);
125          elapsedTimer = elapsedTimerNano / 1000000;
126          pingValue = elapsedTimer;
127          //System.out.println("<Ping: " + elapsedTimer + ">");
128
129          elapsedTimer = 0;
130      } else {
131          pingValue = 0.00;
132          ping = "<Ping:null>";
133      }
134      return pingValue;
135  }
136
137  /**
138   * Sends the given data string to the TCP server.
139   *
140   * @param sentence the given string
141   * @return the server response
142   */
143  public String sendData(String sentence) {
144      try {
145          outToServer.println(sentence);
146          //System.out.println("Data is sent...");
147          outToServer.flush();
148          serverResponse = inFromServer.readLine();
149
150      } catch (SocketTimeoutException ste) {
151          System.out.println("SocketTimeoutException: " + ste.getMessage());
152      } catch (Exception e) {
153          System.out.println("Exception: " + e.getMessage());
154      }
155      return serverResponse;
156  }
157
158  /**
159   * Returns the connection status of the socket
160   *
161   * @return The connection status of the socket
162   */
163  public boolean isConnected() {
164      return connected;
165  }
166
167  /**
168   * Returns the port of the server
169   *
170   * @return the port of the server
171   */
```

```
172     public int getPort() {
173         return port;
174     }
175 
176     /**
177      * Sets the port of the server
178      *
179      * @param port the port of the server
180      */
181     public void setPort(int port) {
182         this.port = port;
183     }
184 
185     /**
186      * Returns the IP of the server
187      *
188      * @return the IP of the server
189      */
190     public String getIP() {
191         return IP;
192     }
193 
194     /**
195      * Sets the IP of the server
196      *
197      * @param IP the IP of the server
198      */
199     public void setIP(String IP) {
200         this.IP = IP;
201     }
202 }
203 
```

D:\Dokumenter\Skule\04 - NTNU\Bachelor\Github\TowedROV_GUI\src\ntnusubsea\gui\UDPServer.java

```
1 /*
2  * This code is for the bachelor thesis named "Towed-ROV".
3  * The purpose is to build a ROV which will be towed behind a surface vessel
4  * and act as a multi-sensor platform, were it shall be easy to place new
5  * sensors. There will also be a video stream from the ROV.
6  *
7  * The system consists of two Raspberry Pis in the ROV that is connected to
8  * several Arduino micro controllers. These micro controllers are connected to
9  * feedback from the actuators, the echo sounder and extra optional sensors.
10  * The external computer which is on the surface vessel is connected to a GPS,
11  * echo sounder over USB, and the ROV over ethernet. It will present and
12  * log data in addition to handle user commands for controlling the ROV.
13  */
14 package ntnusubsea.gui;
15
16 import java.awt.image.BufferedImage;
17 import java.io.ByteArrayInputStream;
18 import java.io.IOException;
19 import java.net.DatagramPacket;
20 import java.net.DatagramSocket;
21 import java.net.SocketException;
22 import javax.imageio.ImageIO;
23 import java.io.File;
24 import java.net.InetAddress;
25
26 /**
27  * This class handles incoming images from a DatagramPacket. It receives the
28  * image on a DatagramSocket and returns it as a BufferedImage.
29  *
30  */
31 public class UDPServer implements Runnable {
32
33     static BufferedImage videoImage;
34     //static Socket videoSocket;
35     private int photoNumber = 1;
36     boolean lastPhotoMode = false;
37     private boolean debug = false;
38     private Data data;
39     private int test = 0;
40     //private String IP;
41     private int port;
42     private DatagramSocket videoSocket;
43     private long timer = System.currentTimeMillis();
44     private File photoDirectory;
45     private DatagramPacket receivePacket;
46     private InetAddress returnIP;
47     private int returnPort;
48     private boolean connected = false;
49     private double endTime;
50     private double startTime;
51
52     /**
53      * The constructor of the UDPServer class. Sets up a DatagramSocket at the
54      * given port.
55      *
56      * @param port the given port
57      * @param data the shared resource class Data
58      */
59     public UDPServer(int port, Data data) {
60         try {
61             this.data = data;
62             this.port = port;
63             //this.IP = IP;
64             videoSocket = new DatagramSocket(this.port);
65             photoDirectory = new File("C://TowedROV/ROV_Photos/");
66         } catch (Exception e) {
67             System.out.println("Error setting up UDP server: " + e.getMessage());
68         }
69     }
70
71     /**
```

```
72      * Sends the photo mode delay value to the UDP client
73      */
74     public void sendDelayCommand() {
75        try {
76           String message = "photoDelay:" + String.valueOf(data.getPhotoModeDelay());
77           byte arr[] = message.getBytes();
78           DatagramPacket sendPacket = new DatagramPacket(arr, arr.length, this.returnIP, this.returnPort);
79           videoSocket.send(sendPacket);
80           System.out.println("Delay command sent to Camera RPi!");
81
82        } catch (SocketException ex) {
83           System.out.println("SocketException in UDPServer: " + ex.getMessage());
84
85        } catch (IOException ex) {
86           System.out.println("IOException in UDPServer: " + ex.getMessage());
87        } catch (Exception ex) {
88           System.out.println("Exception in UDPServer: " + ex.getMessage());
89        }
90     }
91
92     /**
93      * Sends the reset image number command to the UDP client
94      */
95     public void sendResetIMGcommand() {
96        try {
97           String message = "resetImgNumber";
98           byte arr[] = message.getBytes();
99           DatagramPacket sendPacket = new DatagramPacket(arr, arr.length, this.returnIP, this.returnPort);
100           videoSocket.send(sendPacket);
101           System.out.println("resetImgNumber command sent to Camera RPi!");
102
103        } catch (SocketException ex) {
104           System.out.println("SocketException in UDPServer: " + ex.getMessage());
105
106        } catch (IOException ex) {
107           System.out.println("IOException in UDPServer: " + ex.getMessage());
108        } catch (Exception ex) {
109           System.out.println("Exception in UDPServer: " + ex.getMessage());
110        }
111     }
112
113     /**
114      * Runs the UDPServer thread. Receives the image frames from the video
115      * stream.
116      */
117     @Override
118     public void run() {
119        try {
120           if (System.currentTimeMillis() - timer > 60000) {
121              videoSocket.close();
122              videoSocket = new DatagramSocket(this.port);
123              timer = System.currentTimeMillis();
124              //System.out.println("Reconnected");
125              this.connected = true;
126              this.data.setStreaming(true);
127           }
128
129           //Creates new DatagramSocket for reciving DatagramPackets
130           //Creating new DatagramPacket form the packet recived on the videoSocket
131           byte[] receivedData = new byte[60000];
132           receivePacket = new DatagramPacket(receivedData,
133                 receivedData.length);
134           this.connected = true;
135           this.data.setStreaming(true);
136           if (receivePacket.getLength() > 0) {
137              startTime = System.currentTimeMillis();
138              //Updates the videoImage from the received DatagramPacket
139              videoSocket.receive(receivePacket);
140              this.returnIP = receivePacket.getAddress();
141              this.returnPort = receivePacket.getPort();
142              endTime = System.currentTimeMillis();
143              data.setPhotoModeDelay_FB((endTime - startTime) / 1000);
144              if (debug) {
```

```
145                 System.out.println("Videopackage received");
146             }
147             //Reads incomming byte array into a BufferedImage
148             ByteArrayInputStream bais = new ByteArrayInputStream(receivedData);
149             videoImage = ImageIO.read(bais);
150             data.setVideoImage(videoImage);
151 
152             if (lastPhotoMode && (endTime - startTime) > 500) {
153                 data.increaseImageNumberByOne();
154             }
155 
156             // Saves the photo to disk if photo mode is true
157 //            if (data.isPhotoMode())
158 //            {
159 //                try
160 //                {
161 //                    if (this.photoDirectory.exists() && this.photoDirectory.isDirectory())
162 //                    {
163 //                        ImageIO.write(videoImage, "jpg", new File(this.photoDirectory.toString() + "/image" + this.photoNumber + ".png"));
164 //                        this.photoNumber++;
165 //                    } else
166 //                    {
167 //                        System.out.println("No directory found, creating a new one at C://TowedROV/ROV_Photos/");
168 //                        this.photoDirectory.mkdir();
169 //                    }
170 //
171 //                } catch (Exception e)
172 //                {
173 //                    System.out.println("Exception occured :" + e.getMessage());
174 //                }
175 //                System.out.println("Image were saved to disk succesfully at C://TowedROV/ROV_Photos");
176 //            }
177             // Sends the command to the ROV
178             if (data.isPhotoMode() != lastPhotoMode) {
179                 String message = "photoMode:" + String.valueOf(data.isPhotoMode());
180                 byte arr[] = message.getBytes();
181                 DatagramPacket sendPacket = new DatagramPacket(arr, arr.length, this.returnIP, this.returnPort);
182                 videoSocket.send(sendPacket);
183                 lastPhotoMode = data.isPhotoMode();
184             }
185 
186             receivedData = null;
187             bais = null;
188             test++;
189             //System.out.println(endTime - startTime);
190         }
191 
192     } catch (SocketException sex) {
193         System.out.println("SocketException: " + sex.getMessage());
194         this.connected = false;
195         this.data.setStreaming(false);
196 
197     } catch (IOException ioex) {
198         System.out.println("IOException: " + ioex.getMessage());
199         this.connected = false;
200         this.data.setStreaming(false);
201     } catch (Exception ex) {
202         System.out.println("Exception: " + ex.getMessage());
203         this.connected = false;
204         this.data.setStreaming(false);
205     }
206 }
207 }
208 
```

D:\Dokumenter\Skule\04 - NTNU\Bachelor\Github\TowedROV_GUI\src\ntnusubsea\gui\VideoEncoder.java

```
1  /*
2   * This code is for the bachelor thesis named "Towed-ROV".
3   * The purpose is to build a ROV which will be towed behind a surface vessel
4   * and act as a multi-sensor platform, were it shall be easy to place new
5   * sensors. There will also be a video stream from the ROV.
6   *
7   * The system consists of two Raspberry Pis in the ROV that is connected to
8   * several Arduino micro controllers. These micro controllers are connected to
9   * feedback from the actuators, the echo sounder and extra optional sensors.
10   * The external computer which is on the surface vessel is connected to a GPS,
11   * echo sounder over USB, and the ROV over ethernet. It will present and
12   * log data in addition to handle user commands for controlling the ROV.
13   */
14  package ntnusubsea.gui;
15
16  import java.awt.image.BufferedImage;
17  import java.io.File;
18  import java.time.LocalDateTime;
19  import java.util.Observable;
20  import java.util.Observer;
21  import org.jcodec.api.awt.AWTSequenceEncoder;
22
23  /**
24   * The class video encoder uses buffered image and encodes there images to a
25   * video file.
26   *
27   */
28  public class VideoEncoder implements Runnable, Observer {
29  // private ArrayList<BufferedImage> list = new ArrayList();
30
31      private BufferedImage videoImage;
32      private Data data;
33      private AWTSequenceEncoder enc;
34      private int frame = 0;
35      private LocalDateTime startTime;
36
37      /**
38       * Creates an instance of video encoder and create the MP4 file and gives it
39       * a unique name
40       *
41       * @param data Data containing the images
42       */
43      public VideoEncoder(Data data) {
44          try {
45              startTime = LocalDateTime.now();
46              String minute, hour, day, month, year;
47              if (startTime.getMinute() < 10) {
48                  minute = "0" + Integer.toString(startTime.getMinute());
49              } else {
50                  minute = Integer.toString(startTime.getMinute());
51              }
52              if (startTime.getHour() < 10) {
53                  hour = "0" + Integer.toString(startTime.getHour());
54              } else {
```

```
55              hour = Integer.toString(startTime.getHour());
56          }
57          if (startTime.getDayOfMonth() < 10) {
58              day = "0" + Integer.toString(startTime.getDayOfMonth());
59          } else {
60              day = Integer.toString(startTime.getDayOfMonth());
61          }
62          if (startTime.getMonthValue() < 10) {
63              month = "0" + Integer.toString(startTime.getMonthValue());
64          } else {
65              month = Integer.toString(startTime.getMonthValue());
66          }
67          year = Integer.toString(startTime.getYear());
68
69          File dir = null;
70          try {
71              dir = new File("C:\\TowedROV\\ROV_Video\\");
72              if (!dir.exists() || !dir.isDirectory()) {
73                  System.out.println("No directory found, creating a new one at C://TowedROV/ROV_Video/");
74                  dir.mkdir();
75              }
76          } catch (Exception e) {
77              System.out.println("No directory found, creating a new one at C://TowedROV/ROV_Video/");
78              dir.mkdir();
79          }
80
81          String fileName = dir.getPath() + "\\ROV Video" + hour + minute + " " + day + "."
82                  + month + "." + year + ".mp4";
83          enc = AWTSequenceEncoder.create24Fps(new File(fileName));
84          this.data = data;
85      } catch (Exception ex) {
86          System.out.println("Exception while starting video encoder: " + ex.getMessage());
87      }
88  }
89
90  /**
91   * Runs the VideoEncoder thread and encodes each frame.
92   */
93  @Override
94  public void run() {
95      if (data.isStreaming()) {
96          // if (frame < list.size()) {
97          try {
98              // BufferedImage image = list.get(frame);
99              enc.encodeImage(videoImage);
100              // frame++;
101          } catch (Exception ex) {
102              System.out.println("Error encoding frame: " + ex.getMessage());;
103          }
104      }
105
106  }
107
108  /**
109   * Updates the video image to encode by observing the Data class.
110   *
111   * @param o
112   * @param arg
```

```
113     */
114    @Override
115    public void update(Observable o, Object arg) {
116        videoImage = data.getVideoImage();
117    }
118
119    /**
120     * Finishes the video
121     */
122    public void finishVideo() {
123        try {
124            enc.finish();
125        } catch (Exception ex) {
126            System.out.println("Exception while finishing the video: " + ex.getMessage());
127        }
128    }
129 }
130
```

K ROV Java code

D:\Dokumenter\Skule\04 - NTNU\Bachelor\Github\TowedROV\src\ROV\AlarmSystem\AlarmHandler.java

```
1 /*
2  * This code is for the bachelor thesis named "Towed-ROV".
3  * The purpose is to build a ROV which will be towed behind a surface vessel
4  * and act as a multi-sensor platform, were it shall be easy to place new
5  * sensors. There will also be a video stream from the ROV.
6  *
7  * The system consists of two Raspberry Pis in the ROV that is connected to
8  * several Arduino micro controllers. These micro controllers are connected to
9  * feedback from the actuators, the echo sounder and extra optional sensors.
10  * The external computer which is on the surface vessel is connected to a GPS,
11  * echo sounder over USB, and the ROV over ethernet. It will present and
12  * log data in addition to handle user commands for controlling the ROV.
13  */
14 package ROV.AlarmSystem;
15
16 import ROV.*;
17 import java.util.HashMap;
18 import java.util.Map.Entry;
19 import java.util.concurrent.ConcurrentHashMap;
20
21 /**
22  * The alarm handler of the ROV is responsible for handling any alarms boolean
23  * alarms or time based alarms on the ROV.
24  *
25  *
26  * Not yet implemented
27  */
28 public class AlarmHandler implements Runnable {
29
30 //    long lastTime = 0;
31 //    long elapsedTime = 0;
32 //
33 //    Data dh = null;
34 //    private ConcurrentHashMap<String, Boolean> listOfBooleanAlarms = new ConcurrentHashMap<>();
35 //    private ConcurrentHashMap<String, Boolean> listOfTimerAlarms = new ConcurrentHashMap<>();
36 //    // Input data for the alarms to deside if the alarm is in alarm state or not
37 //
38 //    /**
39 //     * A list over the necesarry data for the alarms to function.
40 //     */
41 //    public ConcurrentHashMap<String, Integer> alarmDataList = new ConcurrentHashMap<>();
42 //    // List over all active or dormant alarms
43 //
44 //    /**
45 //     *
46 //     */
47 //    public ConcurrentHashMap<String, Boolean> completeAlarmList = new ConcurrentHashMap<>();
48 //    private ConcurrentHashMap<String, Boolean> lastCompleteAlarmList = new ConcurrentHashMap<>();
49 //
50 //    /**
51 //     * A list over all inhibited alarms
52 //     */
53 //    public ConcurrentHashMap<String, Boolean> inhibitedAlarms = new ConcurrentHashMap<>();
54 //
55 //    /**
56 //     *
```

```
57 //     */
58 //    public boolean ack = false;
59 //
60 //    /**
61 //     *
62 //     */
63 //    public boolean inhibit_waterLeakSensor_1_Alarm = false;
64 //
65 //    /**
66 //     * Timebased alarms
67 //     *
68 //     * @param dh
69 //     */
70 //    TimeBasedAlarms sbActuatorFeedbackAlarm;
71 //
72 //    /**
73 //     *
74 //     * @param dh
75 //     */
76 //    BooleanBasedAlarms waterLeakSensor_1_Alarm;
77 //
78 //    /**
79 //     *
80 //     * @param dh
81 //     */
82 //    public AlarmHandler(Data dh)
83 //    {
84 //        this.dh = dh;
85 //    }
86 //
87 //    /**
88 //     *
89 //     */
90     public void run() {
91 //        fillAlarmListWithAlarms();
92 //        fillCompleteAlarmList();
93 //
94 //        updateAlarmInputData();
95 //        initiateAlarmThreads();
96 //
97 //        lastCompleteAlarmList.putAll(completeAlarmList);
98 //        inhibitedAlarms.putAll(completeAlarmList);
99 //        updateDataHandlerWithAlarms();
100 //
101 //        while (true)
102 //        {
103 //            if (System.nanoTime() - lastTime >= 250000000)
104 //            {
105 //                checkForAck();
106 //
107 //                lastTime = System.nanoTime();
108 //            }
109 //
110 //            updateAlarmInputData();
111 //
112 //            if (!completeAlarmList.equals(lastCompleteAlarmList))
113 //            {
114 //                System.out.println("An alarm has been changed");
```

```
115 //             lastCompleteAlarmList.putAll(completeAlarmList);
116 //             updateDataHandlerWithAlarms();
117 //         } else
118 //         {
119 //             // System.out.println("Nothing");
120 //             //lastCompleteAlarmList.putAll(completeAlarmList);
121 //         }
122 //     }
123 // }
124 //
125 // private boolean isAck()
126 // {
127 //     return ack;
128 // }
129 //
130 // private void setAck(boolean ack)
131 // {
132 //     this.ack = ack;
133 // }
134 //
135 // private void checkForAck()
136 // {
137 //     if (dh.isCmd_ack() && !ack)
138 //     {
139 //         System.out.println("Ack is True");
140 //         ack = true;
141 //         //setAck(true);
142 //     }
143 //     if (!dh.isCmd_ack() && ack)
144 //     {
145 //         ack = false;
146 //         //setAck(false);
147 //     }
148 // }
149 //
150 // private void fillCompleteAlarmList()
151 // {
152 //     for (Entry e : listOfBooleanAlarms.entrySet())
153 //     {
154 //         String key = (String) e.getKey();
155 //         Boolean value = (Boolean) e.getValue();
156 //         completeAlarmList.put(key, value);
157 //     }
158 //     for (Entry e : listOfTimerAlarms.entrySet())
159 //     {
160 //         String key = (String) e.getKey();
161 //         Boolean value = (Boolean) e.getValue();
162 //         completeAlarmList.put(key, value);
163 //     }
164 // }
165 //
166 // private void fillAlarmListWithAlarms()
167 // {
168 //     //Boolean alarms
169 //     listOfBooleanAlarms.put("waterLeakSensor_1_Alarm", false);
170 //
171 //     //Timer alarms
172 //     listOfTimerAlarms.put("sbActuatorFeedbackAlarm", false);
```

```
173 //    }
174 //
175 //    private void initiateAlarmThreads()
176 //    {
177 //        //Boolean Based Alarms
178 //        waterLeakSensor_1_Alarm = new BooleanBasedAlarms(dh,
179 //                this, "fb_waterLeakChannel_1", 1, "waterLeakSensor_1_Alarm",
180 //                true, ack, inhibit_waterLeakSensor_1_Alarm);
181 //
182 //        //Create threads
183 //        Thread waterLeakSensor_1_Alarm_Thread = new Thread(waterLeakSensor_1_Alarm);
184 //        Thread sbActuatorFeedbackAlarm_Thread = new Thread(sbActuatorFeedbackAlarm);
185 //
186 //        //Name threads
187 //        waterLeakSensor_1_Alarm_Thread.setName("waterLeakSensor_1_Alarm_Thread");
188 //        sbActuatorFeedbackAlarm_Thread.setName("sbActuatorFeedbackAlarm");
189 //
190 //        //Start threads
191 //        sbActuatorFeedbackAlarm_Thread.start();
192 //        waterLeakSensor_1_Alarm_Thread.start();
193 //    }
194 //
195 //    private void updateAlarmInputData()
196 //    {
197 ////        alarmDataList.put("fb_waterLeakChannel_1", dh.isFb_waterLeakChannel_1() ? 1 : 0);
198 //    }
199 //
200 //    private void updateDataHandlerWithAlarms()
201 //    {
202 //        dh.completeAlarmListDh.putAll(completeAlarmList);
203 //    }
204 //
205 ////    private void buildalarmListFrom()
206 ////    {
207 ////        for (Entry e : alarmDataList.entrySet())
208 ////        {
209 ////            String key = (String) e.getKey();
210 ////            alarmList.put(key, false);
211 ////        }
212 ////
213     }
214 }
215
```

D:\Dokumenter\Skule\04 - NTNU\Bachelor\Github\TowedROV\src\ROV\AlarmSystem\BooleanBasedAlarms.java

```
1 /*
2  * This code is for the bachelor thesis named "Towed-ROV".
3  * The purpose is to build a ROV which will be towed behind a surface vessel
4  * and act as a multi-sensor platform, were it shall be easy to place new
5  * sensors. There will also be a video stream from the ROV.
6  *
7  * The system consists of two Raspberry Pis in the ROV that is connected to
8  * several Arduino micro controllers. These micro controllers are connected to
9  * feedback from the actuators, the echo sounder and extra optional sensors.
10  * The external computer which is on the surface vessel is connected to a GPS,
11  * echo sounder over USB, and the ROV over ethernet. It will present and
12  * log data in addition to handle user commands for controlling the ROV.
13  */
14 package ROV.AlarmSystem;
15
16 import ROV.Data;
17
18 /**
19  * Creates boolean alarms for the ROV
20  *
21  *
22  * Not yet implemented
23  */
24 public class BooleanBasedAlarms implements Runnable {
25
26 //    long currentTime = 0;
27 //    long lastTime = 0;
28 //
29 //    String input;
30 //    int setPoint;
31 //    Boolean alarm;
32 //    boolean HAlarm;
33 //    boolean ack;
34 //    boolean inhibit;
35 //    String alarmName;
36 //    Data dh;
37 //    AlarmHandler alarmHandler;
38 //
39 //
40 //    public BooleanBasedAlarms(Data dh, AlarmHandler alarmHandler, String input, int setPoint, String alarmName,
41 //            boolean HAlarm, boolean ack, boolean inhibit)
42 //    {
43 //        this.alarmHandler = alarmHandler;
44 //        this.dh = dh;
45 //        this.input = input;
46 //        this.setPoint = setPoint;
47 //
48 //        this.alarmName = alarmName;
49 //        this.HAlarm = HAlarm;
50 //        this.ack = ack;
51 //        this.inhibit = inhibit;
52 //
53 //        currentTime = System.nanoTime();
54 //
55 //    }
56     public void run() {
57 //
58 //        while (true)
59 //        {
60 //            while (!alarmHandler.inhibitedAlarms.get(alarmName))
```

```
61 //          {
62 //             if (HAlarm && alarmHandler.alarmDataList.get(input) >= setPoint)
63 //             {
64 //                // High Alarm
65 //                alarm = true;
66 //
67 //                alarmHandler.completeAlarmList.put(alarmName, alarm);
68 //                //dh.handleDataFromAlarmList(alarmName, alarm);
69 //                while (alarm)
70 //                {
71 //                   try
72 //                   {
73 //                      Thread.sleep(10);
74 //                   } catch (Exception e)
75 //                   {
76 //                   }
77 //                   //System.out.println("waiting...");
78 //                   if (alarmHandler.ack)
79 //                   {
80 //                      //System.out.println("test");
81 //                      alarm = false;
82 //                      alarmHandler.completeAlarmList.put(alarmName, alarm);
83 //                      System.out.println("Alarm is acked");
84 //                      //dh.handleDataFromAlarmList(alarmName, alarm);
85 //                   }
86 //                }
87 //             }
88 //             if (!HAlarm && alarmHandler.alarmDataList.get(input) <= setPoint)
89 //             {
90 //                // Low Alarm
91 //                alarm = true;
92 //                alarmHandler.completeAlarmList.put(alarmName, alarm);
93 //                //dh.handleDataFromAlarmList(alarmName, alarm);
94 //                while (alarm)
95 //                {
96 //                   if (alarmHandler.ack)
97 //                   {
98 //                      alarm = false;
99 //                      alarmHandler.completeAlarmList.put(alarmName, alarm);
100 //                      System.out.println("Alarm is acked");
101 //                      //dh.handleDataFromAlarmList(alarmName, alarm);
102 //                   }
103 //                }
104 //             }
105 //          }
106 //
107 //       }
108    }
109
110 }
111
```

D:\Dokumenter\Skule\04 - NTNU\Bachelor\Github\TowedROV\src\ROV\Data.java

```
1 /*
2  * This code is for the bachelor thesis named "Towed-ROV".
3  * The purpose is to build a ROV which will be towed behind a surface vessel
4  * and act as a multi-sensor platform, were it shall be easy to place new
5  * sensors. There will also be a video stream from the ROV.
6  *
7  * The system consists of two Raspberry Pis in the ROV that is connected to
8  * several Arduino micro controllers. These micro controllers are connected to
9  * feedback from the actuators, the echo sounder and extra optional sensors.
10  * The external computer which is on the surface vessel is connected to a GPS,
11  * echo sounder over USB, and the ROV over ethernet. It will present and
12  * log data in addition to handle user commands for controlling the ROV.
13  */
14 package ROV;
15
16 import java.util.HashMap;
17 import java.util.Map;
18 import java.util.Observable;
19 import java.util.concurrent.ConcurrentHashMap;
20
21 /**
22  * This class is used for storing and sharing all variables and resources
23  * between the threads
24  */
25 public class Data extends Observable {
26
27     //Internal variables for ROV
28     boolean ERROR_I2C = false;
29     String dataToSend = "";
30     boolean gatheringDataToSend = false;
31     int actuatorDifference = 0;
32
33     // Calibration values
34     int pressureSensorOffset = 0;
35     private final static int PS_ACTUATOR_ANGLEADJUST = 0;
36     private final static int SB_ACTUATOR_ANGLEADJUST = 0;
37
38     // Command values
39     int cmd_lightMode = 0;
40     int cmd_actuatorPS = 0;
41     int cmd_actuatorSB = 0;
42     int cmd_bothActuators = 0;
43 //     int cmd_actuatorPSMaxPos = 0;
44 //     int cmd_actuatorPSMinPos = 0;
45 //     int cmd_actuatorSBMaxPos = 0;
46 //     int cmd_actuatorSBMinPos = 0;
47     int cmd_BlueLED = 0;
48
49     boolean cmd_disableMotors = true;
50
51     int cmd_pressureAtSeaLevel = 0;
52
53     int cmd_targetMode = 0; // Mode 0 = depth, 1 = seafloor, 2 = manual
54     double cmd_targetDistance = 0;
55
```

```
56    double cmd_offsetDepthBeneathROV = 0;
57    double cmd_offsetROVdepth = 0;
58
59    //int cmd_depth = 0;
60 //   int cmd_cameraPitch = 0;
61 //   int cmd_cameraRoll = 0;
62 //   byte cmd_cameraMode = 0;
63    double cmd_pid_p = 0;
64    double cmd_pid_i = 0;
65    double cmd_pid_d = 0;
66    double cmd_pid_gain = 0;
67
68    boolean cmd_emergencySurface = false;
69    boolean cmd_ack = false;
70 //   boolean cmd_manualWingControl = false;
71
72    double cmd_imuCalibrateRoll = 0;
73    double cmd_imuCalibratePitch = 0;
74
75    boolean cmd_ping = false;
76    boolean clientConnected = false;
77
78    // Sensor values
79    boolean fb_ROVReady = false;
80
81    double fb_depthBeneathROV = 12;
82    double fb_depthBeneathBoat = 0;
83    double cmd_currentROVdepth = 0;
84
85    int fb_speedThroughWather = 0;
86    int fb_waterTemperature = 0;
87
88    int fb_actuatorPSPos = 0;
89    int fb_actuatorSBPos = 0;
90
91    double fb_tempMainElBoxFront = 0;
92    double fb_tempMainElBoxRear = 0;
93
94    int fb_currentDraw = 0;
95    double fb_pitchAngle = 0;
96    double fb_rollAngle = 0;
97
98    int fb_heading = 0;
99
100   //Input channels
101   double analogInputChannel_1 = 0.00;
102   double analogInputChannel_2 = 0.00;
103
104   boolean digitalInputChannel_3 = false;
105   boolean digitalInputChannel_4 = false;
106
107   //Class variables
108   private int counter = 0;
109   private boolean i2cRequest = false;
110
111   /**
112    * A list over availible com ports
113    */
```

```
114  public HashMap<String, String> comPortList = new HashMap<>();
115
116  /**
117   * A list over all internal alarms in the ROV
118   */
119  public ConcurrentHashMap<String, Boolean> completeAlarmListDh = new ConcurrentHashMap<>();
120
121  /**
122   * Returns the current ack status
123   *
124   * @return ack status
125   */
126  public boolean isCmd_ack() {
127      return cmd_ack;
128  }
129
130  /**
131   * Sets the current ack status
132   *
133   * @param cmd_ack the current ack status
134   */
135  public void setCmd_ack(boolean cmd_ack) {
136      this.cmd_ack = cmd_ack;
137  }
138
139  //Sensor values getters and setters
140  /**
141   * Returns the depth beneath the ROV in meters
142   *
143   * @return depth beneath the ROV in meters
144   */
145  public double getFb_depthBeneathROV() {
146      return fb_depthBeneathROV;
147  }
148
149  /**
150   * Sets the depth beneath the ROV in meters
151   *
152   * @param fb_depthBeneathROV the depth beneath the ROV in meters
153   */
154  public void setFb_depthBeneathROV(double fb_depthBeneathROV) {
155      this.fb_depthBeneathROV = fb_depthBeneathROV;
156  }
157
158  /**
159   * Returns the depth below the boat in meters
160   *
161   * @return the depth below the boat in meters
162   */
163  public double getFb_depthBeneathBoat() {
164      return fb_depthBeneathBoat;
165  }
166
167  /**
168   * Sets the depth beaneath the boat
169   *
170   * @param fb_depthBeneathBoat the depth beaneath the boat
171   */
```

```
172     public void setFb_depthBeneathBoat(double fb_depthBeneathBoat) {
173         this.fb_depthBeneathBoat = fb_depthBeneathBoat;
174     }
175 
176     /**
177      * Returns the speed through water
178      *
179      * @return the speed through water
180      */
181     public int getFb_speedThroughWather() {
182         return fb_speedThroughWather;
183     }
184 
185     /**
186      * Sets the speed through water
187      *
188      * @param fb_speedThroughWather the speed through water
189      */
190     public void setFb_speedThroughWather(int fb_speedThroughWather) {
191         this.fb_speedThroughWather = fb_speedThroughWather;
192     }
193 
194     /**
195      * Returns the water temperature
196      *
197      * @return the water temperature
198      */
199     public int getFb_waterTemperature() {
200         return fb_waterTemperature;
201     }
202 
203     /**
204      * Sets the water temperature
205      *
206      * @param fb_waterTemperature the water temperature
207      */
208     public void setFb_waterTemperature(int fb_waterTemperature) {
209         this.fb_waterTemperature = fb_waterTemperature;
210     }
211 
212     /**
213      * Returns the PS actuator position
214      *
215      * @return the PS actuator position
216      */
217     public int getFb_actuatorPSPos() {
218 
219         return fb_actuatorPSPos;
220     }
221 
222     /**
223      * Sets the PS actuator position
224      *
225      * @param fb_actuatorPSPos the PS actuator position
226      */
227     public void setFb_actuatorPSPos(int fb_actuatorPSPos) {
228 //        setChanged();
229 //        notifyObservers();
```

```
230         this.fb_actuatorPSPos = fb_actuatorPSPos + PS_ACTUATOR_ANGLEADJUST;
231     }
232
233     /**
234      * Returns the SB actuator feedback
235      *
236      * @return the SB actuator feedback
237      */
238     public int getFb_actuatorSBPos() {
239         return fb_actuatorSBPos;
240     }
241
242     /**
243      * Sets the SB actuator position
244      *
245      * @param fb_actuatorSBPos the SB actuator position
246      */
247     public void setFb_actuatorSBPos(int fb_actuatorSBPos) {
248 //          setChanged();
249 //          notifyObservers();
250         this.fb_actuatorSBPos = fb_actuatorSBPos + SB_ACTUATOR_ANGLEADJUST;
251
252     }
253
254     /**
255      * Returns the temperature in the front of the main electronic box
256      *
257      * @return the temperature in the front of the main electronic box
258      */
259     public double getFb_tempMainElBoxFront() {
260         return fb_tempMainElBoxFront;
261     }
262
263     /**
264      * Sets the temperature in front of the the main electronic box
265      *
266      * @param fb_tempMainElBoxFront the temperature in front of the in the main
267      * electronic box
268      */
269     public void setFb_tempMainElBoxFront(double fb_tempMainElBoxFront) {
270         this.fb_tempMainElBoxFront = fb_tempMainElBoxFront;
271     }
272
273     /**
274      * Returns the remperature in the rear of the main electronic box
275      *
276      * @return the remperature in the rear of the main electronic box
277      */
278     public double getFb_tempMainElBoxRear() {
279         return fb_tempMainElBoxRear;
280     }
281
282     /**
283      * Sets the remperature in the rear of the main electronic box
284      *
285      * @param fb_tempMainElBoxRear the remperature in the rear of the main
286      * electronic box
287      */
```

```
288     public void setFb_tempMainElBoxRear(double fb_tempMainElBoxRear) {
289         this.fb_tempMainElBoxRear = fb_tempMainElBoxRear;
290     }
291 
292     /**
293      * Returns the current draw
294      *
295      * @return the current draw
296      */
297     public int getFb_currentDraw() {
298         return fb_currentDraw;
299     }
300 
301     /**
302      * Sets the current draw
303      *
304      * @param fb_currentDraw the current draw
305      */
306     public void setFb_currentDraw(int fb_currentDraw) {
307         this.fb_currentDraw = fb_currentDraw;
308     }
309 
310     /**
311      * Returns the pitch andgle of the ROV
312      *
313      * @return the pitch andgle of the ROV
314      */
315     public double getFb_pitchAngle() {
316         return fb_pitchAngle + getCmd_imuCalibratePitch();
317     }
318 
319     /**
320      * Sets the pitch andgle of the ROV
321      *
322      * @param fb_pitchAngle the pitch andgle of the ROV
323      *
324      */
325     public void setFb_pitchAngle(double fb_pitchAngle) {
326 //        setChanged();
327 //        notifyObservers();
328         this.fb_pitchAngle = fb_pitchAngle;
329     }
330 
331     /**
332      * Returns the roll angle of the ROV
333      *
334      * @return the roll angle of the ROV
335      */
336     public double getFb_rollAngle() {
337 
338         return fb_rollAngle + getCmd_imuCalibrateRoll();
339     }
340 
341     /**
342      * Sets the roll angle of the ROV
343      *
344      * @param fb_rollAngle the roll angle of the ROV
345      */
```

```
346     public void setFb_rollAngle(double fb_rollAngle) {
347 //      setChanged();
348 //      notifyObservers();
349         this.fb_rollAngle = fb_rollAngle;
350     }
351 
352     /**
353      * Returns the heading of the ROV
354      *
355      * @return the heading of the ROV
356      */
357     public int getFb_heading() {
358         return fb_heading;
359     }
360 
361     /**
362      * Sets the heading of the ROV
363      *
364      * @param fb_heading the heading of the ROV
365      */
366     public void setFb_heading(int fb_heading) {
367         this.fb_heading = fb_heading;
368     }
369 
370     /**
371      * Returns the light mode of the ROV
372      *
373      * @return the light mode of the ROV
374      */
375     public int getCmd_lightMode() {
376         return cmd_lightMode;
377     }
378 
379     /**
380      * Sets the light mode of the ROV
381      *
382      * @param cmd_lightMode the light mode of the ROV
383      */
384     public void setCmd_lightMode(int cmd_lightMode) {
385         this.cmd_lightMode = this.cmd_lightMode;
386     }
387 
388     /**
389      * Return the PS actuator target command
390      *
391      * @return the PS actuator target command
392      */
393     public int getCmd_actuatorPS() {
394         return cmd_actuatorPS;
395     }
396 
397     /**
398      * Sets the PS actuator target command and notifiy the observers
399      *
400      * @param cmd_actuatorPS the PS actuator target command
401      */
402     public void setCmd_actuatorPS(int cmd_actuatorPS) {
403         this.cmd_actuatorPS = cmd_actuatorPS;
```

```
404         setChanged();
405         notifyObservers();
406 
407     }
408 
409     /**
410      * Return the SB actuator target command
411      *
412      * @return the SB actuator target command
413      */
414     public int getCmd_actuatorSB() {
415         return cmd_actuatorSB;
416     }
417 
418     /**
419      * Sets the SB actuator target command and notifiy the observers
420      *
421      * @param cmd_actuatorSB the SB actuator target command and notifiy the
422      * observers
423      */
424     public void setCmd_actuatorSB(int cmd_actuatorSB) {
425         this.cmd_actuatorSB = cmd_actuatorSB;
426         setChanged();
427         notifyObservers();
428     }
429 
430     /**
431      * Return the command target mode
432      *
433      * @return the command target mode
434      */
435     public int getcmd_targetMode() {
436         return cmd_targetMode;
437     }
438 
439     /**
440      * Sets the command target mode and notifiy the observers
441      *
442      * @param cmd_targetMode the command target mode and notifiy the observers
443      */
444     public void setcmd_targetMode(int cmd_targetMode) {
445         this.cmd_targetMode = cmd_targetMode;
446         setChanged();
447         notifyObservers();
448     }
449 
450     /**
451      * Returns the command value for the PID P value
452      *
453      * @return the command value for the PID P value
454      */
455     public double getCmd_pid_p() {
456         return cmd_pid_p;
457     }
458 
459     /**
460      * Sets the command value for the PID P value
461      *
```

```
462     * @param cmd_pid_p the command value for the PID P value
463     */
464    public void setCmd_pid_p(double cmd_pid_p) {
465        this.cmd_pid_p = cmd_pid_p;
466    }
467
468    /**
469     * Return the command value for the PID I value
470     *
471     * @return the command value for the PID I value
472     */
473    public double getCmd_pid_i() {
474        return cmd_pid_i;
475    }
476
477    /**
478     * Sets the command value for the PID I value
479     *
480     * @param cmd_pid_i the command value for the PID I value
481     */
482    public void setCmd_pid_i(double cmd_pid_i) {
483        this.cmd_pid_i = cmd_pid_i;
484    }
485
486    /**
487     * Sets the command value for the PID D value
488     *
489     * @return the command value for the PID D value
490     */
491    public double getCmd_pid_d() {
492        return cmd_pid_d;
493    }
494
495    /**
496     * Sets the command value for the PID D value
497     *
498     * @param cmd_pid_d the command value for the PID D value
499     */
500    public void setCmd_pid_d(double cmd_pid_d) {
501        this.cmd_pid_d = cmd_pid_d;
502    }
503
504    /**
505     * Return the command value for the PID gain value
506     *
507     * @return the command value for the PID gain value
508     */
509    public double getCmd_pid_gain() {
510        return cmd_pid_gain;
511    }
512
513    /**
514     * Sets the command value for the PID gain value
515     *
516     * @param cmd_pid_gain the command value for the PID gain value
517     */
518    public void setCmd_pid_gain(double cmd_pid_gain) {
519        this.cmd_pid_gain = cmd_pid_gain;
```

```
520     }
521
522     /**
523      * Return the emergency surface value of the ROV
524      *
525      * @return the emergency surface value of the ROV
526      */
527     public boolean isCmd_emergencySurface() {
528         return cmd_emergencySurface;
529     }
530
531     /**
532      * Sets the emergency surface value of the ROV
533      *
534      * @param cmd_emergencySurface the emergency surface value of the ROV
535      */
536     public void setCmd_emergencySurface(boolean cmd_emergencySurface) {
537         this.cmd_emergencySurface = cmd_emergencySurface;
538     }
539
540     /**
541      * Return the ROV current depth
542      *
543      * @return the ROV current depth
544      */
545     public double getCmd_currentROVdepth() {
546         return cmd_currentROVdepth;
547     }
548
549     /**
550      * Sets the ROV current depth
551      *
552      * @param cmd_currentROVdepth the ROV current depth
553      */
554     public void setCmd_currentROVdepth(double cmd_currentROVdepth) {
555         this.cmd_currentROVdepth = cmd_currentROVdepth + cmd_offsetDepthBeneathROV;
556         setChanged();
557         notifyObservers();
558
559     }
560
561     /**
562      * Returns the counter value
563      *
564      * @return the counter value
565      */
566     public int getCounter() {
567         return counter;
568     }
569
570     /**
571      * Sets the counter value
572      *
573      * @param counter the counter value
574      */
575     public void setCounter(int counter) {
576         this.counter = counter;
577     }
```

```
578
579     /**
580      * Return the value of analog input channel 1
581      *
582      * @return the value of analog input channel 1
583      */
584     public double getAnalogInputChannel_1() {
585         return analogInputChannel_1;
586     }
587
588     /**
589      * Sets the value of analog input channel 1
590      *
591      * @param analogInputChannel_1 the value of analog input channel 1
592      */
593     public void setAnalogInputChannel_1(double analogInputChannel_1) {
594         this.analogInputChannel_1 = analogInputChannel_1;
595     }
596
597     /**
598      *
599      * Return the value of analog input channel 2
600      *
601      * @return the value of analog input channel 2
602      */
603     public double getAnalogInputChannel_2() {
604         return analogInputChannel_2;
605     }
606
607     /**
608      * Sets the value of analog input channel 2
609      *
610      * @param analogInputChannel_2 the value of analog input channel 2
611      */
612     public void setAnalogInputChannel_2(double analogInputChannel_2) {
613         this.analogInputChannel_2 = analogInputChannel_2;
614     }
615
616     /**
617      * Return the value of digital input 3
618      *
619      * @return the value of digital input 3
620      */
621     public boolean isDigitalInputChannel_3() {
622         return digitalInputChannel_3;
623     }
624
625     /**
626      * Sets the value of digital input 3
627      *
628      * @param digitalInputChannel_3 the value of digital input 3
629      */
630     public void setDigitalInputChannel_3(boolean digitalInputChannel_3) {
631         this.digitalInputChannel_3 = digitalInputChannel_3;
632     }
633
634     /**
635      * Return the value of digital input 4
```

```
636     *
637     * @return the value of digital input 4
638     */
639    public boolean isDigitalInputChannel_4() {
640        return digitalInputChannel_4;
641    }
642
643    /**
644     * Sets the value of digital input 4
645     *
646     * @param digitalInputChannel_4 the value of digital input 4
647     */
648    public void setDigitalInputChannel_4(boolean digitalInputChannel_4) {
649        this.digitalInputChannel_4 = digitalInputChannel_4;
650    }
651
652    /**
653     * Return the IMU Roll calobratio value
654     *
655     * @return the IMU Roll calobratio value
656     */
657    public double getCmd_imuCalibrateRoll() {
658        return cmd_imuCalibrateRoll;
659    }
660
661    /**
662     * Sets the IMU Roll calobratio value
663     *
664     * @param cmd_imuCalibrateRoll the IMU Roll calobratio value
665     */
666    public void setCmd_imuCalibrateRoll(double cmd_imuCalibrateRoll) {
667        this.cmd_imuCalibrateRoll = cmd_imuCalibrateRoll;
668    }
669
670    /**
671     * Return the IMU Pitch calobratio value
672     *
673     * @return the IMU Pitch calobratio value
674     */
675    public double getCmd_imuCalibratePitch() {
676        return cmd_imuCalibratePitch;
677    }
678
679    /**
680     * Sets the IMU Pitch calobratio value
681     *
682     * @param cmd_imuCalibratePitch the IMU Pitch calobratio value
683     */
684    public void setCmd_imuCalibratePitch(double cmd_imuCalibratePitch) {
685        this.cmd_imuCalibratePitch = cmd_imuCalibratePitch;
686    }
687
688    /**
689     * Returns the command value for the blue LED
690     *
691     * @return the command value for the blue LED
692     */
693    public int getCmd_BlueLED() {
```

```
694         return cmd_BlueLED;
695     }
696 
697     /**
698      * Sets the command value for the blue LED
699      *
700      * @param cmd_BlueLED the command value for the blue LED and notify the
701      * observers
702      */
703     public void setCmd_BlueLED(int cmd_BlueLED) {
704         this.cmd_BlueLED = cmd_BlueLED;
705         setChanged();
706         notifyObservers();
707     }
708 
709     /**
710      * Returns the ROV ready value
711      *
712      * @return the ROV ready value
713      */
714     public boolean getFb_ROVReady() {
715 //         setChanged();
716 //         notifyObservers();
717         return fb_ROVReady;
718     }
719 
720     /**
721      * Sets the ROV ready value
722      *
723      * @param fb_ROVReady the ROV ready value
724      */
725     public void setFb_ROVReady(boolean fb_ROVReady) {
726         this.fb_ROVReady = fb_ROVReady;
727     }
728 
729     /**
730      * Return the I2C error value
731      *
732      * @return the I2C error value
733      */
734     public boolean getERROR_I2C() {
735         return ERROR_I2C;
736     }
737 
738     /**
739      * Sets the I2C error value
740      *
741      * @param ERROR_I2C the I2C error value
742      */
743     public void setERROR_I2C(boolean ERROR_I2C) {
744         this.ERROR_I2C = ERROR_I2C;
745     }
746 
747     /**
748      * Returns the dataToSend
749      *
750      * @return the dataToSend
751      */
```

```
752     public String getDataToSend() {
753         return dataToSend;
754     }
755 
756     /**
757      * Sets the dataToSend
758      *
759      * @param dataToSend the dataToSend
760      */
761     public void setDataToSend(String dataToSend) {
762         this.dataToSend = dataToSend;
763     }
764 
765     /**
766      * Returns the status of gatheringDataToSend
767      *
768      * @return the status of gatheringDataToSend
769      */
770     public boolean isGatheringDataToSend() {
771         return gatheringDataToSend;
772     }
773 
774     /**
775      * Sets the status of gatheringDataToSend
776      *
777      * @param gatheringDataToSend the status of gatheringDataToSend and notify
778      * the observers
779      */
780     public void setGatheringDataToSend(boolean gatheringDataToSend) {
781         this.gatheringDataToSend = gatheringDataToSend;
782         setChanged();
783         notifyObservers();
784     }
785 
786     /**
787      * Returns the command value for both the actuators
788      *
789      * @return the command value for both the actuators
790      */
791     public int getCmd_bothActuators() {
792         return cmd_bothActuators;
793     }
794 
795     /**
796      * Sets the command value for both the actuators
797      *
798      * @param cmd_bothActuators the command value for both the actuators and
799      * notify the observers
800      */
801     public void setCmd_bothActuators(int cmd_bothActuators) {
802         this.cmd_bothActuators = cmd_bothActuators;
803         setChanged();
804         notifyObservers();
805     }
806 
807     /**
808      * Returns the actuator difference
809      *
```

```
810     * @return the actuator difference
811     */
812    public int getActuatorDifference() {
813        return actuatorDifference;
814    }
815
816    /**
817     * Sets the actuator difference
818     *
819     * @param actuatorDifference the actuator difference
820     */
821    public void setActuatorDifference(int actuatorDifference) {
822        this.actuatorDifference = actuatorDifference;
823    }
824
825    /**
826     * Returns the target distance command
827     *
828     * @return the target distance command
829     */
830    public double getCmd_targetDistance() {
831        return cmd_targetDistance;
832    }
833
834    /**
835     * Sets the target distance command
836     *
837     * @param cmd_targetDistance the target distance command
838     */
839    public void setCmd_targetDistance(double cmd_targetDistance) {
840        this.cmd_targetDistance = cmd_targetDistance;
841    }
842
843    /**
844     * Returns the disable motor controllers command
845     *
846     * @return the disable motor controllers command
847     */
848    public boolean getCmd_disableMotors() {
849        return cmd_disableMotors;
850    }
851
852    /**
853     * Sets the disable motor controllers command
854     *
855     * @param cmd_disableMotors the disable motor controllers command
856     */
857    public void setCmd_disableMotors(boolean cmd_disableMotors) {
858        this.cmd_disableMotors = cmd_disableMotors;
859    }
860
861    /**
862     * Returns the offset depth beneath the ROV
863     *
864     * @return the offset depth beneath the ROV
865     */
866    public double getCmd_offsetDepthBeneathROV() {
867        return cmd_offsetDepthBeneathROV;
```

```
868     }
869
870     /**
871      * Sets the offset depth beneath the ROV
872      *
873      * @param cmd_offsetDepthBeneathROV the offset depth beneath the ROV
874      */
875     public void setCmd_offsetDepthBeneathROV(double cmd_offsetDepthBeneathROV) {
876         this.cmd_offsetDepthBeneathROV = cmd_offsetDepthBeneathROV;
877     }
878
879     /**
880      * Returns the offset depth of the ROV
881      *
882      * @return the offset depth of the ROV
883      */
884     public double getCmd_offsetROVdepth() {
885         return cmd_offsetROVdepth;
886     }
887
888     /**
889      * Sets the offset depth of the ROV
890      *
891      * @param cmd_offsetROVdepth the offset depth of the ROV
892      */
893     public void setCmd_offsetROVdepth(double cmd_offsetROVdepth) {
894         this.cmd_offsetROVdepth = cmd_offsetROVdepth;
895     }
896
897     /**
898      * Returns the ping value
899      *
900      * @return the ping value
901      */
902     public boolean isCmd_ping() {
903         return cmd_ping;
904     }
905
906     /**
907      * Sets the ping value
908      *
909      * @param cmd_ping the ping value
910      */
911     public void setCmd_ping(boolean cmd_ping) {
912         this.cmd_ping = cmd_ping;
913     }
914
915     /*'
916      * Returns  the client connected status
917      * @return the client connected status
918      */
919     public boolean isClientConnected() {
920         return clientConnected;
921     }
922
923     /**
924      * Sets the client connected status
925      *
```

```
926     * @param clientConnected the client connected status
927     */
928    public void setClientConnected(boolean clientConnected) {
929        this.clientConnected = clientConnected;
930    }
931 }
932
```

D:\Dokumenter\Skule\04 - NTNU\Bachelor\Github\TowedROV\src\I2CCom\I2CRW.java

```
1  /*
2   * This code is for the bachelor thesis named "Towed-ROV".
3   * The purpose is to build a ROV which will be towed behind a surface vessel
4   * and act as a multi-sensor platform, were it shall be easy to place new
5   * sensors. There will also be a video stream from the ROV.
6   *
7   * The system consists of two Raspberry Pis in the ROV that is connected to
8   * several Arduino micro controllers. These micro controllers are connected to
9   * feedback from the actuators, the echo sounder and extra optional sensors.
10   * The external computer which is on the surface vessel is connected to a GPS,
11   * echo sounder over USB, and the ROV over ethernet. It will present and
12   * log data in addition to handle user commands for controlling the ROV.
13   */
14 package I2CCom;
15
16 import ROV.Data;
17 import com.pi4j.io.i2c.I2CBus;
18 import com.pi4j.io.i2c.I2CDevice;
19 import com.pi4j.io.i2c.I2CFactory;
20
21 /**
22   * This class is responsible for sending and reciving data from and to I2C
23   * devices.
24   */
25 public class I2CRW implements Runnable {
26
27     protected static Data data;
28
29     I2CDevice arduinoIO;
30     I2CDevice actuatorSB;
31     I2CDevice actuatorPS;
32
33     //User settings
34     //SB and PS 180 target equals 0 degrees wing pos
35     private final static int PS_ACTUATOR_SPEED = 50;
36     private final static int SB_ACTUATOR_SPEED = 50;
37
38     //Polulu JRK drive commands
39     private final static int PS_ACTUATOR_ADDRESS = 0x10;
40     private final static int SB_ACTUATOR_ADDRESS = 0x0F;
41     private final static int ACTUATOR_STOP = 0xFF;
42
43     //Arduino Address
44     private final static int ARDUINO_IO_ADDRESS = 0x0B;
45
46     //JRK commands
47     int JRK_setTargetLowResRev = 0xE0;
48     int JRK_setTargetLowResFwd = 0xE1;
49     int JRK_getScaledFeedback = 0xA7; //The low byte of “Feedback”
50
51     String start_char = "<";
52     String end_char = ">";
53     String sep_char = ":";
54
55     /**
```

```
56   * Constructor of the i2CRW class Initiates the bus and adds slaves
57   *
58   * @param data the shared recource data class
59   */
60  public I2CRW(Data data) {
61      this.data = data;
62
63      try {
64          //System.out.println("Creatingbus");
65          I2CBus bus = I2CFactory.getInstance(I2CBus.BUS_3);
66          //System.out.println("Creatingdevices");
67          arduinoIO = bus.getDevice(ARDUINO_IO_ADDRESS);
68          actuatorSB = bus.getDevice(SB_ACTUATOR_ADDRESS);
69          actuatorPS = bus.getDevice(PS_ACTUATOR_ADDRESS);
70
71      } catch (Exception e) {
72          System.out.println("Failed to instansiate I2 Bus");
73      }
74
75  }
76
77  /**
78   * The run method does nothing in this class
79   */
80  @Override
81  public void run() {
82      while (true) {
83
84      }
85  }
86
87  /**
88   * This method is responsible for sending data to an I2C device
89   *
90   * @param device the device the data should be sent to
91   * @param commandValue the command value that should be sent
92   */
93  public void sendI2CData(String device, int commandValue) {
94      if (!data.getCmd_disableMotors()) {
95          try {
96              switch (device) {
97                  case "ActuatorPS_setTarget":
98
99                      if (commandValue > data.getFb_actuatorPSPos()
100                             && commandValue > 0
101                             && commandValue <= 254) {
102                         actuatorPS.write(JRK_setTargetLowResFwd, (byte) PS_ACTUATOR_SPEED);
103                     }
104
105                     if (commandValue < data.getFb_actuatorPSPos()
106                             && commandValue > 0
107                             && commandValue <= 254) {
108                         actuatorPS.write(JRK_setTargetLowResRev, (byte) PS_ACTUATOR_SPEED);
109                     }
110                     if (commandValue == 0) {
111                         actuatorPS.write(JRK_setTargetLowResRev, (byte) 0);
112                     }
113                     break;
```

```
114                 case "ActuatorSB_setTarget":
115
116                     if (commandValue > data.getFb_actuatorSBPos()
117                             && commandValue > 0
118                             && commandValue <= 254) {
119                         actuatorSB.write(JRK_setTargetLowResFwd, (byte) SB_ACTUATOR_SPEED);
120                     }
121
122                     if (commandValue < data.getFb_actuatorSBPos()
123                             && commandValue > 0
124                             && commandValue <= 254) {
125                         actuatorSB.write(JRK_setTargetLowResRev, (byte) SB_ACTUATOR_SPEED);
126                     }
127
128                     if (commandValue == 0) {
129                         actuatorSB.write(JRK_setTargetLowResRev, (byte) 0);
130                     }
131                     break;
132                 case "ActuatorSB_stopMotor":
133                     //actuatorSB.write((byte) ACTUATOR_STOP);
134                     try {
135                         Thread.sleep(10);
136                     } catch (Exception e) {
137                     }
138
139                     actuatorSB.write(JRK_setTargetLowResRev, (byte) 0);
140                     break;
141                 case "ActuatorPS_stopMotor":
142                     try {
143                         Thread.sleep(10);
144                     } catch (Exception e) {
145                     }
146                     //actuatorPS.write((byte) ACTUATOR_STOP);
147                     actuatorPS.write(JRK_setTargetLowResRev, (byte) 0);
148                     break;
149
150             }
151         } catch (Exception e) {
152             data.setERROR_I2C(true);
153             System.out.println("Error at I2C read write");
154             System.out.println(e);
155             //Error writing to i2c
156         }
157     }
158 }
159
160 /**
161  * Reads I2C data from an device
162  *
163  * Not used!
164  *
165  * @param device the device it should gather data from
166  */
167 public void readI2CData(String device) {
168     byte[] inputDataRaw = new byte[32];
169
170     String dataRecieved = "";
171     try {
```

```
172         switch (device) {
173
174             case "ActuatorSB_Feedback":
175
176                 // int test = actuatorSB.read(JRK_getScaledFeedback);
177                 byte[] byteArraySB = new byte[2];
178                 actuatorSB.read(0xA7, byteArraySB, 0, 2);
179                 int posSB = byteArraySB[0] + 256 * byteArraySB[1];
180
181 //                int posSB = actuatorSB.read(JRK_getScaledFeedback);
182                 data.setFb_actuatorSBPos(posSB);
183                 break;
184
185             case "ActuatorPS_Feedback":
186                 byte[] byteArrayPS = new byte[2];
187                 actuatorPS.read(0xA7, byteArrayPS, 0, 2);
188                 int posPS = byteArrayPS[0] + 256 * byteArrayPS[1];
189                 data.setFb_actuatorPSPos(posPS);
190                 break;
191
192             case "ArduinoIO":
193                 byte[] buffer = new byte[20];
194                 arduinoIO.read(buffer, 0, 20);
195                 String bufferData = new String(buffer);
196                 //arduinoIO.read(inputDataRaw, 0, 6);
197                 int sizeOfRecievedData = 0;
198
199                 for (byte b : inputDataRaw) {
200                     if (b != -1) {
201                         sizeOfRecievedData++;
202                     }
203                 }
204
205                 byte[] inputData = new byte[sizeOfRecievedData];
206                 System.arraycopy(inputDataRaw, 0, inputData, 0, sizeOfRecievedData);
207
208                 dataRecieved = new String(inputData);
209
210                 break;
211
212         }
213     } catch (Exception e) {
214         System.out.println(e);
215     }
216
217 }
218 }
219
```

D:\Dokumenter\Skule\04 - NTNU\Bachelor\Github\TowedROV\src\ROV\Logic.java

```
1 /*
2  * This code is for the bachelor thesis named "Towed-ROV".
3  * The purpose is to build a ROV which will be towed behind a surface vessel
4  * and act as a multi-sensor platform, were it shall be easy to place new
5  * sensors. There will also be a video stream from the ROV.
6  *
7  * The system consists of two Raspberry Pis in the ROV that is connected to
8  * several Arduino micro controllers. These micro controllers are connected to
9  * feedback from the actuators, the echo sounder and extra optional sensors.
10  * The external computer which is on the surface vessel is connected to a GPS,
11  * echo sounder over USB, and the ROV over ethernet. It will present and
12  * log data in addition to handle user commands for controlling the ROV.
13  */
14 package ROV;
15
16 import I2CCom.*;
17 import ROV.TCPCom.*;
18 import com.pi4j.io.gpio.GpioController;
19 import com.pi4j.io.gpio.GpioFactory;
20 import com.pi4j.io.gpio.GpioPinDigitalOutput;
21 import com.pi4j.io.gpio.PinState;
22 import com.pi4j.io.gpio.RaspiPin;
23 import java.util.HashMap;
24 import java.util.Map;
25
26 import java.util.Observable;
27 import java.util.Observer;
28
29 /**
30  * This class handels the actuator logic of the ROV and any other tasks that has
31  * a crucial update rate
32  */
33 public class Logic implements Runnable, Observer {
34
35     Data data = null;
36     I2CRW i2cRw = null;
37     int old_cmd_actuatorPS = 0;
38     int old_cmd_actuatorSB = 0;
39     int old_cmd_bothActuators = 0;
40
41     int old_cmd_BlueLED = 0;
42
43     int old_cmd_actuatorPS_2 = 0;
44     int old_cmd_actuatorSB_2 = 0;
45
46     int old_cmd_actuatorSB_man = 0;
47     int old_cmd_actuatorPS_man = 0;
48
49     double elapsedTimer = 0;
50     double elapsedTimerNano = 0;
51     long lastTime = 0;
52
53     double elapsedTimer_sendData = 0;
54     double elapsedTimerNano_sendData = 0;
55     long lastTime_sendData = 0;
56
57     private final static int actuatorPShysteresis = 8;
58     private final static int actuatorSBhysteresis = 8;
59     private HashMap<String, String> newDataToSend = new HashMap<>();
60
61     final GpioController gpio = GpioFactory.getInstance();
62
63     final GpioPinDigitalOutput BlueLED_PIN = gpio.provisionDigitalOutputPin(RaspiPin.GPIO_00, "BlueLED", PinState.LOW);
64
65     /**
66      *
67      * @param data the shared recource data class
```

```
68   * @param i2cRw the I2C class for handeling I2C commands
69   */
70  public Logic(Data data, I2CRW i2cRw) {
71      this.data = data;
72      this.i2cRw = i2cRw;
73
74  }
75
76  /**
77   * Crucial tasks is runned in this method:
78   *
79   * The run method checks if the current actuator position against the given
80   * command value. If they are equal an stop command wil bes sendt.
81   *
82   * It also checks if the communication to the GUI has been lost for more
83   * than five seconds
84   *
85   * Updates the gatherFbData list so it is ready to be sent to the GUI
86   */
87  @Override
88  public void run() {
89      try {
90          if (old_cmd_actuatorPS_2 != data.getCmd_actuatorPS()) {
91              System.out.println("Actuator PS FB : " + data.fb_actuatorPSPos);
92              if (data.fb_actuatorPSPos >= data.getCmd_actuatorPS() - actuatorPShysteresis
93                      && data.fb_actuatorPSPos <= data.getCmd_actuatorPS() + actuatorPShysteresis) {
94                  i2cRw.sendI2CData("ActuatorPS_stopMotor", (byte) 0);
95                  old_cmd_actuatorPS_2 = data.getCmd_actuatorPS();
96              }
97          }
98          if (old_cmd_actuatorSB_2 != data.getCmd_actuatorSB()) {
99              System.out.println("Actuator SB FB : : " + data.fb_actuatorSBPos);
100 //             System.out.println("Distance to target(SB): " + (data.getFb_actuatorSBPos() - data.getCmd_actuatorSB()));
101             if (data.fb_actuatorSBPos >= data.getCmd_actuatorSB() - actuatorSBhysteresis
102                     && data.fb_actuatorSBPos <= data.getCmd_actuatorSB() + actuatorSBhysteresis) {
103                 i2cRw.sendI2CData("ActuatorSB_stopMotor", (byte) 0);
104                 old_cmd_actuatorSB_2 = data.getCmd_actuatorSB();
105             }
106         }
107         if (data.isCmd_ping()) {
108             data.setClientConnected(true);
109             elapsedTimer = 0;
110             data.setCmd_ping(false);
111         }
112         elapsedTimerNano = (System.nanoTime() - lastTime);
113         elapsedTimer = elapsedTimerNano / 1000000;
114         if (elapsedTimer > 5000 && data.isClientConnected()) {
115             //System.out.println("Lost connection go to emergency");
116         }
117
118         elapsedTimerNano_sendData = (System.nanoTime() - lastTime_sendData);
119         elapsedTimer_sendData = elapsedTimerNano_sendData / 1000000;
120         if (elapsedTimerNano_sendData > 50) {
121             gatherFbData();
122             lastTime_sendData = 0;
123         }
124
125     } catch (Exception e) {
126     }
127
128 }
129
130 /**
131  * The update method is used for updating variables only when the GUI is
132  * sending data to the ROV
133  *
134  * @param o the observer
135  * @param arg the observer arguments
136  */
137 @Override
```

```
138   public void update(Observable o, Object arg) {
139       //Commands
140       if (data.getcmd_targetMode() != 2) {
141           if (old_cmd_actuatorPS != data.getCmd_actuatorPS()) {
142               old_cmd_actuatorPS = data.getCmd_actuatorPS();
143               i2cRw.sendI2CData("ActuatorPS_setTarget", data.cmd_actuatorPS);
144           }
145
146           if (old_cmd_actuatorSB != data.getCmd_actuatorSB()) {
147               old_cmd_actuatorSB = data.getCmd_actuatorSB();
148               i2cRw.sendI2CData("ActuatorSB_setTarget", data.cmd_actuatorSB);
149           }
150       } else {
151           if (old_cmd_actuatorSB_man != data.getCmd_actuatorSB()
152                   || old_cmd_actuatorPS_man != data.getCmd_actuatorPS()) {
153               try {
154                   if (old_cmd_actuatorSB_man != data.getCmd_actuatorSB()) {
155                       i2cRw.sendI2CData("ActuatorSB_setTarget", data.getCmd_actuatorSB());
156                   }
157
158                   Thread.sleep(50);
159                   if (old_cmd_actuatorPS_man != data.getCmd_actuatorPS()) {
160                       i2cRw.sendI2CData("ActuatorPS_setTarget", data.getCmd_actuatorPS());
161                   }
162                   old_cmd_actuatorSB_man = data.getCmd_actuatorSB();
163                   old_cmd_actuatorPS_man = data.getCmd_actuatorPS();
164               } catch (Exception e) {
165               }
166
167           }
168
169       }
170       if (old_cmd_BlueLED != data.getCmd_BlueLED()) {
171           BlueLED_PIN.toggle();
172           old_cmd_BlueLED = data.getCmd_BlueLED();
173       }
174   }
175
176   /**
177    * This method is responsible to update the newDataToSend list so the data
178    * is ready to be sendt to the GUI
179    */
180   public void gatherFbData() {
181
182       newDataToSend.put("Fb_actuatorPSPos", String.valueOf(data.getFb_actuatorPSPos()));
183       newDataToSend.put("Fb_actuatorSBPos", String.valueOf(data.getFb_actuatorSBPos()));
184       newDataToSend.put("Fb_rollAngle", String.valueOf(data.getFb_rollAngle()));
185       newDataToSend.put("Fb_pitchAngle", String.valueOf(data.getFb_pitchAngle()));
186       newDataToSend.put("Fb_depthBeneathROV", String.valueOf(data.getCmd_currentROVdepth()));
187       newDataToSend.put("Fb_tempElBoxFront", String.valueOf(data.getFb_tempMainElBoxFront()));
188       newDataToSend.put("Fb_tempElBoxRear", String.valueOf(data.getFb_tempMainElBoxRear()));
189       newDataToSend.put("Fb_ROVReady", String.valueOf(data.getFb_ROVReady()));
190       newDataToSend.put("ERROR_I2C", String.valueOf(data.ERROR_I2C));
191
192       String dataToSend = "<";
193       for (Map.Entry e : newDataToSend.entrySet()) {
194           String key = (String) e.getKey();
195           String value = (String) e.getValue();
196           dataToSend = dataToSend + key + ":" + value + ":";
197       }
198       dataToSend = dataToSend.substring(0, dataToSend.length() - 1);
199       dataToSend = dataToSend + ">";
200
201       data.setDataToSend(dataToSend);
202
203   }
204 }
205
```

D:\Dokumenter\Skule\04 - NTNU\Bachelor\Github\TowedROV\src\com\stormbots\Main.java

```
1  /*
2   * This code is for the bachelor thesis named "Towed-ROV".
3   * The purpose is to build a ROV which will be towed behind a surface vessel
4   * and act as a multi-sensor platform, were it shall be easy to place new
5   * sensors. There will also be a video stream from the ROV.
6   *
7   * The system consists of two Raspberry Pis in the ROV that is connected to
8   * several Arduino micro controllers. These micro controllers are connected to
9   * feedback from the actuators, the echo sounder and extra optional sensors.
10  * The external computer which is on the surface vessel is connected to a GPS,
11  * echo sounder over USB, and the ROV over ethernet. It will present and
12  * log data in addition to handle user commands for controlling the ROV.
13  */
14 package com.stormbots;
15
16 public class Main {
17
18     /**
19      * @param args Any arguments passed from stdin
20      */
21     public static void main(String[] args) {
22         MiniPID miniPID;
23
24         miniPID = new MiniPID(0.25, 0.01, 0.4);
25         miniPID.setOutputLimits(10);
26         //miniPID.setMaxIOutput(2);
27         //miniPID.setOutputRampRate(3);
28         //miniPID.setOutputFilter(.3);
29         miniPID.setSetpointRange(40);
30
31         double target = 100;
32
33         double actual = 0;
34         double output = 0;
35
36         miniPID.setSetpoint(0);
37         miniPID.setSetpoint(target);
38
39         System.err.printf("Target\tActual\tOutput\tError\n");
40         //System.err.printf("Output\tP\tI\tD\n");
41
42         // Position based test code
43         for (int i = 0; i < 100; i++) {
44
45             //if(i==50)miniPID.setI(.05);
46             if (i == 60) {
47                 target = 50;
48             }
49
50             //if(i==75)target=(100);
51             //if(i>50 && i%4==0)target=target+(Math.random()-.5)*50;
52             output = miniPID.getOutput(actual, target);
53             actual = actual + output;
54
55             //System.out.println("==========================");
```

```
56          //System.out.printf("Current: %3.2f , Actual: %3.2f, Error: %3.2f\n",actual, output, (target-actual));
57          System.err.printf("%3.2f\t%3.2f\t%3.2f\t%3.2f\n", target, actual, output, (target - actual));
58
59          //if(i>80 && i%5==0)actual+=(Math.random()-.5)*20;
60      }
61  }
62 }
63
```

D:\Dokumenter\Skule\04 - NTNU\Bachelor\Github\TowedROV\src\com\stormbots\MiniPID.java

```
1 /*
2  * This code is for the bachelor thesis named "Towed-ROV".
3  * The purpose is to build a ROV which will be towed behind a surface vessel
4  * and act as a multi-sensor platform, were it shall be easy to place new
5  * sensors. There will also be a video stream from the ROV.
6  *
7  * The system consists of two Raspberry Pis in the ROV that is connected to
8  * several Arduino micro controllers. These micro controllers are connected to
9  * feedback from the actuators, the echo sounder and extra optional sensors.
10  * The external computer which is on the surface vessel is connected to a GPS,
11  * echo sounder over USB, and the ROV over ethernet. It will present and
12  * log data in addition to handle user commands for controlling the ROV.
13  */
14 package com.stormbots;
15
16 /**
17  * Small, easy to use PID implementation with advanced controller capability.
18  *
19  * Minimal usage:
20  *
21  * MiniPID pid = new MiniPID(p,i,d);
22  *
23  * ...looping code...{
24  *
25  * output= pid.getOutput(sensorvalue,target);
26  *
27  *
28  * }
29  *
30  * see
31  * http://brettbeauregard.com/blog/2011/04/improving-the-beginners-pid-direction/improving-the-beginners-pid-introduction
32  */
33 public class MiniPID {
34     //**********************************
35     // Class private variables
36     //**********************************
37
38     private double P = 0;
39     private double I = 0;
40     private double D = 0;
41     private double F = 0;
42
43     private double maxIOutput = 0;
44     private double maxError = 0;
45     private double errorSum = 0;
46
47     private double maxOutput = 0;
48     private double minOutput = 0;
49
50     private double setpoint = 0;
51
52     private double lastActual = 0;
53
54     private boolean firstRun = true;
55     private boolean reversed = false;
56
57     private double outputRampRate = 0;
58     private double lastOutput = 0;
59
60     private double outputFilter = 0;
61
62     private double setpointRange = 0;
63
64     //**********************************
65     // Constructor functions
66     //**********************************
67     /**
68      * Create a MiniPID class object. See setP, setI, setD methods for more
```

```
69   * detailed parameters.
70   *
71   * @param p Proportional gain. Large if large difference between setpoint
72   * and target.
73   * @param i Integral gain. Becomes large if setpoint cannot reach target
74   * quickly.
75   * @param d Derivative gain. Responds quickly to large changes in error.
76   * Small values prevents P and I terms from causing overshoot.
77   */
78  public MiniPID(double p, double i, double d) {
79      P = p;
80      I = i;
81      D = d;
82      checkSigns();
83  }
84
85  /**
86   * Create a MiniPID class object. See setP, setI, setD, setF methods for
87   * more detailed parameters.
88   *
89   * @param p Proportional gain. Large if large difference between setpoint
90   * and target.
91   * @param i Integral gain. Becomes large if setpoint cannot reach target
92   * quickly.
93   * @param d Derivative gain. Responds quickly to large changes in error.
94   * Small values prevents P and I terms from causing overshoot.
95   * @param f Feed-forward gain. Open loop "best guess" for the output should
96   * be. Only useful if setpoint represents a rate.
97   */
98  public MiniPID(double p, double i, double d, double f) {
99      P = p;
100     I = i;
101     D = d;
102     F = f;
103     checkSigns();
104  }
105
106  //********************************
107  // Configuration functions
108  //********************************
109  /**
110   * Configure the Proportional gain parameter.
111   *
112   * This responds quickly to changes in setpoint, and provides most of the
113   * initial driving force to make corrections.
114   *
115   * Some systems can be used with only a P gain, and many can be operated
116   * with only PI.
117   *
118   * For position based controllers, this is the first parameter to tune, with
119   * I second.
120   *
121   * For rate controlled systems, this is often the second after F.
122   *
123   * @param p Proportional gain. Affects output according to
124   * output+=P*(setpoint-current_value)
125   */
126  public void setP(double p) {
127      P = p;
128      checkSigns();
129  }
130
131  /**
132   * Changes the I parameter
133   *
134   * This is used for overcoming disturbances, and ensuring that the
135   * controller always gets to the control mode. Typically tuned second for
136   * "Position" based modes, and third for "Rate" or continuous based modes.
137   *
138   * Affects output through output+=previous_errors*Igain
139   * ;previous_errors+=current_error
```

```
140     *
141     *  * see {@link #setMaxIOutput(double) setMaxIOutput} for how to restrict
142     *
143     * @param i New gain value for the Integral term
144     */
145    public void setI(double i) {
146        if (I != 0) {
147            errorSum = errorSum * I / i;
148        }
149        if (maxIOutput != 0) {
150            maxError = maxIOutput / i;
151        }
152        I = i;
153        checkSigns();
154        // Implementation note:
155        // This Scales the accumulated error to avoid output errors.
156        // As an example doubling the I term cuts the accumulated error in half, which results in the
157        // output change due to the I term constant during the transition.
158    }
159
160    /**
161     * Changes the D parameter This has two primary effects:
162     *
163     * Adds a "startup kick" and speeds up system response during setpoint
164     * changes
165     *
166     * Adds "drag" and slows the system when moving toward the target
167     *
168     * A small D value can be useful for both improving response times, and
169     * preventing overshoot. However, in many systems a large D value will cause
170     * significant instability, particularly for large setpoint changes.
171     *
172     * Affects output through output += -D*(current_input_value -
173     * last_input_value)
174     *
175     * @param d New gain value for the Derivative term
176     */
177    public void setD(double d) {
178        D = d;
179        checkSigns();
180    }
181
182    /**
183     * Configure the FeedForward parameter.
184     *
185     * This is excellent for velocity, rate, and other continuous control modes
186     * where you can expect a rough output value based solely on the setpoint.
187     *
188     * Should not be used in "position" based control modes.
189     *
190     * Affects output according to output+=F*Setpoint. Note, that a F-only
191     * system is actually open loop.
192     *
193     * @param f Feed forward gain.
194     */
195    public void setF(double f) {
196        F = f;
197        checkSigns();
198    }
199
200    /**
201     * Configure the PID object. See setP, setI, setD methods for more detailed
202     * parameters.
203     *
204     * @param p Proportional gain. Large if large difference between setpoint
205     * and target.
206     * @param i Integral gain. Becomes large if setpoint cannot reach target
207     * quickly.
208     * @param d Derivative gain. Responds quickly to large changes in error.
209     * Small values prevents P and I terms from causing overshoot.
210     */
```

```
211   public void setPID(double p, double i, double d) {
212       P = p;
213       D = d;
214       //Note: the I term has additional calculations, so we need to use it's
215       //specific method for setting it.
216       setI(i);
217       checkSigns();
218   }
219
220   /**
221    * Configure the PID object. See setP, setI, setD, setF methods for more
222    * detailed parameters.
223    *
224    * @param p Proportional gain. Large if large difference between setpoint
225    * and target.
226    * @param i Integral gain. Becomes large if setpoint cannot reach target
227    * quickly.
228    * @param d Derivative gain. Responds quickly to large changes in error.
229    * Small values prevents P and I terms from causing overshoot.
230    * @param f Feed-forward gain. Open loop "best guess" for the output should
231    * be. Only useful if setpoint represents a rate.
232    */
233   public void setPID(double p, double i, double d, double f) {
234       P = p;
235       D = d;
236       F = f;
237       //Note: the I term has additional calculations, so we need to use it's
238       //specific method for setting it.
239       setI(i);
240       checkSigns();
241   }
242
243   /**
244    * Set the maximum output value contributed by the I component of the system
245    * This can be used to prevent large windup issues and make tuning simpler
246    *
247    * @param maximum. Units are the same as the expected output value
248    */
249   public void setMaxIOutput(double maximum) {
250       // Internally maxError and Izone are similar, but scaled for different purposes.
251       // The maxError is generated for simplifying math, since calculations against
252       // the max error are far more common than changing the I term or Izone.
253       maxIOutput = maximum;
254       if (I != 0) {
255           maxError = maxIOutput / I;
256       }
257   }
258
259   /**
260    * Specify a maximum output range.
261    *
262    * When one input is specified, output range is configured to [-output,
263    * output]
264    *
265    * @param output the output
266    */
267   public void setOutputLimits(double output) {
268       setOutputLimits(-output, output);
269   }
270
271   /**
272    * Specify a maximum output. When two inputs specified, output range is
273    * configured to [minimum, maximum]
274    *
275    * @param minimum possible output value
276    * @param maximum possible output value
277    */
278   public void setOutputLimits(double minimum, double maximum) {
279       if (maximum < minimum) {
280           return;
281       }
```

```
282         maxOutput = maximum;
283         minOutput = minimum;
284 
285         // Ensure the bounds of the I term are within the bounds of the allowable output swing
286         if (maxIOutput == 0 || maxIOutput > (maximum - minimum)) {
287             setMaxIOutput(maximum - minimum);
288         }
289     }
290 
291     /**
292      * Set the operating direction of the PID controller
293      *
294      * @param reversed Set true to reverse PID output
295      */
296     public void setDirection(boolean reversed) {
297         this.reversed = reversed;
298     }
299 
300     //************************************
301     // Primary operating functions
302     //************************************
303     /**
304      * Configure setpoint for the PID calculations
305      *
306      * This represents the target for the PID system's, such as a position,
307      * velocity, or angle.
308      *
309      * see MiniPID#getOutput(actual)
310      *
311      * @param setpoint the setpotn
312      */
313     public void setSetpoint(double setpoint) {
314         this.setpoint = setpoint;
315     }
316 
317     /**
318      * Calculate the output value for the current PID cycle.
319      *
320      *
321      * @param actual The monitored value, typically as a sensor input.
322      * @param setpoint The target value for the system
323      * @return calculated output value for driving the system
324      */
325     public double getOutput(double actual, double setpoint) {
326         double output;
327         double Poutput;
328         double Ioutput;
329         double Doutput;
330         double Foutput;
331 
332         this.setpoint = setpoint;
333 
334         // Ramp the setpoint used for calculations if user has opted to do so
335         if (setpointRange != 0) {
336             setpoint = constrain(setpoint, actual - setpointRange, actual + setpointRange);
337         }
338 
339         // Do the simple parts of the calculations
340         double error = setpoint - actual;
341 
342         // Calculate F output. Notice, this depends only on the setpoint, and not the error.
343         Foutput = F * setpoint;
344 
345         // Calculate P term
346         Poutput = P * error;
347 
348         // If this is our first time running this, we don't actually _have_ a previous input or output.
349         // For sensor, sanely assume it was exactly where it is now.
350         // For last output, we can assume it's the current time-independent outputs.
351         if (firstRun) {
352             lastActual = actual;
```

```
353         lastOutput = Poutput + Foutput;
354         firstRun = false;
355       }
356
357       // Calculate D Term
358       // Note, this is negative. This actually "slows" the system if it's doing
359       // the correct thing, and small values helps prevent output spikes and overshoot
360       Doutput = -D * (actual - lastActual);
361       lastActual = actual;
362
363       // The Iterm is more complex. There's several things to factor in to make it easier to deal with.
364       // 1. maxIoutput restricts the amount of output contributed by the Iterm.
365       // 2. prevent windup by not increasing errorSum if we're already running against our max Ioutput
366       // 3. prevent windup by not increasing errorSum if output is output=maxOutput
367       Ioutput = I * errorSum;
368       if (maxIOutput != 0) {
369         Ioutput = constrain(Ioutput, -maxIOutput, maxIOutput);
370       }
371
372       // And, finally, we can just add the terms up
373       output = Foutput + Poutput + Ioutput + Doutput;
374
375       // Figure out what we're doing with the error.
376       if (minOutput != maxOutput && !bounded(output, minOutput, maxOutput)) {
377         errorSum = error;
378         // reset the error sum to a sane level
379         // Setting to current error ensures a smooth transition when the P term
380         // decreases enough for the I term to start acting upon the controller
381         // From that point the I term will build up as would be expected
382       } else if (outputRampRate != 0 && !bounded(output, lastOutput - outputRampRate, lastOutput + outputRampRate)) {
383         errorSum = error;
384       } else if (maxIOutput != 0) {
385         errorSum = constrain(errorSum + error, -maxError, maxError);
386         // In addition to output limiting directly, we also want to prevent I term
387         // buildup, so restrict the error directly
388       } else {
389         errorSum += error;
390       }
391
392       // Restrict output to our specified output and ramp limits
393       if (outputRampRate != 0) {
394         output = constrain(output, lastOutput - outputRampRate, lastOutput + outputRampRate);
395       }
396       if (minOutput != maxOutput) {
397         output = constrain(output, minOutput, maxOutput);
398       }
399       if (outputFilter != 0) {
400         output = lastOutput * outputFilter + output * (1 - outputFilter);
401       }
402
403       // Get a test printline with lots of details about the internal
404       // calculations. This can be useful for debugging.
405       // System.out.printf("Final output %5.2f [ %5.2f, %5.2f , %5.2f  ], eSum %.2f\n",output,Poutput, Ioutput, Doutput,errorSum );
406       // System.out.printf("%5.2f\t%5.2f\t%5.2f\t%5.2f\n",output,Poutput, Ioutput, Doutput );
407       lastOutput = output;
408       return output;
409     }
410
411     /**
412      * Calculate the output value for the current PID cycle.
413      *
414      * In no-parameter mode, this uses the last sensor value, and last setpoint
415      * value.
416      *
417      * Not typically useful, and use of parameter modes is suggested.
418      *
419      *
420      * @return calculated output value for driving the system
421      */
422     public double getOutput() {
423       return getOutput(lastActual, setpoint);
```

```
424   }
425
426   /**
427    * Calculate the output value for the current PID cycle.
428    *
429    * In one parameter mode, the last configured setpoint will be used.
430    *
431    *
432    * see MiniPID#setSetpoint()
433    *
434    * @param actual The monitored value, typically as a sensor input. param
435    * setpoint The target value for the system
436    * @return calculated output value for driving the system
437    */
438   public double getOutput(double actual) {
439      return getOutput(actual, setpoint);
440   }
441
442   /**
443    * Resets the controller. This erases the I term buildup, and removes D gain
444    * on the next loop.
445    *
446    * This should be used any time the PID is disabled or inactive for extended
447    * duration, and the controlled portion of the system may have changed due
448    * to external forces.
449    */
450   public void reset() {
451      firstRun = true;
452      errorSum = 0;
453   }
454
455   /**
456    * Set the maximum rate the output can increase per cycle.
457    *
458    * This can prevent sharp jumps in output when changing setpoints or
459    * enabling a PID system, which might cause stress on physical or electrical
460    * systems.
461    *
462    * Can be very useful for fast-reacting control loops, such as ones with
463    * large P or D values and feed-forward systems.
464    *
465    * @param rate, with units being the same as the output
466    */
467   public void setOutputRampRate(double rate) {
468      outputRampRate = rate;
469   }
470
471   /**
472    * Set a limit on how far the setpoint can be from the current position
473    *
474    *
475    * Can simplify tuning by helping tuning over a small range applies to a
476    * much larger range.
477    *
478    *
479    * This limits the reactivity of P term, and restricts impact of large D
480    * term during large setpoint adjustments. Increases lag and I term if range
481    * is too small.
482    *
483    * @param range, with units being the same as the expected sensor range.
484    */
485   public void setSetpointRange(double range) {
486      setpointRange = range;
487   }
488
489   /**
490    * Set a filter on the output to reduce sharp oscillations.
491    *
492    * 0.1 is likely a sane starting value. Larger values use historical data
493    * more heavily, with low values weigh newer data. 0 will disable,
494    * filtering, and use only the most recent value.
```

```
495     *
496     * Increasing the filter strength will P and D oscillations, but force
497     * larger I values and increase I term overshoot.
498     *
499     * Uses an exponential wieghted rolling sum filter, according to a simple
500     *
501     *
502     * output*(1-strength)*sum(0..n){output*strength^n} algorithm.
503     *
504     * param output valid between [0..1), meaning [current output only..
505     * historical output only)
506     */
507    public void setOutputFilter(double strength) {
508        if (strength == 0 || bounded(strength, 0, 1)) {
509            outputFilter = strength;
510        }
511    }
512
513    //************************************
514    // Helper functions
515    //************************************
516    /**
517     * Forces a value into a specific range
518     *
519     * @param value input value
520     * @param min maximum returned value
521     * @param max minimum value in range
522     * @return Value if it's within provided range, min or max otherwise
523     */
524    private double constrain(double value, double min, double max) {
525        if (value > max) {
526            return max;
527        }
528        if (value < min) {
529            return min;
530        }
531        return value;
532    }
533
534    /**
535     * Test if the value is within the min and max, inclusive
536     *
537     * @param value to test
538     * @param min Minimum value of range
539     * @param max Maximum value of range
540     * @return true if value is within range, false otherwise
541     */
542    private boolean bounded(double value, double min, double max) {
543        // Note, this is an inclusive range. This is so tests like
544        // `bounded(constrain(0,0,1),0,1)` will return false.
545        // This is more helpful for determining edge-case behaviour
546        // than <= is.
547        return (min < value) && (value < max);
548    }
549
550    /**
551     * To operate correctly, all PID parameters require the same sign This
552     * should align with the {@literal}reversed value
553     */
554    private void checkSigns() {
555        if (reversed) {  // all values should be below zero
556            if (P > 0) {
557                P *= -1;
558            }
559            if (I > 0) {
560                I *= -1;
561            }
562            if (D > 0) {
563                D *= -1;
564            }
565            if (F > 0) {
```

```
566             F *= -1;
567         }
568     } else {  // all values should be above zero
569         if (P < 0) {
570             P *= -1;
571         }
572         if (I < 0) {
573             I *= -1;
574         }
575         if (D < 0) {
576             D *= -1;
577         }
578         if (F < 0) {
579             F *= -1;
580         }
581     }
582 }
583 }
584
```

D:\Dokumenter\Skule\04 - NTNU\Bachelor\Github\TowedROV\src\ROV\PID.java

```
1 /*
2  * This code is for the bachelor thesis named "Towed-ROV".
3  * The purpose is to build a ROV which will be towed behind a surface vessel
4  * and act as a multi-sensor platform, were it shall be easy to place new
5  * sensors. There will also be a video stream from the ROV.
6  *
7  * The system consists of two Raspberry Pis in the ROV that is connected to
8  * several Arduino micro controllers. These micro controllers are connected to
9  * feedback from the actuators, the echo sounder and extra optional sensors.
10  * The external computer which is on the surface vessel is connected to a GPS,
11  * echo sounder over USB, and the ROV over ethernet. It will present and
12  * log data in addition to handle user commands for controlling the ROV.
13  */
14 package ROV;
15
16 import java.util.Observable;
17 import java.util.Observer;
18 import com.stormbots.MiniPID;
19 import java.util.concurrent.atomic.*;
20
21 /**
22  * This class is responisble for the PID controller and its output to the
23  * actuators
24  *
25  */
26 public class PID implements Runnable, Observer {
27
28     Data data = null;
29     MiniPID miniPID;
30
31     AtomicInteger atomicTarget = new AtomicInteger(0);
32     AtomicInteger atomicActual = new AtomicInteger(0);
33
34     double target = 0;
35     double actual = 0;
36     Double output = new Double(0);
37
38     /**
39      * The constructor of the PID class. Creates a miniPID instance.
40      *
41      * @param data the shared recource data class
42      */
43     public PID(Data data) {
44         this.data = data;
45         miniPID = new MiniPID(data.getCmd_pid_p(), data.getCmd_pid_i(), data.getCmd_pid_d());
46         miniPID.setOutputLimits(0, 254);
47
48     }
49
50     /**
51      * Run methods gathers the input data and calculates an output value
52      */
53     @Override
54     public void run() {
55
```

```
56          if (data.getcmd_targetMode() != 2) {
57              if (data.getcmd_targetMode() == 0) {
58                  //Goal: Get to desired depth
59                  actual = data.getCmd_currentROVdepth();
60              }
61              if (data.getcmd_targetMode() == 1) {
62                  //Goal: Get to desired elevation above seafloor
63                  actual = data.getFb_depthBeneathROV();
64              }
65
66              target = data.getCmd_targetDistance();
67
68              miniPID.setSetpoint(target);
69              output = miniPID.getOutput(actual, target);
70
71              data.setCmd_bothActuators(output.intValue());
72          }
73      }
74
75      /**
76       *
77       * @param o the observer
78       * @param arg the arguments for the observer
79       */
80      @Override
81      public void update(Observable o, Object arg) {
82  //          target = data.getCmd_setDepth();
83  //          actual = data.getFb_depthFromPressure();
84
85  //          atomicTarget.set(data.getCmd_setDepth());
86  //          atomicTarget.set(data.getCmd_bothActuators());
87  //
88  //          atomicActual.set((data.getFb_actuatorPSPos() + data.getFb_actuatorPSPos()) / 2);
89  //          data.setActuatorDifference(data.getFb_actuatorPSPos() - data.getFb_actuatorSBPos());
90      }
91
92  }
93
```

D:\Dokumenter\Skule\04 - NTNU\Bachelor\Github\TowedROV\src\SerialCom\ReadSerialData.java

```
1 /*
2  * This code is for the bachelor thesis named "Towed-ROV".
3  * The purpose is to build a ROV which will be towed behind a surface vessel
4  * and act as a multi-sensor platform, were it shall be easy to place new
5  * sensors. There will also be a video stream from the ROV.
6  *
7  * The system consists of two Raspberry Pis in the ROV that is connected to
8  * several Arduino micro controllers. These micro controllers are connected to
9  * feedback from the actuators, the echo sounder and extra optional sensors.
10  * The external computer which is on the surface vessel is connected to a GPS,
11  * echo sounder over USB, and the ROV over ethernet. It will present and
12  * log data in addition to handle user commands for controlling the ROV.
13  */
14 package SerialCom;
15
16 import java.util.HashMap;
17 import java.util.Map;
18 import java.util.concurrent.ConcurrentHashMap;
19 import jssc.SerialPort;
20 import jssc.SerialPortList;
21 import jssc.SerialPortException;
22 import ROV.Data;
23
24 /**
25  * This class is eespnsible for reading serial data from the IMU, echo sounder
26  * and the Arduino I/O
27  *
28  *
29  */
30 public class ReadSerialData implements Runnable {
31
32     // Filter values
33     int actuatorFbFilter = 1; // 1 equals off
34
35     int actuatorPSFbFilterStorage = 0;
36     int actuatorSBFbFilterStorage = 0;
37
38     int actuatorPSFbFilterCounter = 0;
39     int actuatorSBFbFilterCounter = 0;
40
41     boolean portIsOpen = false;
42     String comPort = "";
43     String myName = "";
44     int baudRate = 0;
45     Data data = null;
46
47     /**
48      * The complete list of alla incomming data and its values
49      */
50     public HashMap<String, String> incommingData = new HashMap<>();
51
52     private static volatile double depth;
53     private static volatile double tempC;
54
55     /**
```

```
56   *
57   * @param data the shared recource data class
58   * @param comPort the com port it should use
59   * @param baudRate the baud rate of the com port
60   * @param myName the name of the com device it should connect to
61   */
62  public ReadSerialData(Data data, String comPort, int baudRate, String myName) {
63      this.comPort = comPort;
64      this.myName = myName;
65      this.baudRate = baudRate;
66      this.data = data;
67  }
68
69  /**
70   * Run command loops through the readData
71   */
72  @Override
73  public void run() {
74      while (true) {
75          try {
76
77              readData(comPort, baudRate);
78          } catch (Exception e) {
79          }
80
81      }
82  }
83
84  /**
85   * readData is responsible for gathering data from the serial devices
86   *
87   * @param comPort the com port it should connect to
88   * @param baudRate the boud rate of the com port
89   */
90  public void readData(String comPort, int baudRate) {
91
92      boolean recievedData = false;
93      //Declare special symbol used in serial data stream from Arduino
94      String startChar = "<";
95      String endChar = ">";
96      String seperationChar = ":";
97
98      SerialPort serialPort = new SerialPort(comPort);
99
100     if (!portIsOpen) {
101         try {
102             serialPort.openPort();
103             portIsOpen = true;
104         } catch (SerialPortException ex) {
105             System.out.println(ex);
106         }
107     }
108
109     while (recievedData == false) {
110         try {
111             Thread.sleep(50);
112         } catch (Exception ex) {
113
```

```
114         }
115         String buffer;
116 
117         try {
118            serialPort.setParams(baudRate, 8, 1, 0);
119            buffer = serialPort.readString();
120 
121            // System.out.println(buffer);
122            boolean dataNotNull = false;
123            boolean dataHasFormat = false;
124 
125            if ((buffer != null)) {
126               dataHasFormat = true;
127            } else {
128               dataHasFormat = false;
129               dataNotNull = false;
130 
131            }
132 
133            if (dataHasFormat) {
134               String dataStream = buffer;
135 
136               dataStream = dataStream.substring(dataStream.indexOf(startChar) + 1);
137               dataStream = dataStream.substring(0, dataStream.indexOf(endChar));
138               dataStream = dataStream.replace("?", "");
139               String[] data = dataStream.split(seperationChar);
140 
141               for (int i = 0; i < data.length; i = i + 2) {
142                  //this.data.data.put(data[i], data[i + 1]);
143                  incommingData.put(data[i], data[i + 1]);
144 
145               }
146 
147               sendIncommingDataToDataHandler();
148            }
149 
150 //          if (elapsedTimer != 0)
151 //          {
152 //
153 //             System.out.println("Data is recieved in: " + elapsedTimer + " millis"
154 //                   + " or with: " + 1000 / elapsedTimer + " Hz");
155 //          } else
156 //          {
157 //             System.out.println("Data is recieved in: " + elapsedTimer + " millis"
158 //                   + " or with: unlimited Hz!");
159 //          }
160         } catch (Exception ex) {
161            // System.out.println("Lost connection to " + myName);
162         }
163 
164      }
165   }
166 
167   private void sendIncommingDataToDataHandler() {
168      for (Map.Entry e : incommingData.entrySet()) {
169         String key = (String) e.getKey();
170         String value = (String) e.getValue();
171 
```

```
172         switch (key) {
173
174             case "D":
175                 double doubleValue = Double.parseDouble(value) * -1;
176                 data.setFb_depthBeneathROV(doubleValue);
177                 break;
178 //              case "DBT":
179 //                  data.setFb_depthBelowTransduser(Double.parseDouble(value));
180 //                  break;
181             case "ch1":
182                 data.setAnalogInputChannel_1(Double.parseDouble(value));
183                 break;
184             case "ch2":
185                 data.setAnalogInputChannel_2(Double.parseDouble(value));
186                 break;
187
188             case "ch3":
189                 if (value.equals("1.00")) {
190                     data.setDigitalInputChannel_3(true);
191                 } else {
192                     data.setDigitalInputChannel_3(false);
193                 }
194                 break;
195             case "ch4":
196                 if (value.equals("1.00")) {
197                     data.setDigitalInputChannel_4(true);
198                 } else {
199                     data.setDigitalInputChannel_4(false);
200                 }
201                 break;
202
203             case "PsActuatorFb":
204                 int tempValuePs = Integer.parseInt(value);
205                 if (tempValuePs < 0) {
206                     tempValuePs = 0;
207                 }
208                 if (tempValuePs > 254) {
209                     tempValuePs = 254;
210                 }
211
212                 if (actuatorPSFbFilterCounter <= actuatorFbFilter) {
213                     actuatorPSFbFilterStorage = actuatorPSFbFilterStorage + tempValuePs;
214                     actuatorPSFbFilterCounter++;
215
216                 } else {
217                     actuatorPSFbFilterStorage = actuatorPSFbFilterStorage / actuatorFbFilter;
218                     actuatorPSFbFilterCounter = 0;
219
220                     data.setFb_actuatorPSPos(tempValuePs);
221                 }
222                 break;
223
224             case "SbActuatorFb":
225                 int tempValueSb = Integer.parseInt(value);
226                 if (tempValueSb < 0) {
227                     tempValueSb = 0;
228                 }
229                 if (tempValueSb > 254) {
```

```
230                 tempValueSb = 254;
231             }
232
233             if (actuatorSBFbFilterCounter <= actuatorFbFilter) {
234                 actuatorSBFbFilterStorage = actuatorSBFbFilterStorage + tempValueSb;
235                 actuatorSBFbFilterCounter++;
236
237             } else {
238                 actuatorSBFbFilterStorage = actuatorSBFbFilterStorage / actuatorFbFilter;
239                 actuatorSBFbFilterCounter = 0;
240
241                 data.setFb_actuatorSBPos(tempValueSb);
242             }
243             break;
244
245         case "Roll":
246             data.setFb_rollAngle(Double.parseDouble(value));
247             //setRoll(Integer.parseInt(value));
248             break;
249         case "Pitch":
250             data.setFb_pitchAngle(Double.parseDouble(value));
251             //setPitch(Integer.parseInt(value));
252             break;
253         case "Heading":
254             data.setFb_heading(Integer.parseInt(value));
255             //setHeading(Integer.parseInt(value));
256             break;
257
258         case "tmp1":
259             data.setFb_tempMainElBoxFront(Double.parseDouble(value));
260             break;
261
262         case "tmp2":
263             data.setFb_tempMainElBoxRear(Double.parseDouble(value));
264             break;
265         }
266     }
267 }
268 }
269
```

D:\Dokumenter\Skule\04 - NTNU\Bachelor\Github\TowedROV\src\ROV\ROVMain.java

```
1 /*
2  * This code is for the bachelor thesis named "Towed-ROV".
3  * The purpose is to build a ROV which will be towed behind a surface vessel
4  * and act as a multi-sensor platform, were it shall be easy to place new
5  * sensors. There will also be a video stream from the ROV.
6  *
7  * The system consists of two Raspberry Pis in the ROV that is connected to
8  * several Arduino micro controllers. These micro controllers are connected to
9  * feedback from the actuators, the echo sounder and extra optional sensors.
10  * The external computer which is on the surface vessel is connected to a GPS,
11  * echo sounder over USB, and the ROV over ethernet. It will present and
12  * log data in addition to handle user commands for controlling the ROV.
13  */
14 package ROV;
15
16 import ROV.*;
17 import I2CCom.*;
18 import SerialCom.*;
19 import ROV.TCPCom.Server;
20 import java.util.Map;
21 import java.util.concurrent.TimeUnit;
22 import jssc.SerialPortList;
23 import java.util.concurrent.Executors;
24 import java.util.concurrent.ScheduledExecutorService;
25
26 /**
27  * The main class of the ROV. It is responsible for starting up the ROVs
28  * threads.
29  *
30  * @author Robin S. Thorholm
31  */
32 public class ROVMain {
33
34     private final static int serverPort = 8080;
35     private static Thread serialRW;
36     private static Thread I2CComHandler;
37
38     static boolean dataIsRecieved = false;
39     static boolean testIsDone = false;
40     private static Thread I2CRW;
41
42     /**
43      *
44      */
45     protected static Data dh;
46
47     private static Thread Server;
48     private static Thread alarmHandler;
49
50     private static Thread imuThread;
51     private static Thread ArduinoIOThread;
52     private static Thread ArduinoActuatorFBThread;
53
54     /**
55      * The main method of th ROV. Starts up the threads and instnaciates
56      * necessary tasks.
57      *
58      * @param args the command line arguments
59      */
60     public static void main(String[] args) {
```

```
61      boolean foundComPort = false;
62
63      ScheduledExecutorService executor
64              = Executors.newScheduledThreadPool(8);
65
66      String osName = System.getProperty("os.name");
67
68      String[] portNames = SerialPortList.getPortNames();
69      for (int i = 0; i < portNames.length; i++) {
70         System.out.println(portNames[i]);
71      }
72      if (!osName.contains("Windows")) {
73
74      } else {
75         System.out.println("OS is windows, does not start raspberry libraries");
76      }
77      dh = new Data();
78
79      I2CRW I2CRW_this = new I2CRW(dh);
80      Logic logic = new Logic(dh, I2CRW_this);
81
82      PID pid = new PID(dh);
83      dh.addObserver(logic);
84
85      SerialDataHandler sdh = new SerialDataHandler(dh);
86
87      executor.scheduleAtFixedRate(I2CRW_this,
88              20, 40, TimeUnit.MILLISECONDS);
89
90      executor.scheduleAtFixedRate(logic,
91              10, 5, TimeUnit.MILLISECONDS);
92
93      executor.scheduleAtFixedRate(pid,
94              20, 10, TimeUnit.MILLISECONDS);
95
96      Server = new Thread(new Server(serverPort, dh));
97      Server.start();
98      Server.setName("Server");
99      int inputData = 0;
100      if (!foundComPort) {
101         System.out.println("Searching for com ports...");
102         sdh.findComPorts();
103         foundComPort = true;
104      }
105
106      for (Map.Entry e : dh.comPortList.entrySet()) {
107         String comPortKey = (String) e.getKey();
108         String comPortValue = (String) e.getValue();
109         if (comPortValue.contains("IMU")) {
110            imuThread = new Thread(new ReadSerialData(dh, comPortKey, 115200, comPortValue));
111            imuThread.start();
112            imuThread.setName(comPortValue);
113            System.out.println("IMU found");
114
115         }
116
117         if (comPortValue.contains("EchoSounder")) {
118            ArduinoIOThread = new Thread(new ReadSerialData(dh, comPortKey, 4800, comPortValue));
119            ArduinoIOThread.start();
120            ArduinoIOThread.setName(comPortValue);
121            System.out.println("EchoSounder found");
122
```

```
123             }
124
125             if (comPortValue.contains("ActuatorFBArduino")) {
126                 ArduinoActuatorFBThread = new Thread(new ReadSerialData(dh, comPortKey, 38400, comPortValue));
127                 ArduinoActuatorFBThread.start();
128                 ArduinoActuatorFBThread.setName(comPortValue);
129                 System.out.println("ActuatorFBArduino found");
130
131             }
132
133         }
134         System.out.println("Done");
135
136         dh.setFb_ROVReady(true);
137         try {
138             dh.setCmd_BlueLED(1);
139             Thread.sleep(500);
140             dh.setCmd_BlueLED(0);
141         } catch (Exception e) {
142         }
143
144         while (true) {
145             try {
146
147             } catch (Exception e) {
148             }
149         }
150
151     }
152 }
153
```

D:\Dokumenter\Skule\04 - NTNU\Bachelor\Github\TowedROV\src\SerialCom\SerialDataHandler.java

```
1 /*
2  * This code is for the bachelor thesis named "Towed-ROV".
3  * The purpose is to build a ROV which will be towed behind a surface vessel
4  * and act as a multi-sensor platform, were it shall be easy to place new
5  * sensors. There will also be a video stream from the ROV.
6  *
7  * The system consists of two Raspberry Pis in the ROV that is connected to
8  * several Arduino micro controllers. These micro controllers are connected to
9  * feedback from the actuators, the echo sounder and extra optional sensors.
10  * The external computer which is on the surface vessel is connected to a GPS,
11  * echo sounder over USB, and the ROV over ethernet. It will present and
12  * log data in addition to handle user commands for controlling the ROV.
13  */
14 package SerialCom;
15
16 import ROV.Data;
17 import java.util.HashMap;
18 import java.util.Map.Entry;
19 import jssc.SerialPort;
20 import jssc.SerialPortList;
21
22 /**
23  * This class is resposible for handling the search for com ports and store a
24  * list over them.
25  */
26 public class SerialDataHandler {
27
28     private HashMap<String, String> portNamesList = new HashMap<>();
29
30     String comPort = "";
31     SerialPort serialPort;
32     Data data;
33
34     String start_char = "<";
35     String end_char = ">";
36     String sep_char = ":";
37
38     /**
39      * The constructor for the SerialDataHandler class
40      *
41      * @param data the shared recource data class
42      */
43     public SerialDataHandler(Data data) {
44         this.data = data;
45     }
46
47     private void saveUsableComPorts() {
48         int comCheck = 0;
49         for (Entry e : portNamesList.entrySet()) {
50             String comPortKey = (String) e.getKey();
51             String comPortValue = (String) e.getValue();
52             if (!comPortValue.contains("Unknown")) {
53                 data.comPortList.put(comPortKey, comPortValue);
54 //                comCheck++;
55             }
56         }
57 //        if (comCheck < 3)
58 //        {
59 //            //Not all comports was found
60 //            System.out.println("ERROR: Not all com ports was found, trying again...");
61 //            findComPorts();
62 //        }
63
64     }
65
66     /**
```

```
67     * This method is responsible or finding the desired com ports. Since the
68     * com ports may change we have to search and store which com port is
69     * connected to which device
70     */
71    public void findComPorts() {
72        int baudrate = 0;
73        int searchRuns = 0;
74
75        String[] portNames = getAvailableComPorts();
76        for (int i = 0; i < portNames.length; i++) {
77            //
78            if (portNames[i].contains("dev") && !portNames[i].contains("AMA0")) {
79                portNamesList.put(portNames[i], "Unknown");
80            }
81        }
82        while (searchRuns != 3) {
83
84            if (searchRuns == 0) {
85
86                baudrate = 38400;
87            }
88            if (searchRuns == 1) {
89                baudrate = 4800;
90            }
91            if (searchRuns == 2) {
92                baudrate = 115200;
93            }
94
95            for (Entry e : portNamesList.entrySet()) {
96
97                String comPortKey = (String) e.getKey();
98                String comPortValue = (String) e.getValue();
99                if (comPortValue.contains("Unknown")) {
100                   serialPort = new SerialPort(comPortKey);
101
102                   try {
103                       serialPort.openPort();
104                       serialPort.setParams(baudrate, 8, 1, 0);
105                       String buffer = "";
106                       Thread.sleep(5000);
107                       buffer = serialPort.readString();
108
109                       if (buffer != null) {
110
111                           if (buffer.contains("<") && buffer.contains(">")) {
112                               buffer = buffer.substring(buffer.indexOf(start_char) + 1);
113                               buffer = buffer.substring(0, buffer.indexOf(end_char));
114                               //  buffer = buffer.replace("?", "");
115                               String[] data = buffer.split(sep_char);
116
117                               for (int i = 0; i < data.length; i = i + 2) {
118                                   if (data[i].contains("Roll")) {
119                                       String key = (String) e.getKey();
120                                       portNamesList.put(key, "IMU");
121                                   }
122                                   if (data[i].contains("EchoSounder")) {
123                                       String key = (String) e.getKey();
124                                       portNamesList.put(key, "EchoSounder");
125                                   }
126                                   if (data[i].contains("ActuatorFBArduino")
127                                           || data[i].contains("PsActuatorFb")) {
128                                       String key = (String) e.getKey();
129                                       portNamesList.put(key, "ActuatorFBArduino");
130                                   }
131   //                              if (data[i].contains("EchoSounder"))
132   //                              {
133   //                                  String key = (String) e.getKey();
134   //                                  portNamesList.put(key, "EchoSounder");
135   //                              }
```

```
136                         }
137
138                     }
139                 }
140                 serialPort.closePort();
141
142             } catch (Exception ex) {
143                 try {
144                     serialPort.closePort();
145                 } catch (Exception exep) {
146                 }
147
148             }
149         }
150     }
151     searchRuns++;
152 }
153 saveUsableComPorts();
154
155 }
156
157 private String[] getAvailableComPorts() {
158     // getting serial ports list into the array
159
160     String[] portNames = SerialPortList.getPortNames();
161
162     if (portNames.length == 0) {
163         System.out.println("There are no serial-ports :( You can use an emulator, such ad VSPE, to create a virtual serial port.");
164         System.out.println("Press Enter to exit...");
165         try {
166             System.in.read();
167         } catch (Exception e) {
168             // TODO Auto-generated catch block
169             e.printStackTrace();
170         }
171     }
172     return portNames;
173 }
174 }
175
```

D:\Dokumenter\Skule\04 - NTNU\Bachelor\Github\TowedROV\src\ROV\TCPCom\Server.java

```
1  /*
2   * This code is for the bachelor thesis named "Towed-ROV".
3   * The purpose is to build a ROV which will be towed behind a surface vessel
4   * and act as a multi-sensor platform, were it shall be easy to place new
5   * sensors. There will also be a video stream from the ROV.
6   *
7   * The system consists of two Raspberry Pis in the ROV that is connected to
8   * several Arduino micro controllers. These micro controllers are connected to
9   * feedback from the actuators, the echo sounder and extra optional sensors.
10   * The external computer which is on the surface vessel is connected to a GPS,
11   * echo sounder over USB, and the ROV over ethernet. It will present and
12   * log data in addition to handle user commands for controlling the ROV.
13   */
14  package ROV.TCPCom;
15
16  import ROV.*;
17  import ROV.AlarmSystem.AlarmHandler;
18  import java.net.ServerSocket;
19  import java.net.Socket;
20  import java.io.IOException;
21
22  /**
23   * Code inspired from
24   * http://tutorials.jenkov.com/java-multithreaded-servers/multithreaded-server.html
25   *
26   * This class is responsible for reciving connection and handle them in a new
27   * seperat threads
28   *
29   */
30  public class Server implements Runnable {
31
32      protected int serverPort;
33      protected ServerSocket serverSocket = null;
34      protected boolean isStopped = false;
35      protected Thread runningThread = null;
36      Data dh = null;
37      AlarmHandler alarmHandler = null;
38
39      /**
40       * Constructor for server class
41       *
42       * @param port the port the server is running on
43       * @param dh the shared recource data class
44       */
45      public Server(int port, Data dh) {
46          this.serverPort = port;
47          this.dh = dh;
48      }
49
50      /**
51       * Responsible for reciving connection and handle them in a new seperat
52       * threads
53       */
54      public void run() {
55          synchronized (this) {
```

```
56          this.runningThread = Thread.currentThread();
57      }
58      openServerSocket();
59      while (!isStopped()) {
60          Socket clientSocket = null;
61          try {
62              clientSocket = this.serverSocket.accept();
63          } catch (IOException e) {
64              if (isStopped()) {
65                  System.out.println("Server Stopped.");
66                  return;
67              }
68              throw new RuntimeException(
69                      "Error accepting client connection", e);
70          }
71          new Thread(
72                  new WorkerRunnable(clientSocket, dh)
73          ).start();
74      }
75      System.out.println("Server Stopped.");
76  }
77
78  private synchronized boolean isStopped() {
79      return this.isStopped;
80  }
81
82  /**
83   * Responsible for shutting down the server
84   */
85  public synchronized void stop() {
86      this.isStopped = true;
87      try {
88          this.serverSocket.close();
89      } catch (IOException e) {
90          throw new RuntimeException("Error closing server", e);
91      }
92  }
93
94  private void openServerSocket() {
95      try {
96          this.serverSocket = new ServerSocket(this.serverPort);
97      } catch (IOException e) {
98          throw new RuntimeException("Cannot open port 8080", e);
99      }
100   }
101
102 }
103
```

D:\Dokumenter\Skule\04 - NTNU\Bachelor\Github\TowedROV\src\ROV\StartupCalibration.java

```
1 /*
2  * This code is for the bachelor thesis named "Towed-ROV".
3  * The purpose is to build a ROV which will be towed behind a surface vessel
4  * and act as a multi-sensor platform, were it shall be easy to place new
5  * sensors. There will also be a video stream from the ROV.
6  *
7  * The system consists of two Raspberry Pis in the ROV that is connected to
8  * several Arduino micro controllers. These micro controllers are connected to
9  * feedback from the actuators, the echo sounder and extra optional sensors.
10  * The external computer which is on the surface vessel is connected to a GPS,
11  * echo sounder over USB, and the ROV over ethernet. It will present and
12  * log data in addition to handle user commands for controlling the ROV.
13  */
14 package ROV;
15
16 /**
17  * Startup calibration of the ROV. Used for monitoring the speed of wings on the
18  * rov incase of performance loss between startups.
19  *
20  * Not yet implemented
21  */
22 public class StartupCalibration {
23
24     Data dh = null;
25     long currentTime = 0;
26     long lastTimePS = 0;
27     long lastTimeSB = 0;
28
29     public StartupCalibration(Data dh) {
30         this.dh = dh;
31     }
32
33     // Startup calibration not yet implemented
34 //     public String doStartupCalibration()
35 //     {
36 //         calibrateActuators();
37 //         testLights();
38 //
39 //         return "Calibration complete...";
40 //     }
41 //
42 //     public void calibrateActuators()
43 //     {
44 //         int accuracy = 4;
45 //         int lastActuatorPSPos = 0;
46 //         int lastActuatorSBPos = 0;
47 //         boolean findingMinPS = true;
48 //         boolean findingMinSB = true;
49 //         boolean findingMaxPS = true;
50 //         boolean findingMaxSB = true;
51 //
52 //         currentTime = System.nanoTime();
53 //
54 //         //Move actuators to minimum pos;
55 //         dh.cmd_actuatorPS = 0;
```

```
56 //      dh.cmd_actuatorSB = 0;
57 //      while (findingMinPS && findingMinSB)
58 //
59 //      {
60 //         lastActuatorPSPos = dh.fb_actuatorPSPos;
61 //         if (lastActuatorPSPos - accuracy <= dh.fb_actuatorPSPos
62 //               && lastActuatorPSPos + accuracy >= dh.fb_actuatorPSPos
63 //               && findingMinPS)
64 //         {
65 //            lastTimePS = System.nanoTime();
66 //
67 //            if (System.nanoTime() - lastTimePS >= 500000000)
68 //            {
69 //               dh.cmd_actuatorPS = dh.fb_actuatorPSPos;
70 //               dh.cmd_actuatorPSMinPos = dh.fb_actuatorPSPos;
71 //               findingMinPS = false;
72 //            }
73 //
74 //         }
75 //
76 //         lastActuatorSBPos = dh.fb_actuatorSBPos;
77 //         if (lastActuatorSBPos - accuracy <= dh.fb_actuatorSBPos
78 //               && lastActuatorSBPos + accuracy >= dh.fb_actuatorSBPos
79 //               && findingMinSB)
80 //         {
81 //            lastTimeSB = System.nanoTime();
82 //
83 //            if (System.nanoTime() - lastTimeSB >= 500000000)
84 //            {
85 //               dh.cmd_actuatorSB = dh.fb_actuatorSBPos;
86 //               dh.cmd_actuatorSBMinPos = dh.fb_actuatorSBPos;
87 //               findingMinSB = false;
88 //            }
89 //         }
90 //      }
91 //
92 //      //Move actuators to maximum pos
93 //      dh.cmd_actuatorPS = 4000;
94 //      dh.cmd_actuatorSB = 4000;
95 //      while (findingMaxPS && findingMaxSB)
96 //
97 //      {
98 //         lastActuatorPSPos = dh.fb_actuatorPSPos;
99 //         if (lastActuatorPSPos - accuracy <= dh.fb_actuatorPSPos
100 //               && lastActuatorPSPos + accuracy >= dh.fb_actuatorPSPos
101 //               && findingMaxPS)
102 //         {
103 //            lastTimePS = System.nanoTime();
104 //
105 //            if (System.nanoTime() - lastTimePS >= 500000000)
106 //            {
107 //               dh.cmd_actuatorPS = dh.fb_actuatorPSPos;
108 //               dh.cmd_actuatorPSMaxPos = dh.fb_actuatorPSPos;
109 //               findingMaxPS = false;
110 //            }
111 //
112 //         }
113 //
```

```
114 //          lastActuatorSBPos = dh.fb_actuatorSBPos;
115 //          if (lastActuatorSBPos - accuracy <= dh.fb_actuatorSBPos
116 //                && lastActuatorSBPos + accuracy >= dh.fb_actuatorSBPos
117 //                && findingMaxSB)
118 //          {
119 //             lastTimeSB = System.nanoTime();
120 //
121 //             if (System.nanoTime() - lastTimeSB >= 500000000)
122 //             {
123 //                dh.cmd_actuatorSB = dh.fb_actuatorSBPos;
124 //                dh.cmd_actuatorSBMinPos = dh.fb_actuatorSBPos;
125 //                findingMaxSB = false;
126 //             }
127 //
128 //          }
129 //       }
130 //
131 //       //Set wings to neutraPos
132 //       int middlePSPos = dh.getCmd_actuatorPSMaxPos() - dh.getCmd_actuatorPSMinPos();
133 //       dh.cmd_actuatorPS = middlePSPos;
134 //
135 //       int middleSBPos = dh.getCmd_actuatorSBMaxPos() - dh.getCmd_actuatorSBMinPos();
136 //       dh.cmd_actuatorSB = middleSBPos;
137 //    }
138 //
139 //    //Test lights
140 //    public void testLights()
141 //    {
142 //       boolean testingLights = true;
143 //       long lastTime = System.nanoTime();
144 //       int lightIntensity = 1;
145 //       while (testingLights && lightIntensity < 100)
146 //       {
147 //          if (System.nanoTime() - lastTime >= 250000000)
148 //          {
149 //             dh.cmd_lightMode = lightIntensity + 1;
150 //             lastTime = System.nanoTime();
151 //          }
152 //
153 //       }
154 //       try
155 //       {
156 //          Thread.sleep(5000);
157 //          dh.cmd_lightMode = 0;
158 //       } catch (Exception e)
159 //       {
160 //       }
161 //
162 //    }
163 }
164
```

D:\Dokumenter\Skule\04 - NTNU\Bachelor\Github\TowedROV\src\ROV\AlarmSystem\TimeBasedAlarms.java

```java
/*
 * This code is for the bachelor thesis named "Towed-ROV".
 * The purpose is to build a ROV which will be towed behind a surface vessel
 * and act as a multi-sensor platform, were it shall be easy to place new
 * sensors. There will also be a video stream from the ROV.
 *
 * The system consists of two Raspberry Pis in the ROV that is connected to
 * several Arduino micro controllers. These micro controllers are connected to
 * feedback from the actuators, the echo sounder and extra optional sensors.
 * The external computer which is on the surface vessel is connected to a GPS,
 * echo sounder over USB, and the ROV over ethernet. It will present and
 * log data in addition to handle user commands for controlling the ROV.
 */
package ROV.AlarmSystem;

/**
 * The timed based alarms of the rov
 *
 * Not yet implemented
 */
public class TimeBasedAlarms implements Runnable {

    public void run() {
        while (true) {

        }
    }

}

```

D:\Dokumenter\Skule\04 - NTNU\Bachelor\Github\TowedROV\src\ROV\TCPCom\WorkerRunnable.java

```
1  /*
2   * This code is for the bachelor thesis named "Towed-ROV".
3   * The purpose is to build a ROV which will be towed behind a surface vessel
4   * and act as a multi-sensor platform, were it shall be easy to place new
5   * sensors. There will also be a video stream from the ROV.
6   *
7   * The system consists of two Raspberry Pis in the ROV that is connected to
8   * several Arduino micro controllers. These micro controllers are connected to
9   * feedback from the actuators, the echo sounder and extra optional sensors.
10  * The external computer which is on the surface vessel is connected to a GPS,
11  * echo sounder over USB, and the ROV over ethernet. It will present and
12  * log data in addition to handle user commands for controlling the ROV.
13  */
14 package ROV.TCPCom;
15
16 import java.io.BufferedReader;
17 import java.io.InputStream;
18 import java.io.OutputStream;
19 import java.io.IOException;
20 import java.io.InputStreamReader;
21 import java.io.PrintWriter;
22 import java.net.Socket;
23 import ROV.*;
24 import java.util.Map;
25 import java.util.Map.Entry;
26
27 /**
28  *
29  * Code taken from:
30  * http://tutorials.jenkov.com/java-multithreaded-servers/multithreaded-server.html
31  *
32  * Responsible for handling data from the client and answering the client. All
33  * client commands are listed in the switch case
34  *
35  */
36 public class WorkerRunnable implements Runnable {
37
38     protected Socket clientSocket = null;
39
40     Data dh = null;
41
42     StartupCalibration StartupCalibration = null;
43     String start_char = "<";
44     String end_char = ">";
45     String sep_char = ":";
46
47     /**
48      * The constructor for the WorkerRunnable class
49      *
50      * @param clientSocket The socket the client is connecte dto
51      *
52      * @param dh the shared recource data class
53      */
54     public WorkerRunnable(Socket clientSocket, Data dh) {
55         this.clientSocket = clientSocket;
56
57         this.dh = dh;
58
```

```
59  }
60
61  /**
62   * Responsible for handling data from the client and answering the client.
63   * All client commands are listed in the switch case
64   */
65  public void run() {
66      boolean clientOnline = true;
67 //       boolean welcomeMessageIsSent = false;
68      try {
69          BufferedReader inFromClient = new BufferedReader(
70                  new InputStreamReader(
71                          this.clientSocket.getInputStream()));
72
73          PrintWriter outToClient = new PrintWriter(
74                  this.clientSocket.getOutputStream(), true);
75
76          InputStream input = clientSocket.getInputStream();
77          OutputStream output = clientSocket.getOutputStream();
78
79          while (clientOnline) {
80              if (inFromClient.ready()) {
81                  String key = "";
82                  String value = "";
83                  String inputData = inFromClient.readLine();
84                  if (inputData.contains("<") && inputData.contains(">")) {
85                      inputData = inputData.substring(inputData.indexOf(start_char) + 1);
86                      inputData = inputData.substring(0, inputData.indexOf(end_char));
87                      inputData = inputData.replace("?", "");
88                      if (inputData.contains(":")) {
89                          String[] data = inputData.split(sep_char);
90                          key = data[0];
91                          value = data[1];
92                      } else {
93                          String[] data = inputData.split(sep_char);
94                          key = data[0];
95                      }
96
97                  } else {
98                      key = (String) inputData;
99                  }
100                 if (!dh.getFb_ROVReady()) {
101                     outToClient.println("Server: ROV not ready");
102                 } else {
103
104                     switch (key) {
105                         //Commands
106
107                         case "cmd_lightMode":
108                             dh.setCmd_lightMode(parseStringToInt(value));
109                             System.out.println("LightMode: " + dh.getCmd_lightMode());
110                             outToClient.println("Server: OK");
111                             break;
112
113                         case "cmd_actuatorPS":
114                             dh.setCmd_actuatorPS(parseStringToInt(value));
115                             System.out.println("actuatorPS is: " + dh.getCmd_actuatorPS());
116                             outToClient.println("Server: OK");
117
118                             break;
119
```

```
120                 case "cmd_actuatorSB":
121                     dh.setCmd_actuatorSB(parseStringToInt(value));
122                     System.out.println("actuatorSB is: " + dh.getCmd_actuatorSB());
123                     outToClient.println("Server: OK");
124                     break;
125 
126                 case "actuator_test":
127                     dh.setCmd_actuatorSB(parseStringToInt(value));
128 
129                     dh.setCmd_actuatorPS(parseStringToInt(value));
130 
131                     //System.out.println("actuatorSB is: " + dh.getCmd_actuatorSB());
132                     outToClient.println("Server: OK");
133 
134                 case "cmd_targetDistance":
135                     dh.setCmd_targetDistance(Double.valueOf(value));
136                     outToClient.println("Server: OK");
137                     break;
138 
139                 case "cmd_pid_p":
140                     dh.setCmd_pid_p(Double.valueOf(value));
141 //                      System.out.println("Pid_p is: " + dh.getCmd_pid_p());
142                     outToClient.println("Server: OK");
143                     break;
144 
145                 case "cmd_pid_i":
146                     dh.setCmd_pid_i(Double.valueOf(value));
147 //                      System.out.println("Pid_i is: " + dh.getCmd_pid_i());
148                     outToClient.println("Server: OK");
149                     break;
150 
151                 case "cmd_pid_d":
152                     dh.setCmd_pid_d(Double.valueOf(value));
153 //                      System.out.println("Pid_d is: " + dh.getCmd_pid_d());
154                     outToClient.println("Server: OK");
155                     break;
156 
157                 case "cmd_pid_gain":
158                     dh.setCmd_pid_gain(Double.valueOf(value));
159 //                      System.out.println("Pid_gain is: " + dh.getCmd_pid_gain());
160                     outToClient.println("Server: OK");
161                     break;
162 
163                 case "cmd_emergencySurface":
164                     dh.setCmd_emergencySurface(parseStringToBoolean(value));
165                     System.out.println("EmergencySurface is: " + dh.isCmd_emergencySurface());
166                     outToClient.println("Server: OK");
167                     break;
168 
169                 case "cmd_BlueLED":
170                     dh.setCmd_BlueLED(parseStringToInt(value));
171                     outToClient.println("Server: OK");
172                     break;
173 
174                 case "cmd_rovDepth":
175                     dh.setCmd_currentROVdepth(Double.valueOf(value));
176                     outToClient.println("Server: OK");
177                     break;
178 
179                 case "cmd_targetMode":
180                     dh.setcmd_targetMode(parseStringToInt(value));
```

```
181                     outToClient.println("Server: OK");
182                     break;
183
184                 case "cmd_offsetDepthBeneathROV":
185                     dh.setCmd_offsetDepthBeneathROV(Double.valueOf(value));
186                     outToClient.println("Server: OK");
187                     break;
188                 case "cmd_offsetROVdepth":
189                     dh.setCmd_offsetROVdepth(Double.valueOf(value));
190                     outToClient.println("Server: OK");
191                     break;
192
193                 //Feedback commands
194                 case "fb_allData":
195                     outToClient.println(dh.getDataToSend());
196                     //System.out.println("Sent all data");
197                     break;
198
199                 case "fb_depthToSeabedEcho":
200                     outToClient.println("<Fb_depthBeneathROV:" + dh.getFb_depthBeneathROV() + ">");
201                     break;
202
203                 case "fb_speedThroughWather":
204                     outToClient.println("<fb_speedThroughWather:" + dh.getFb_speedThroughWather() + ">");
205                     break;
206
207                 case "fb_waterTemperature":
208                     outToClient.println("<fb_waterTemperature:" + dh.getFb_waterTemperature() + ">");
209                     break;
210
211                 case "fb_actuatorPSPos":
212                     outToClient.println("<fb_actuatorPSPos:" + dh.getFb_actuatorPSPos() + ">");
213                     break;
214
215                 case "fb_actuatorSBPos":
216                     outToClient.println("<fb_actuatorSBPos:" + dh.getFb_actuatorSBPos() + ">");
217                     break;
218
219                 case "fb_tempMainElBoxFront":
220                     outToClient.println("<fb_tempMainElBox:" + dh.getFb_tempMainElBoxFront() + ">");
221                     break;
222
223                 case "fb_tempMainElBoxRear":
224                     outToClient.println("<fb_tempMainElBox:" + dh.getFb_tempMainElBoxRear() + ">");
225                     break;
226
227                 case "fb_currentDraw":
228                     outToClient.println("<fb_currentDraw:" + dh.getFb_currentDraw() + ">");
229                     break;
230
231                 case "fb_pitchAngel":
232                     outToClient.println("<fb_pitchAngle:" + dh.getFb_pitchAngle() + ">");
233                     break;
234
235                 case "fb_rollAngle":
236                     outToClient.println("<fb_rollAngle:" + dh.getFb_rollAngle() + ">");
237                     break;
238
239                 case "fb_heading":
240                     outToClient.println("<fb_heading:" + dh.getFb_heading() + ">");
241                     break;
```

```
242
243                 //Stored Commands
244                 case "get_cmd_lightMode":
245                     outToClient.println("<get_cmd_lightMode:" + dh.getCmd_lightMode() + ">");
246                     break;
247
248                 case "get_cmd_actuatorPS":
249                     outToClient.println("<get_cmd_actuatorPS:" + dh.getCmd_actuatorPS() + ">");
250                     break;
251
252                 case "get_cmd_actuatorSB":
253                     outToClient.println("<get_cmd_actuatorSB:" + dh.getCmd_actuatorSB() + ">");
254                     break;
255
256                 case "get_cmd_pid_p":
257                     outToClient.println("<get_cmd_pid_p:" + dh.getCmd_pid_p() + ">");
258                     break;
259
260                 case "get_cmd_pid_i":
261                     outToClient.println("<get_cmd_pid_i:" + dh.getCmd_pid_i() + ">");
262                     break;
263
264                 case "get_cmd_pid_d":
265                     outToClient.println("<get_cmd_pid_d:" + dh.getCmd_pid_d() + ">");
266                     break;
267
268                 case "get_cmd_pid_gain":
269                     outToClient.println("<get_cmd_pid_gain:" + dh.getCmd_pid_gain() + ">");
270                     break;
271
272                 case "get_ROVComPorts":
273                     String portListString = "<";
274                     for (Entry e : dh.comPortList.entrySet()) {
275                         String comPortKey = (String) e.getKey();
276                         String comPortValue = (String) e.getValue();
277                         portListString = portListString + comPortKey + ":" + comPortValue + ":";
278                     }
279                     portListString = portListString + ">";
280                     outToClient.println(portListString);
281                     break;
282
283                 //Other  commands
284                 case "ping":
285                     //output.write(("<ping:true>").getBytes());
286                     dh.setCmd_ping(true);
287                     outToClient.println("<ping:true>");
288 //                    welcomeMessageIsSent = true;
289                     break;
290
291                 case "ack":
292                     if (dh.isCmd_ack()) {
293                         dh.setCmd_ack(false);
294                     } else {
295                         dh.setCmd_ack(true);
296                     }
297                     outToClient.println("Ack: " + dh.isCmd_ack());
298                     break;
299
300                 case "getAlarms":
301                     String completeAlarmListString = "<";
302
```

```
303                         for (Map.Entry e : dh.completeAlarmListDh.entrySet()) {
304                             key = (String) e.getKey();
305                             if (e.getValue().equals(true)) {
306                                 value = "true";
307                             } else {
308                                 value = "false";
309                             }
310 
311                             completeAlarmListString = completeAlarmListString + key + ":" + value + ":";
312                         }
313                         outToClient.println(completeAlarmListString + ">");
314                         break;
315 
316                     case "exit":
317                         output.close();
318                         input.close();
319                         clientOnline = false;
320                         break;
321 
322                     default:
323                         outToClient.println("Error: Not a command");
324                         break;
325 
326                 }
327             }
328         }
329 
330     }
331 
332     } catch (IOException e) {
333         //report exception somewhere.
334         System.out.println("Exception: " + e);
335         e.printStackTrace();
336     }
337 }
338 
339 private boolean parseStringToBoolean(String value) {
340     Boolean result = false;
341     try {
342         result = Boolean.valueOf(value);
343     } catch (Exception e) {
344         System.out.println("Exception while parsing to double");
345     }
346     return result;
347 }
348 
349 private Integer parseStringToInt(String value) {
350     Integer result = 0;
351 
352     try {
353         result = Integer.valueOf(value);
354     } catch (Exception e) {
355         System.out.println("Exception while parsing to integer");
356     }
357 
358     return result;
359 }
360 
361 }
362 
```

L Python code

```python
# TcpController.py    -    TCP Server and sensor reader code

import time
import ms5837
import smbus
import RPi.GPIO as GPIO
import socket
import threading
import sys
import os
import shutil

# values that can be set from remote:
cameraPitch = 0.01
ledLights = 0.01

# data variables to be sent to the client:
leakAlarm = False
depth = 0.01
pressure = 0.01
outsideTemp = 0.01
insideTemp = 0.01
humidity = 0.01


cameraPitchDutyCycle = 0
ledLightsDutyCycle = 0

PIN_CameraPitch = 40 # pin number 40 used for GIMBAL PITCH
PIN_LedLights = 38 # pin number 38 used for LED LIGHTS
PIN_WaterLeak = 36 # pin number 36 used for WATER LEAKAGE

GPIO.setmode(GPIO.BOARD)
GPIO.setwarnings(False)
GPIO.setup(PIN_CameraPitch, GPIO.OUT)
GPIO.setup(PIN_LedLights, GPIO.OUT)
GPIO.setup(PIN_WaterLeak, GPIO.IN, pull_up_down=GPIO.PUD_DOWN)
pwm_pitch = GPIO.PWM(PIN_CameraPitch, 400) # PIN_CameraPitch = GPIO pin 40, frequency 400
pwm_pitch.start(0)
pwm_led = GPIO.PWM(PIN_LedLights, 200) # PIN_CameraPitch = GPIO pin 38, frequency 200
pwm_led.start(0)

# Reads the outside pressure and temp, depth, and the humidity and temp inside the camera housing.
def SensorReader():
    print("SensorReader thread started.")

    global depth
    global pressure
    global outsideTemp
    global insideTemp
    global humidity

    try:
        # Get I2C bus number 1
        bus = smbus.SMBus(1)

        ###################################################################
        # Setting up the sensor for outside pressure, temp and depth
        # Used i2c address 76 (0x76)
        sensor = ms5837.MS5837_30BA() # Default I2C bus is 1 (Raspberry Pi 3)

        # We must initialize the sensor before reading it
        if not sensor.init():
                print("Sensor could not be initialized")
                exit(1)

        # We have to read values from sensor to update pressure and temperature
        if not sensor.read():
            print("Sensor read failed!")
            exit(1)
```

```
71
72          print("\nPressure: {0} mbar \t{1} atm".format(
73          round(sensor.pressure(), 2),
74          round(sensor.pressure(ms5837.UNITS_atm), 2)))
75          print("Temperature: {0} C".format(
76          round(sensor.temperature(ms5837.UNITS_Centigrade), 2)))
77          freshwaterDepth = sensor.depth() # default is freshwater
78          sensor.setFluidDensity(ms5837.DENSITY_SALTWATER)
79          saltwaterDepth = sensor.depth() # No nead to read() again
80          #sensor.setFluidDensity(1000) # kg/m^3
81          print("Depth: {0} m (freshwater)  {1} m (saltwater)".format(
82          round(freshwaterDepth, 3),
83          round(saltwaterDepth, 3)))
84          # fluidDensity doesn't matter for altitude() (always MSL air density)
85          print("MSL Relative Altitude: {0} m".format(round(sensor.altitude(), 2))) #
            relative to Mean Sea Level pressure in air
86          ######################################################################
87          time.sleep(2)
88
89          # Spew readings
90          while True:
91
92              bus.write_quick(0x27) # i2c address 27
93              time.sleep(0.1)
94
95              # HIH6130 address, 0x27(39)
96              # Read data back from 0x00(00), 4 bytes
97              # humidity MSB, humidity LSB, temp MSB, temp LSB
98              data = bus.read_i2c_block_data(0x27, 0x00, 4)
99
100             # Convert the data to 14-bits
101             humidity = ((((data[0] & 0x3F) * 256) + data[1]) * 100.0) / 16383.0
102             temp = (((data[2] & 0xFF) * 256) + (data[3] & 0xFC)) / 4
103             insideTemp = (temp / 16384.0) * 165.0 - 40.0 # Convert it to degrees
                Celsius
104             #fTemp = cTemp * 1.8 + 32
105             humidity = round(humidity, 2)
106             insideTemp = round(insideTemp, 2)
107
108             #print("Relative Humidity: {0} %% \tInside Temp: {1} C".format(
109             #round(humidity, 2),
110             #round(insideTemp, 2)))
111             ##print("Temperature in Celsius: {0} C".format())
112             ##print("Temperature in Fahrenheit: {0} F\n".format(round(fTemp, 2)))
113
114             if sensor.read():
115
116                 ## Sensor for outside pressure, temp and depth:
117                 pressure = round(sensor.pressure(), 2)
118                 outsideTemp = round(sensor.temperature(), 2)
119                 depth = round(sensor.depth(), 2)
120
121                 #print("Pressure: {0} mbar \tOutside Temp: {1} C \tDepth: {2}
                    m\n".format(
122                 #pressure, # Default is mbar (no arguments)
123                 #outsideTemp, # Default is degrees C (no arguments)
124                 #depth)) # Default is depth in meters (no arguments)
125
126             else:
127                 print("Sensor read failed!")
128                 exit(1)
129
130             #time.sleep(0.1)
131
132     except (Exception, KeyboardInterrupt) as e:
133         #pass
134         #connection.close()
135         pwm_pitch.stop()
136         pwm_led.stop()
137         GPIO.cleanup()
138     finally:
139         # Clean up the connection
```

```
            #pass
            #connection.close()
            pwm_pitch.stop()
            pwm_led.stop()
            GPIO.cleanup()


    # Start the SensorReader thread
    sensorReaderThread = threading.Thread(target=SensorReader)
    sensorReaderThread.daemon = True
    sensorReaderThread.start()



    def LeakAlarmListener():
        print("LeakAlarmListener thread started.")
        global leakAlarm
        time.sleep(2)
        try:
            while True:
                leakAlarm = GPIO.input(PIN_WaterLeak)
                #print("Leak alarm: {0}".format(leakAlarm))
                time.sleep(0.1)
        except (Exception, KeyboardInterrupt) as e:
            connection.close()
            pwm_pitch.stop()
            pwm_led.stop()
            GPIO.cleanup()
        finally:
            # Clean up the connection
            connection.close()
            pwm_pitch.stop()
            pwm_led.stop()
            GPIO.cleanup()

    # Start the listening thread.
    leakThread = threading.Thread(target=LeakAlarmListener)
    leakThread.daemon = True
    leakThread.start()


    # Setting up the server
    print('Setting up the server...')
    sock = socket.socket(family=socket.AF_INET, type=socket.SOCK_STREAM)
    #server_address = ('169.254.196.33', 9006) # ROV Camera RPi
    #server_address = ('169.254.99.196', 9006) # Bjørnars RPi
    #server_address = ('10.16.4.187', 9006) # Bjørnars RPi WIFI
    #server_address = ('10.16.8.140', 9006) # ROV Camera RPi WIFI
    server_address = ('192.168.0.102', 9006) # ROV Camera RPi Static IP address to be used
    print('Starting up on {} port {} ' .format(*server_address))
    sock.bind(server_address)
    sock.listen(1)

    # Listen for incoming connections and do the work
    while True:
        print('Waiting for connection...')
        connection, client_address = sock.accept()
        try:
            print('connection from', client_address)
                # Receive the data in small chunks and retransmit it
            while True:

                data = connection.recv(4096)
                dataString = str(data, 'utf-8').rstrip("\r\n")
                print('Received:', dataString)
                if data:
                    if dataString == "<getData>":
                        dataToBeSent =
                        "<leakAlarm:{0}:depth:{1}:pressure:{2}:outsideTemp:{3}:insideTemp:
                        {4}:humidity:{5}>\r\n".format(
                        leakAlarm, depth, pressure, outsideTemp, insideTemp, humidity)

```

```
210                     bytesToBeSent = dataToBeSent.encode()
211                     print('Sending back data:', dataToBeSent.rstrip("\r\n"))
212                     connection.sendall(bytesToBeSent)
213                 elif "<setPitch:" in dataString:
214                         dataString = dataString.replace("<","")
215                         dataString = dataString.replace(">","")
216                         #print(dataString)
217                         arr = dataString.split(":")
218                         value = arr[1]
219                         intValue = int(value)
220                         if intValue < 50: # min value for the LED
221                             value = 50
222                         elif intValue > 75: # max value for the LED
223                             value = 75
224                         cameraPitchDutyCycle = float(value)
225                         pwm_pitch.ChangeDutyCycle(float(cameraPitchDutyCycle))
226                         print("cameraPitch has been set to {0} from the
                            GUI!".format(cameraPitchDutyCycle))
227                         replyMessage = 'cameraPitch has been set to ' + str(value) +
                            '!\r\n'
228                         connection.sendall(replyMessage.encode())
229                 elif "<setLed:" in dataString:
230                         dataString = dataString.replace("<","")
231                         dataString = dataString.replace(">","")
232                         #print(dataString)
233                         arr = dataString.split(":")
234                         value = arr[1]
235                         intValue = int(value)
236                         if intValue < 23: # min value for the LED
237                             value = 0
238                         elif intValue > 40: # max value for the LED
239                             value = 40
240                         ledLightsDutyCycle = float(value)
241                         pwm_led.ChangeDutyCycle(ledLightsDutyCycle)
242                         print("ledLights has been set to {0} from the
                            GUI!".format(ledLightsDutyCycle))
243                         replyMessage = 'ledLights has been set to ' + str(value) +
                            '!\r\n'
244                         connection.sendall(replyMessage.encode())
245                 elif "<clearImages>" in dataString:
246                     dataString = dataString.replace("<","")
247                     dataString = dataString.replace(">","")
248                     # clear the image folder
249                     numCleared = 0
250                     numAmount = 0
251                     folder = '/home/pi/ftp/images'
252                     for the_file in os.listdir(folder):
253                         numAmount = numAmount + 1
254                         file_path = os.path.join(folder, the_file)
255                         try:
256                             if os.path.isfile(file_path):
257                                 os.unlink(file_path)
258                                 numCleared = numCleared + 1
259                             #elif os.path.isdir(file_path): shutil.rmtree(file_path)
260                         except Exception as e:
261                             print(e)
262                     print('Cleared the image folder.')
263                     replyMessage = 'Cleared the image folder: ' + str(numCleared) +
                        " of " + str(numAmount) + " images cleared." + '\r\n'
264                     connection.sendall(replyMessage.encode())
265 
266                 else:
267                     print('Wrong command received: ' + dataString)
268                     connection.sendall(b'Wrong command! Only <getData>,
                        <setPitch:value> and <setLed:value> and <clearImages> will work
                        on this device.\r\n')
269             else:
270                 print('no data from', client_address)
271                 break
272     except (Exception, KeyboardInterrupt) as e:
273         connection.close()
274         #pwm_pitch.stop()
```

```
275             #pwm_led.stop()
276             #GPIO.cleanup()
277         finally:
278             # Clean up the connection
279             connection.close()
280             #pwm_pitch.stop()
281             #pwm_led.stop()
282             #GPIO.cleanup()
283 
284 
```

```python
# udpvs.py    -    Video Stream source code

from picamera.array import PiRGBArray
from picamera import PiCamera
import socket
import time
import cv2
import threading
import os
import shutil

# initialize the camera and grab a reference to the raw camera capture.
camera = PiCamera()
camera.resolution = (680, 420)
#camera.resolution = (2304, 1296)
# any higher 16:9 resolutions results in "out of resources" error. (2304, 1296)
camera.framerate = 90
camera.vflip = True
UDP_IP = "192.168.0.20"  # The static IP of the OPERATOR PC (håkon sin)
UDP_PORT = 8083
rawCapture = PiRGBArray(camera, size=(680, 420))
#rawCapture = PiRGBArray(camera, size=(2304, 1296))
photoMode = False
photoDelay = 1
imageNumber = 1

## clear the image folder
#folder = '/home/pi/ftp/images'
#for the_file in os.listdir(folder):
    #file_path = os.path.join(folder, the_file)
    #try:
        #if os.path.isfile(file_path):
            #os.unlink(file_path)
        ##elif os.path.isdir(file_path): shutil.rmtree(file_path)
    #except Exception as e:
        #print(e)
#print('Cleared the image folder.')


sock = socket.socket(family=socket.AF_INET, type=socket.SOCK_DGRAM)

# allow the camera to warmup.
time.sleep(0.1)

# listening function for the thread
def udpListener():
    global photoMode
    global photoDelay
    global imageNumber
    while True:
        print('Listening...')
        data, server = sock.recvfrom(4096)
        s = str(data, 'utf-8')
        print("Received:", s)

        if s == "photoMode:true":
            photoMode = True
            print('Set photoMode to True from GUI!')
        if s == "photoMode:false":
            photoMode = False
            print('Set photoMode to False from GUI!')
        if "photoDelay" in s:
            arr = s.split(":")
            value = arr[1]
            photoDelay = float(value)
            print('Set photoDelay to ' + str(value) + " from GUI!")
        if s == "resetImgNumber":
            imageNumber = 1
            print("Reset imageNumber to 1 from GUI!")
        time.sleep(1)

# Start the listening thread
```

```
t = threading.Thread(target=udpListener)
t.daemon = True
t.start()


while True:

    if photoMode == False:
        print('Video Mode ON')

        # capture frames from the camera
        for frame in camera.capture_continuous(rawCapture, format="bgr",
        use_video_port=True):

            #time.sleep(0.1)
            # grab the raw NumPy array representing the image - this array.
            # will be 3D, representing the width, height, and # of channels.

            image = frame.array
            #img = cv2.resize(image, (680, 420))

            # Converting image to bufferdimage of JPEG.
            x = [int(cv2.IMWRITE_JPEG_QUALITY), 80]

            # Compressniong image before sending.
            __,compressed = cv2.imencode(".jpg", image, x)

            # Sending datagramPacket through the socket.
            sock.sendto(compressed,(UDP_IP,UDP_PORT))

            # clear the stream in preparation for the next frame.
            rawCapture.truncate(0)

            # print
            #print('Video Frame sent!')

            # if the `q` key was pressed, break from the loop.
            if input == ord("q"):
                break

            if photoMode == True:
                print('Video Mode OFF')
                break

    else:
        print('Photo Mode ON')
        startTime = time.time()

        # capture frames from the camera
        for frame in camera.capture_continuous(rawCapture, format="bgr",
        use_video_port=True):

            time.sleep(photoDelay)
            # grab the raw NumPy array representing the image - this array.
            # will be 3D, representing the width, height, and # of channels.

            image = frame.array
            #img = cv2.resize(image, (680, 420))

            # Converting image to bufferdimage of JPEG.
            x = [int(cv2.IMWRITE_JPEG_QUALITY), 80]

            # Compressing image before sending.
            __,compressed = cv2.imencode(".jpg", image, x)

            # Sending datagramPacket through the socket.
            sock.sendto(compressed,(UDP_IP,UDP_PORT))

            #Save image to file
            camera.resolution = (3280, 2464)
            imagePath = '/home/pi/ftp/images/image' + str(imageNumber) + '.jpg'
            #print(imagePath)
```

```
            #cv2.imwrite(imagePath, image)
            camera.capture(imagePath)
            imageNumber = imageNumber + 1
            camera.resolution = (680, 420)

            # clear the stream in preparation for the next frame.
            rawCapture.truncate(0)

            # print
            endTime = time.time()
            print('Photo sent! ' + str(endTime - startTime))
            startTime = time.time()

            # if the `q` key was pressed, break from the loop.
            if input == ord("q"):
                break

            if photoMode == False:
                print('Photo Mode OFF')
                break
            if imageNumber > 999:
                print('Photo Mode OFF')
                photoMode = False
                break
            #print(str(imageNumber))

```

M Arduino code

```
//SerialCommunicationBaseStation.ino

// This class handles the NMEA sentences from the GPS, reads the voltage
// of the batteries, builds a string of all the data and sends it as a
// byte array over serial.

#include <Adafruit_GPS.h> // Adafruit GPS library
#include <SoftwareSerial.h> // Software serial library

SoftwareSerial mySerial(3, 2); // GPS connection

//HardwareSerial mySerial = Serial1;

Adafruit_GPS GPS(&mySerial);

#define GPSECHO false
#define NUM_SAMPLES 10

int sum = 0;                    // sum of samples taken
unsigned char sample_count = 0; // current sample number
float voltage = 0.0;            // calculated voltage
float calibratedVoltage = 0.0;

boolean usingInterrupt = false;
void useInterrupt(boolean); // Func prototype keeps Arduino 0023 happy
String data[14];


void setup()
{

 Serial.begin(115200);
 Serial.println("<GPS:0>");

 GPS.begin(9600);

GPS.sendCommand(PMTK_SET_NMEA_OUTPUT_RMCGGA);

  // Set the update rate
 GPS.sendCommand(PMTK_SET_NMEA_UPDATE_1HZ);   // 1 Hz update rate
GPS.sendCommand(PGCMD_ANTENNA);

 useInterrupt(true);

 delay(1000);
mySerial.println(PMTK_Q_RELEASE);
}
```

```
SIGNAL(TIMER0_COMPA_vect) {
  char c = GPS.read();

#ifdef UDR0
  if (GPSECHO)
    if (c) UDR0 = c;
#endif
}

void useInterrupt(boolean v) {
  if (v) {

    OCR0A = 0xAF;
    TIMSK0 |= _BV(OCIE0A);
    usingInterrupt = true;
  } else {
    TIMSK0 &= ~_BV(OCIE0A);
    usingInterrupt = false;
  }
}

// Collects all data to one string.
void sendDataOverSerial(String data[14])
{
  String dataString = "";
  dataString = String("<");



  for (byte i = 0; i < 14; i++)
  {
    dataString = String(dataString + data[i]);
    if (i < 14 - 1)
    {
      dataString = String(dataString + ":");
    }
  }
  dataString = String(dataString + ">");
 Serial.println(dataString);
  delay(1000);
}



// places data value to correct key.
void setToByteArray()
{
  data[0] = "Satellites";
  data[1] = String(GPS.satellites);
```

```
  data[2] = "Altitude";
  data[3] = String(GPS.altitude);
  data[4] = "GPSAngle";
  data[5] = String(GPS.angle);
  data[6] = "Speed";
  data[7] = String(GPS.speed);
  data[8] = "Latitude";
  data[9] = String(GPS.latitudeDegrees , 5);
  data[10] = "Longitude";
  data[11] = String(GPS.longitudeDegrees, 5);
  data[12] = "Voltage";
  data[13] = String(calibratedVoltage);
}

uint32_t timer = millis();
void loop()                     // run over and over again
{
  if (! usingInterrupt)
  {
    char c = GPS.read();
    if (GPSECHO)
      if (c) Serial.print(c);
  }
  if (GPS.newNMEAreceived())
  {
   if (!GPS.parse(GPS.lastNMEA()))
      return;
  }
  if (timer > millis())  timer = millis();

  if (millis() - timer > 100)
  {
    timer = millis(); // reset the timer
    setToByteArray(); // Set data to key
    sendDataOverSerial(data); // Send data over serial.
  }

  if (GPS.latitudeDegrees > 0)
  {
    // take a number of analog samples and add them up
    while (sample_count < NUM_SAMPLES) {
      sum += analogRead(A0);
      sample_count++;
      delay(10);
    }
    // calculate the voltage
    // use 5.0 for a 5.0V ADC reference voltage
    // 5.015V is the calibrated reference voltage
    voltage = ((float)sum / (float)NUM_SAMPLES * 4.6) / 1024.0;
```

```
    calibratedVoltage = voltage * (11.53);
    // send voltage for display on Serial Monitor
    // voltage multiplied by 11 when using voltage divider that
    // divides by 11. 11.132 is the calibrated voltage divide
    // value
    // Serial.print(voltage * 11.132);
    // Serial.println (" V");
    sample_count = 0;
    sum = 0;
    if (voltage < 25)
    {
      voltage == 0;
    }
  }
}
```

```
// EchoSounderROV.ino

////////////////////////////////////////////////////////////////////////////////
// NMEA PARESER FOR ROV ONBOARD ECHO SOUNDER
// This appliocations parses the NMEA0183 data for the ROV
// onboard echo sounder and sends the data to the RBPi via
// I2C bus.
#include <nmea.h> // NMEA libarary
#include <SoftwareSerial.h> // Software serial library
//#include <Wire.h> //I2C Library
#include <stdlib.h>
#define arduinoAddress 11 // I2C Address
// Connect Vin (5V) and Gnd
// GPS TX --> Digital 11
// GPS RX --> Digital 10

//--------------------
//Briefcase uno  5 RX pin, 6 TX pin
//ROV nano 2 RX pin, 3 TX pin
//--------------------
SoftwareSerial nmeaSerial(5, 6); // 5 RX pin, 6 TX pin, if
// is unreadable try switching.
NMEA nmeaDecoder(ALL);
// Set true if debugging
static boolean debug = false;
// Variables for NMEA data
float depthBelowTd = 0.00;
float depth = 0.00;
float offsetFromTd = 0.00;
float speedKnots = 0.00;
float speedKm = 0.00;
float temp1 = 0.00;
// Variables for splitting NMEA sentence
char* t0;
char* t1;
char* t2;
char* t3;
char* t4;
char* t5;
char* t6;
char* t7;
char* t8;
char* t9;
boolean LED = false;


void setup() {
  // Serial setup
```

```
  Serial.begin(4800);
  nmeaSerial.begin(4800);
  Serial.println("<EchoSounder:0>"); // Sends a setup string to Java.
  pinMode(LED_BUILTIN, OUTPUT);

  //
------------------------------------------------------------------------------ -
  // I2C setup
  // join I2C bus (I2Cdev library doesn't do this automatically)
  // Wire.begin(arduinoAddress);
  // listen on the I2C bus for incoming messages defined by the address
  // if called with data to process go to recieve
  // Wire.onReceive(receiveData); //register events (and name methods)
  // If called to send data send data to the master calling for it.
  // Wire.onRequest(sendData);
  //
------------------------------------------------------------------------------ -
  delay(1000);
  Serial.println("Setup done");
}


// Main loop, collecting data, parsing and sending to RBPi or Surface vessel
computer
void loop() {
  if (LED)
  {
   digitalWrite(LED_BUILTIN, HIGH);
    LED = false;
  }
  if (LED)
  {
   digitalWrite(LED_BUILTIN, LOW);
    LED = true;
  }

  //Checks for avilable NMEA sentences
  if (nmeaSerial.available()) {
    if (nmeaDecoder.decode(nmeaSerial.read())) { // if we get a valid NMEA
sentence
      if (debug) {
       Serial.println(nmeaDecoder.sentence()); // Pritint NMEA sentence if
debuging
      }

      // Decoding the nmea sentence, splitting it up where
      // it is seperated by a comma.
      decodeNmeaSentence();
      // Paring the NMEA sentence, extracting the desiered data
```

```
      parseNmeaSentence();
      // Sending updated information to the RBPi.
      convertDataToString();
     }
   }
}
//--------------------------------------------------------------------------------
-
// NMEA decoder, Splitts the NMEA sentence where it is seperated by commas
void decodeNmeaSentence() {
  t0 = nmeaDecoder.term(0);
  t1 = nmeaDecoder.term(1);
  t2 = nmeaDecoder.term(2);
  t3 = nmeaDecoder.term(3);
  t4 = nmeaDecoder.term(4);
  t5 = nmeaDecoder.term(5);
  t6 = nmeaDecoder.term(6);
  t7 = nmeaDecoder.term(7);
  t8 = nmeaDecoder.term(8);
  t9 = nmeaDecoder.term(9);
}
//--------------------------------------------------------------------------------
-
// Parses the NMEA sentences and extracts the desiered data
void parseNmeaSentence()
{
  //if t0 is Depth Below Transducer
  if (strcmp(t0, "SDDBT") == 0)
  {
    depthBelowTd = atof(t3); //Extracts field #3 which is depth below transducer.
    if (debug)
    {
      Serial.print("Depth below Transducer: ");
      Serial.println(depthBelowTd);
      Serial.println("");
      Serial.println("");
    }
  }
  //if t0 is Depth of Water
  if (strcmp(t0, "SDDPT") == 0)
  {
    depth = atof(t1); //Extracts field #1 which is depth of water.
    offsetFromTd = atof(t3); //Extracts field #3 which is transducer offset.
    if (debug)
    {
      Serial.print("Depth: ");
      Serial.println(depth);
      Serial.print("Transducer offset: ");
      Serial.println(offsetFromTd);
```

```
    }
  }
}

// For converting the parsen NMEA values to string
void convertDataToString()
{
  char str_depth[6]; // Depth
  char str_depthBelowTd[6]; //Depth Below Transducer

  for (int i = 0; i < 6; i++)
  {
    dtostrf(depth, 4, 2, str_depth);

  }

  for (int i = 0; i < 6; i++)
  {
    dtostrf(depthBelowTd, 4, 2, str_depthBelowTd);
  }


  // Start char, separation char and end char with key
  char* a = "<D:";
  char* b = ":DBT:";
  char* k = ">";


  // Creating one big string of <Key:Value> format
  char bigstring[32] = "";  // enough room for all strings together
  strcat(bigstring, a);   // add first string
 strcat(bigstring, str_depth);
 strcat(bigstring, b);
 strcat(bigstring, str_depthBelowTd);
 strcat(bigstring, k);


  // Print the final big string
 Serial.println(bigstring);
 //Serial.write(bigstring); // For serial communication
 // Wire.write(bigstring); // For I2C communication
}
```