NTNU

Kunnskap for en bedre verden

Towed-ROV

BACHELOR THESIS

Kristian Salvesen Homdrom (IIR: 10004)
Marius Nonsvik (IIR: 10025)
Daniel Rasmus Reite (IHB: 10004)
Morten Lerstad Solli (IIR: 10012)

Total number of pages: 140/478

Delivered: 31.05.2018, Ålesund

IE303612 Department of ICT and Natural Sciences
IP305012 Department of Ocean Operations and Civil Engineering

Norwegian University of Science and Technology

Supervisor 1: Ottar L. Osen
Supervisor 2: 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 L. 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 en vederlagsfri rett til å gjøre oppgaven tilgjengelig for elektronisk publisering:	**☒ja**	**☐nei**
Er oppgaven båndlagt (konfidensiell)? (Båndleggingsavtale må fylles ut)	**☐ja**	**☒nei**
- Hvis ja: **Kan oppgaven publiseres når båndleggingsperioden er over?**	**☐ja**	**☐nei**
Er oppgaven unntatt offentlighet? (inneholder taushetsbelagt informasjon. Jfr. Offl. §13/Fvl. §13)	**☐ja**	**☒nei**

Dato: 31.05.2018

ABSTRACT

This thesis evaluates the concept of a "Towed-ROV", and is provided by NTNU Ålesund. The purpose of this project is to design and develop a remote operated vehicle, to be towed behind a vessel. It functions as a multi-purpose platform with the capabilities of performing exploration, searches and surveys. Creating such an ROV simplifies these tasks by providing real-time video and sensor data, removing the need of recharging batteries and the continuous attention from an operator. To achieve this, it was developed a design, autonomous depth controller, graphical user interface, server application and option to easily add external equipment on the ROV. A prototype was created to prove this concept. Tests performed revealed acceptable performance from the prototype, using affordable materials. These results validate the concept of a towed ROV as a suitable solution for the designated area of application.

Preface

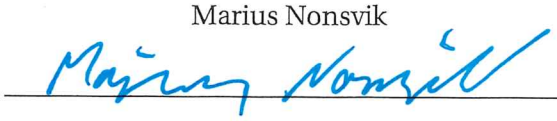
This bachelor thesis is written by four students at NTNU in Aalesund, three students from Automation and one from Product and Systems design. The project and thesis was carried out during the spring term of 2018, and its purpose being to investigate possible solutions for a towed-ROV, including; design, building of prototype for test of instruments, software applications and control system.

We would like to thank the following people:

- Our supervisors Ottar L. 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 André Tranvåg and the lab apprentices for assistance.
- Lastly, to anyone who has contributed with our project.

Kristian Salvesen Homdrom

Marius Nonsvik



Daniel Rasmus Reite

Morten Lerstad Solli

Ålesund, 31.05.2018

Contents

ABSTRACT i
Preface ii
List of figures vii
List of tables xi
Terminology xiii
Concepts xiii
Abbreviations xiii

1 Preliminaries **1**
1.1 Introduction 2
1.2 Problem Formulation 2
1.3 Objectives 2
1.4 Approach 3
1.5 Outline 3

2 Theoretical background **5**
2.1 Remotely Operated Underwater Vehicle 5
2.2 Material 5
2.3 Hardware 6
2.3.1 Microcontroller 6
2.3.2 Actuator 7
2.3.3 MOSFET Transistor 7
2.3.4 ROV umbilical cable 8
2.3.5 IMU 9
2.3.6 Magnetometer 11
2.3.7 Echo Sounder 11
2.4 Software 12
2.4.1 Concurrency 12
2.4.2 Threads 13
2.4.3 ThreadPool 13

2.4.4 Executor & Scheduler . . . 13
2.5 Control system . . . 14
2.5.1 PID . . . 14
2.5.2 MPC . . . 15
2.5.3 State-Space . . . 15
2.5.4 S-plane . . . 16
2.5.5 Ziegler–Nichols Closed Loop Method . . . 16
2.5.6 Fuzzy logic . . . 17
2.6 Communication . . . 18
2.6.1 I2C . . . 18
2.6.2 Power-Line Communication . . . 20
2.6.3 SSH . . . 20
2.6.4 Serial communication . . . 21
2.6.5 NMEA0183 . . . 21
2.6.6 UDP . . . 22
2.6.7 TCP . . . 22
2.7 Calculations . . . 22
2.7.1 Buoyancy . . . 22
2.7.2 Lift coefficient . . . 23
2.7.3 Drag coefficient . . . 23
2.7.4 Moment coefficient . . . 23
2.7.5 Pressure . . . 24
2.7.6 Reynolds number . . . 24
2.7.7 Added mass derivatives for a Prolate ellipsoid . . . 24
2.7.8 Complementary Filter . . . 26
2.7.9 Depth Calculations . . . 27
2.7.10 DC-cable resistance . . . 27
2.7.11 Voltage Drop . . . 28
2.7.12 Cable Calculations . . . 28
2.8 Version control . . . 28
2.8.1 Git . . . 29

3 MATERIAL AND METHOD **30**
3.1 Data . . . 30
3.1.1 NMEA 0183 . . . 30
3.2 Project management . . . 31
3.3 Utilised Software . . . 31
3.4 Concept Studies . . . 32

3.4.1 Frame . . . 32
3.4.2 Material Selection . . . 33
3.4.3 NACA profiles . . . 34
3.4.4 Rotation of Airfoils . . . 35
3.4.5 Housing . . . 36
3.4.6 Placement of wings . . . 36
3.4.7 Power management . . . 36
3.4.8 Control of Actuators . . . 37
3.5 Method . . . 38
3.5.1 Modelling . . . 38
3.5.2 Control System . . . 40
3.5.3 Software development . . . 41
3.5.4 Ethernet Communication . . . 46
3.5.5 Internal communication . . . 46
3.5.6 Calculations . . . 48
3.5.7 FEM-Analysis . . . 51
3.6 Material . . . 52
3.6.1 Raspberry Pi 3 model B . . . 54
3.6.2 Arduino Mega . . . 55
3.6.3 Arduino Pro Mini . . . 55
3.6.4 Raspberry pi fisheye camera . . . 55
3.6.5 Fathom-X Tether Interface . . . 56
3.6.6 Umbilical neutral tether ROV cable . . . 56
3.7 Test Setup . . . 57
3.7.1 Systems Tests . . . 58
3.7.2 Water test . . . 60

4 RESULT **61**
4.1 Prototype . . . 62
4.1.1 Frame . . . 62
4.1.2 Wings/ profiles . . . 63
4.1.3 Front cover . . . 64
4.1.4 Wing control . . . 65
4.1.5 Camera Housing . . . 67
4.1.6 Trim spoiler . . . 68
4.1.7 Boxes and sealing . . . 69
4.1.8 Mounting of cable . . . 72
4.1.9 Instruments . . . 73

4.1.10 Instrument disposition . . . 76
4.1.11 Microcontrollers . . . 78
4.1.12 Onboard Computer . . . 79
4.1.13 Cable fitting . . . 79
4.1.14 Power management . . . 80
4.1.15 Motor Controllers . . . 82
4.1.16 Ethernet . . . 84
4.1.17 Mounting of Equipment in Prototype . . . 84
4.2 Surface vessel . . . 86
4.2.1 Instruments . . . 86
4.2.2 Surface Computer . . . 87
4.2.3 Battery . . . 87
4.2.4 Mounting Echo Sounder to boat . . . 87
4.3 Control System . . . 88
4.3.1 Modelling . . . 88
4.3.2 PID . . . 91
4.3.3 Fuzzy logic . . . 91
4.4 Software design/solution . . . 93
4.4.1 Dataflow diagram /dataflyt . . . 93
4.4.2 I2C and Serial Communication . . . 94
4.4.3 Client-Server Communication . . . 96
4.4.4 GUI-Graphical user interface . . . 98
4.4.5 Server Application . . . 102
4.4.6 Sensor data treatment . . . 106
4.4.7 start-up scripts . . . 108
4.5 Test Results . . . 108
4.5.1 Testing in Tank . . . 109
4.5.2 Dry Test . . . 109
4.5.3 Ocean Test . . . 110
4.5.4 Component tests . . . 112

5 DISCUSSION **114**
5.1 Prototype . . . 114
5.1.1 Design and materials . . . 114
5.1.2 Hardware . . . 115
5.1.3 Depth control system . . . 116
5.1.4 Software solution . . . 117
5.1.5 Test . . . 118

5.1.6 Possible areas for utilisation . . . 119
5.1.7 Prototype to finished product . . . 120
5.1.8 Challenges with a towed-ROV . . . 120
5.1.9 Project Execution . . . 120

6 CONCLUSION **122**
6.1 Further Development . . . 123

Bibliography **124**

Appendices **128**

A Pre-project **129**

B Gannt diagram **154**

C Towed-ROV User Manual **157**

D Meetings with Supervisors **160**

E Wiring Diagrams **173**

F Mechanical Drawings **177**

G Airfoil Calculations **205**

H Class Diagram **207**
H.1 Server Application Class Diagram . . . 207
H.2 Client Application Class Diagram . . . 209

I Code **211**
I.1 Arduino Code . . . 211
I.2 Python Code . . . 238
I.3 Java Server Application Code . . . 241
I.4 Java Client Application Code . . . 325

List of Figures

1.1 Concept . . . 1
1.2 Prototype . . . 2

2.1 MOSFET . . . 8
2.2 Umbilical cable . . . 8
2.3 Magnitude vectors, x,y,z axis . . . 9
2.4 Accelerometer . . . 9
2.5 Gyroscope . . . 10
2.6 Magnetometer . . . 11
2.7 Echo sounder Principe . . . 12
2.8 PID Controller . . . 14
2.9 Discrete time MPC . . . 15
2.10 Degree of membership . . . 17
2.11 Sugeno-style inference . . . 18
2.12 I^2C bus . . . 19
2.13 I^2C I2C master-slave . . . 19
2.14 I^2C bus . . . 20
2.15 SSH . . . 21
2.16 git Snapshot . . . 29

3.1 Different ROVs . . . 33
3.2 different lift forces . . . 34
3.3 different lift forces . . . 35
3.4 Linear Actuator . . . 35
3.5 Rotary actuator . . . 35
3.6 H-bridge . . . 38
3.7 Software Sketch . . . 43
3.8 Producer consumer pattern . . . 43
3.9 I2C register . . . 47
3.10 I2C init . . . 48

3.11 Cl v alpha . . . 49
3.12 Cl v alpha . . . 50
3.13 Estimating distance to ROV . . . 50
3.14 Depth/pressure relationship . . . 51
3.15 Sample of a mesh . . . 51
3.16 Raspberry Pi . . . 54
3.17 Arduino Mega . . . 55
3.18 Raspberry Pi Camera . . . 56
3.19 Fathom-X-3 tether interface . . . 56
3.20 Cable . . . 57
3.21 Echo Sounder Test . . . 59

4.1 Finished prototype . . . 61
4.2 Final design of prototype . . . 62
4.3 The main frame of the ROV . . . 63
4.4 Lift force per module at 6 knots . . . 64
4.5 Demonstration of wing assembly . . . 64
4.6 Drag coefficient of isosceles triangle . . . 65
4.7 Joint connecting the foil to the actuator . . . 66
4.8 3D printed camera housing . . . 67
4.9 Camera housing . . . 68
4.10 Camera house nuts . . . 68
4.11 Demonstration of camera house assembly . . . 68
4.12 Trim spoiler . . . 69
4.13 Electronicsbox oil filled . . . 70
4.14 Actuatorbox oil filled . . . 70
4.15 FEM analysis of actuator box wall . . . 70
4.16 FEM analysis of echo sounder box wall . . . 71
4.17 FEM analysis of electronic box wall . . . 71
4.18 Echo sounder box . . . 72
4.19 Mounting of cable . . . 73
4.20 Adafruit breakout board 9DOF . . . 73
4.21 30Bar High-Resolution Pressure Sensor . . . 74
4.22 Airmar-DST800 . . . 75
4.23 ES Arduino and optoisolator . . . 75
4.24 SOS Leak Sensor . . . 75
4.25 Fish eye camera . . . 76
4.26 Camera houseing front . . . 77

4.27 Camera housing . . . 77
4.28 Electrcial box . . . 77
4.29 Echo Sounder box . . . 78
4.30 Raspberry Pi-Model 3 B + . . . 79
4.31 PG Cable gland . . . 80
4.32 Actuator Boxes . . . 80
4.33 MOSFET H-bridge . . . 82
4.34 L298 Motor driver . . . 83
4.35 Jrk 21 motor driver . . . 83
4.36 Voltage spike . . . 84
4.37 Equipment in boxes . . . 85
4.38 Boxes filled with oil . . . 85
4.39 Boxes sealed shut . . . 85
4.40 Boxes with equipment placed in ROV . . . 85
4.41 Adafruit Ultimate GPS . . . 86
4.42 Mounting echo sounder to boat . . . 88
4.43 Simulink model . . . 90
4.44 Simulink simulation . . . 91
4.45 Input classification Fuzzy . . . 92
4.46 Fuzzy logic Sugeno . . . 92
4.47 DataFlow . . . 93
4.48 Raspberry GPIO . . . 95
4.49 Union Arduino . . . 95
4.50 Depth sensor solution diagram . . . 96
4.51 TCP and UDP communication . . . 97
4.52 Towed ROV GUI . . . 99
4.53 Towed ROV echo sounder . . . 99
4.54 I/O Byte . . . 100
4.55 Towed ROV I/O . . . 100
4.56 Towed ROV Data log example . . . 101
4.57 Towed ROV Options frame . . . 101
4.58 Client connection:Creating Clienthandler . . . 102
4.59 Parsing of data from client . . . 103
4.60 StorageBox:Producer consumer principle . . . 103
4.61 Fuzzy class overview . . . 104
4.62 GPIO Overview with Observer . . . 104
4.63 I2C class overview . . . 105

4.64 Depth test data 111
4.65 The ROV behind the boat 111
4.66 ROV camera capturing propeller 111
4.67 GPS Test 112
4.68 Echo sounder Test Results 113

List of Tables

2.1 Material 6
2.2 Microcontroller comparison 7
2.3 Programming languages 12
2.4 Ziegler–Nichols method 17

3.1 Utilised Programs 32
3.2 Calculated max consumption 37
3.3 Voltage in cable 37
3.4 Degrees of freedom 39
3.5 Java Libraries 45
3.6 Python Libraries 45
3.7 Arduino Libraries 45
3.8 Part and price list 52

4.1 effective consumption 12V 81
4.2 Voltage in cable 81

Terminology

Concepts

Oscillator Et feedback system som bevarer energi med resonans

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

Concurrent Concurrency is when several tasks will run simuntaniously, by running them in overlaping time periodes

Ah Ampere houres

Opto-isolator A device used to optically isolate circuits

Isosceles triangle A triangle which has at least to equal angles

Abbreviations

ROV Remote Operated Vehicle

AUV Autonomous Underwater Vehicle

GPS Global Positioning System

PID Proportional integral derivative

MPC Model predictive control

SISO Single input single output

MIMO Multiple input multiple output

TCP Transmission Control Protocol

UDP User Datagram Protocol

GPS Global Positioning System

PCB Printed Circut Board

FFC Flat Flex Cable

DOF Degrees of freedom

GPIO General-purpose input/output

NMEA	National Marine Electronics Association
IMU	Interial measurements unit
I2C	Inter-Integrated Circuit
JVM	Java viritual machine
DCS	Depth Control System
SBC	Single-Board Computer
OS	Operating System
MOSFET	Metal-oxide-semiconductor field-effect transistor
FTA	Fault tree analysis
MTTF	Mean time to failure
RAMS	Reliability, availability, maintainability, and safety
PLM	Product lifecycle management
IDE	Integrated development environment
NACA	National advisory commitee for aeronautics
CAD	Computer-aided design
ASCII	American standard code for information interchange
SSH	Secure shell protocol
TP	Twisted-pair
IC	Integrated Circuit
PLC	Power-Line Communication
IMCA	International Marine Contractors Association
MEMS	Microelectromechanical systems

Chapter 1

Preliminaries

Over 70 percent of the world is covered by oceans, and yet the underwater domain is less explored than the surface of the moon. Many companies have in the later years shown a growing interest in this domain, and it will be in the years to come a target for future exploration and resource extraction. Exploring the seabed is extremely difficult regarding the immense depths and harsh conditions an underwater craft has to endure. Therefore, a towed ROV (remote operated vehicle) is a solution to explore strange new depths, to seek out new life, to boldly go where no one has gone before. In addition to not be dependent on resurfacing, charging batteries or unload data provides an advantage for the towed ROV. Another edge is the retrieval of sensor data and video in real time, something that is impossible to do with a cable-free AUV. This is a field with big potentials and the rapidly growing interest will lead to major advances in technology in the future.

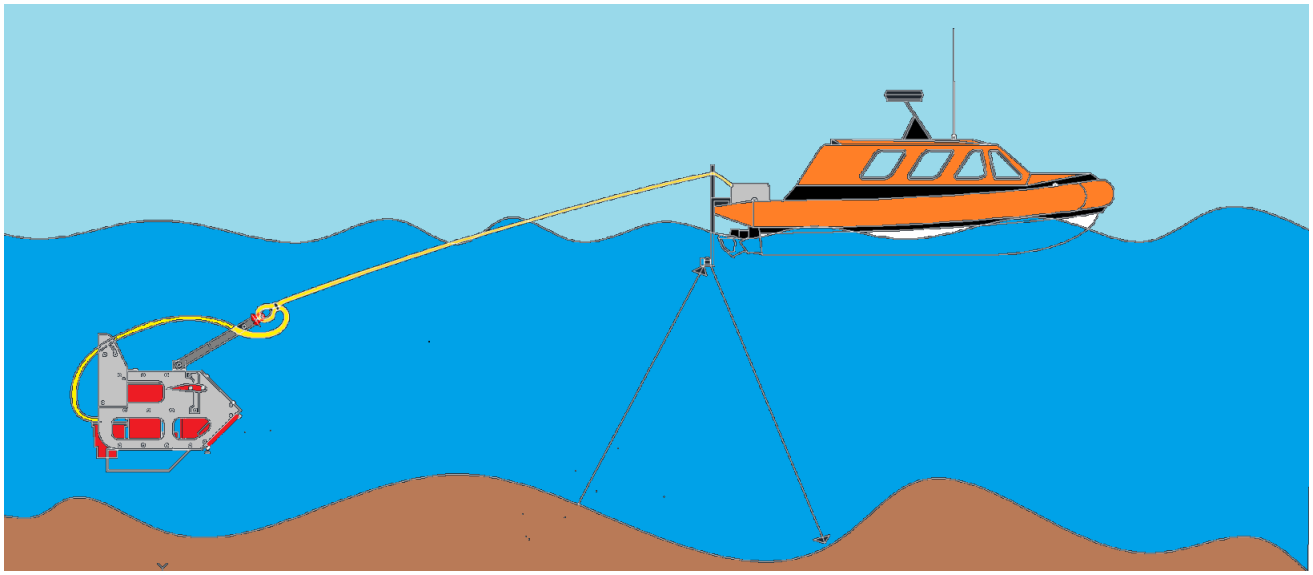


Figure 1.1: Concept

1.1 Introduction

NTNU in Ålesund aspire to develop a functional ROV which can perform countless subsea tasks; exploration, survey, mapping, search etc. The ROV will be towed behind a vessel on the surface (illustrated in figure 1.1, removing the need for self propulsion. The ROV will have automatic depth control, either from surface to target depth or ROV to seafloor. Areas of focus will be to design, build and develop software solution for the ROV. Consideration has been on flexibility, manufacturing and low cost. Additionally there will be built a prototype (Phorcys) shown in figure 1.2 for testing of solutions according to object specifications. The thesis is provided by NTNU in Ålesund.

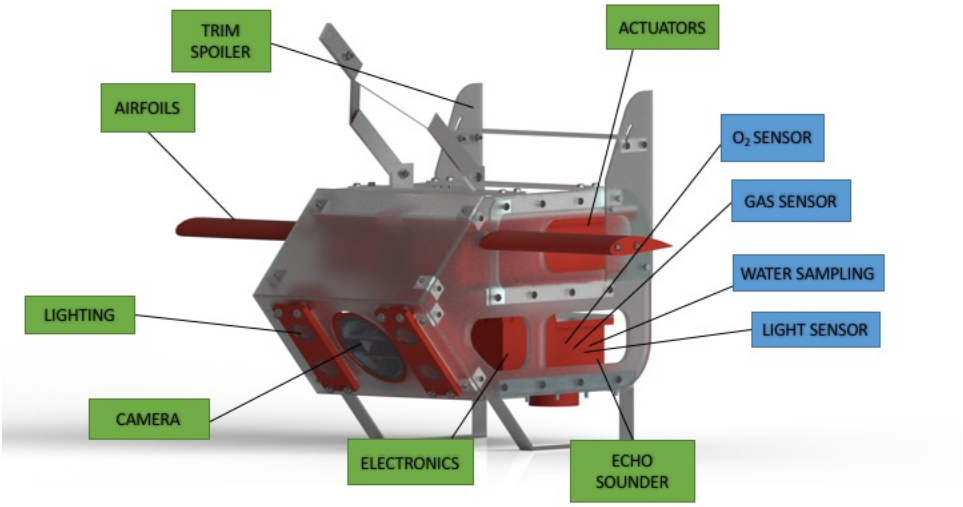


Figure 1.2: Prototype

1.2 Problem Formulation

The problem for this project can be split into several parts:

- Designing a concept for a prototype using materials and methods within the budget and time.
- Building prototype for test of automatic depth control, software, logging of data, instrumentation and sensor pay-load.

1.3 Objectives

The prototype developed during this project should meet the following criteria;

1. Simple assembly, hydrodynamic and modular design.
2. Automatic control depth
3. Software design for connecting new sensors, record and transfer data and video.
4. Clean and functional user interface.

1.4 Approach

During the scope of the thesis, the project were divided in different phases. The first phase dealt with project management and concluded with a pre-project report, which included a Gannt diagram with tasks and deadlines.

The next phase were researching other solutions and gathering of data along with ordering of parts. During the research phase many design forms and needed sensors were considered and studied.

After the research was completed, another phase started which were designing, code layout and control-simulation. This cooperated with the next phase that were hardware testing of equipment and communication transfer.

This led over to a stage of client and server coding phase were data were to be extracted between components, data processing and manufacturing of parts.

Finally were the testing of the full design and writing of thesis.

1.5 Outline

This thesis contains the following chapters:

- Chapter 2. Theoretical background: Theoretical background needed in this report, with proper references to each literature reference used. Theory may involve concepts, definitions, methods, regulations/key standards, theory to explain specific system behavior, and so on.
- Chapter 3. Material and Method: contains a description of the planning and project approach. It also contains important information of what needs to be considered for building a towed ROV. In the end of the chapter there is a description of how the tests will be completed.

- Chapter 4. Result: In this chapter there is a description of the developed prototype, the material for building it, the software developed for the project and the depth control system. In addition to this it contains results from the tests which were done.
- Chapter 5. Discussion: An assessment of the chosen methods and achieved results. As well as limitations, changes done and sources of error.
- Chapter 6. Conclusion: Presenting the results with the experiences gathered troughout the project.

Chapter 2

Theoretical background

The theory needed for the project is presented in this chapter. It is split into areas of use such as; Material, hardware, software, control system, communication protocols, and equations and formulas needed.

2.1 Remotely Operated Underwater Vehicle

A Remotely operated underwater vehicle is an unmanned mobile device used below the surface of the ocean. They are often operated by a team of people on board a ship which the ROV is deployed from. There are several types of ROV's, such as towed ROV's and self propelled. Common for all types is that they have a cable connection running from the ROV to a surface vessel. (Society 2018)

2.2 Material

Several materials must be be considered for this project. The considered materials are shown in table 2.1.

Table 2.1: Material

Materials	About
PLA	PLA is a material used for 3D printing. It has a tensile strength of 37 MPa, which makes it suitable for testing and prototyping.
ABS	ABS is a thermoplastic which is trong and has a high impact resistance.
Plexiglass	Plexiglass (acrylic glass) is a transparent thermoplastic material. It is often used as an alternative to glass,as it is stronger.
Composite	Composite material is made by mixing two components. There are many types of different composites, with different physical properties
Aluminium	Aluminium is a light material with high strength-to-weight ratio. It corrodes in water over long periods of time.
Stainless steel	Stainless steel is a iron alloy with at least 10.5% chromium. It is corrosion resistant and strong compared to aluminium.

2.3 Hardware

In this section the concepts and data about specialised hardware is presented.

2.3.1 Microcontroller

Microcontrollers are used to control inputs and outputs. The Arduino is programmed using the ArduinoC language, which is based on C. The Teensy microcontroller can be programmed both by C and ArduinoC, to use the Arduino IDE an add-on for Teensy must be installed. The microcontrollers considered are presented in table 2.2.

Table 2.2: Microcontroller comparison

Microcontroller	I/O	Clock speed	Size	Timers
Arduino Mega	Digital: 54 Analogue: 16	16MHz	101.2mm x 53.3mm	1
Arduino Uno	Digital: 14 Analogue: 6	16MHz	68.6mm x 53.4mm	4
Arduino Pro Mini	Digital: 14 Analogue: 8	16MHz	18mm x 32mm	1
Teensy 3.5	Digital: 37 Analogue: 25	120MHz	62,3mm x 18mm	12

(Arduino.cc 2018a; Arduino.cc 2018d; Arduino.cc 2018b; Arduino.cc 2018c)

2.3.2 Actuator

An actuator is a component responsible for motion or control in a mechanical system. By sending a control signal, the actuator converts energy from an energy source to a mechanical motion. The most common actuators in industry today are controlled with pneumatic pressure, hydraulic pressure or electrical voltage or current. (Company. 2018)

2.3.3 MOSFET Transistor

The basic principle of the transistor were first patented in 1925 by Julius Edgar Lilienfield. The first MOSFET transistor were not created before 1959, metal-oxide-semiconductor field-effect transistor is a type of field-effect transistor, it has three pins: Gate, Drain and Source, where the voltage on the gate pin determines how much conductivity there is between source and drain pins. (ASPENCORE 2018)

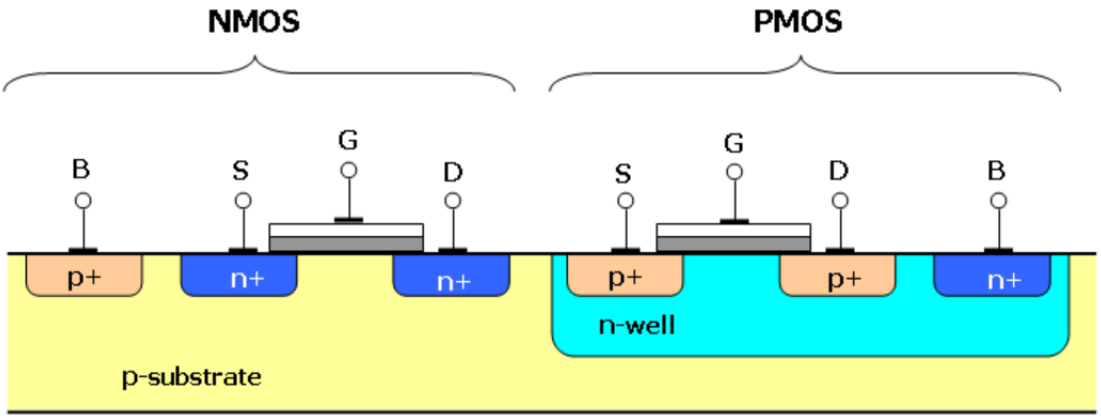


Figure 2.1: MOSFET
Source:elprocus.com

2.3.4 ROV umbilical cable

Tether cables are light weight and robust cables that are installed between a support vessel and a ROV, as a mean to transmit power, signals and mechanical payload (*ROV umbilical and tether* NA). The umbilical cable is armoured and contains a group of electrical conductors . These conductors are used for carrying electrical power, signals and video between the ROV and the surface vessel (*Guidance for diving supervisors IMCA D 022* 2016-08-1).

The tether is exposed to low diameter bending, wire stress and snap loads. The tether is reinforced with kevlar for withstanding the forces it faces. Umbilical cables are either neutral buoyant or heavy in the water, in addition to being waterproof to manufacturer's specifications.

Figure 2.2: Umbilical cable
Source: Nauticexpo.com

2.3.5 IMU

Inertial measurement unit (IMU), is a unit that measures and report force, angular rate and magnetic field around an object. An IMU may be a composition of accelerometers, gyroscope and a magnetometer. The magnitude of vectors are used in the xyz-plane in order to calculate angles.

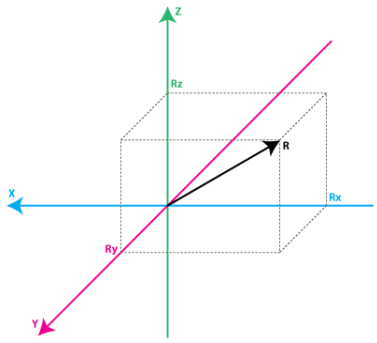


Figure 2.3: Magnitude vectors, x,y,z axis
Source: Starlino.com

Accelerometer

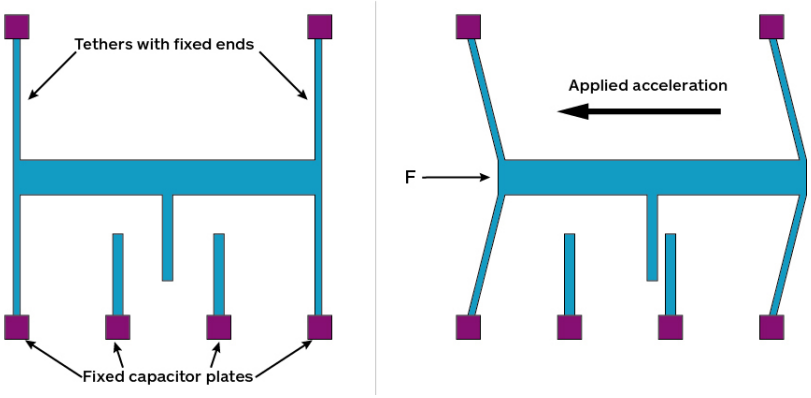


Figure 2.4: Accelerometer
Source: Globalspec.com

The accelerometer works in a way that it uses fixed capacitors and tethers with fixed ends. When a force is applied, the tether will move and a current between capacitors will change.
To achieve accurate readings of the accelerometer all three axes will be used, calculating the magnitude between the two other axes. In other words calculate the angle between the vector and the axis that is divided by the vector.

$$\theta = \arctan\left(\frac{x}{\sqrt{z^2 + y^2}}\right) \tag{2.1}$$

Gyroscope

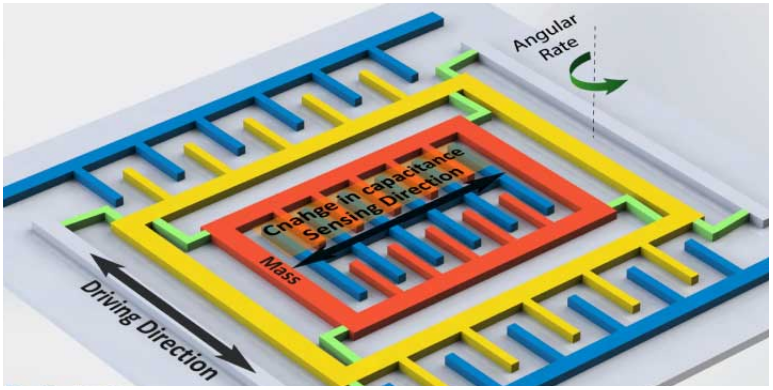


Figure 2.5: Gyroscope
Source: Howtomechatronics.com

When a force is applied to a gyro at tilt angle, a displacement of mass will occur. This displacement can be measured with change in capacitance as with the accelerometer.
Gyroscopes are used for maintaining or measuring orientation, a MEMS gyroscope measures angular velocity of which angle can be calculated.
As distance is an integration of velocity:

$$S_n = S_{n-1} + v \cdot t \tag{2.2}$$

The idea is to take measurements from the gyroscope at constant intervals and estimate distance. However as we can not take infinite readings it will only be an estimation and will therefore drift overtime.A gyroscope also has limited refresh rate, and if set to a update rate of 100Hz it will only provide a new reading every 10ms. If readings are applied faster, the results might be old or corrupt readings.

2.3.6 Magnetometer

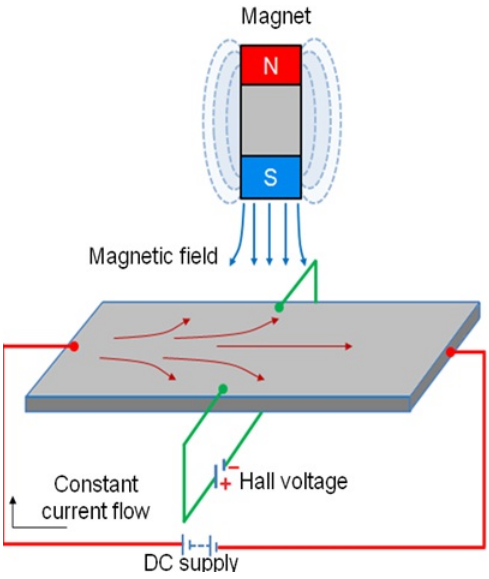


Figure 2.6: Magnetometer
Source: Digilentinc.com

The magnetometer measures the magnetic field of low resitive flat area, sending a constant current to the area. When a magnetic field affect the area the electrons will move further away from the center and this change can be measured. (Nedelkovski 2018)

2.3.7 Echo Sounder

An echo sounder is a device which is used to determining the depth of water by sending sound pulses towards the bottom of the ocean and then receives the echo. The time interval between the transmitted signal until the echo returns is then used to calculate the depth. (*Sound in water* n.d.)

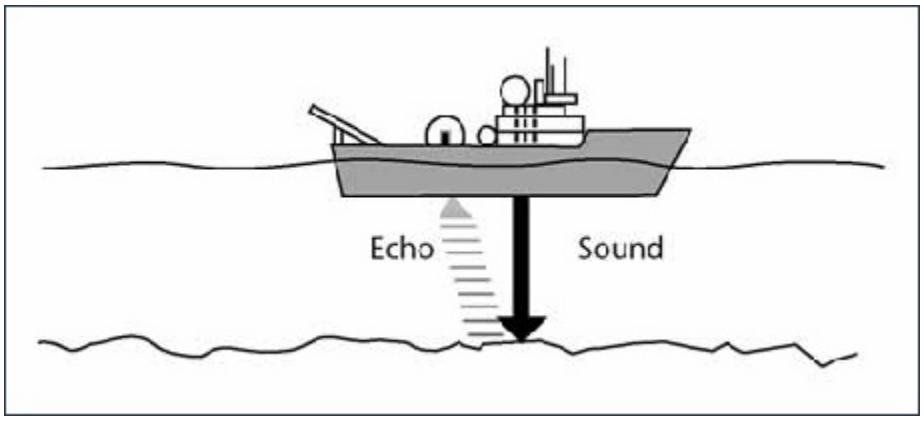


Figure 2.7: Echo sounder Principe
Source: Oicinc.com

2.4 Software

This section present the software development theories used to develop the software and concepts that had to be taken into concern. Table 2.3 shows the the programming languages used in the project.

Table 2.3: Programming languages

Programming languages	**About**
Java	Java is a general-purpose computer programming language that is concurrent, class-based, object-oriented and specifically designed to have as few implementations as possible.
Python	Python is an open source multipurpose programming language.
Arduino C	Arduino C is a sequential programming language for the Arduino microcontroller which is built on C

2.4.1 Concurrency

The principle behind concurrent programing is to run several tasks of a program at almost the same time. Each task will be assigned a thread, where the scheduler or executor decides which thread will run at any given time. If we have a single cored processor, the executor will switch between threads either by time or events. By using concurrency in our program, we can optimise the use of the CPU by fully utilising the speed and cores of the processor. (Gonzalez 2016; Wellings 2004)

2.4.2 Threads

Threads are used to improve performance of complex applications, they improve responsiveness and utilisation of a program. By using threads, we can access more than one processor at once, and with properly design of the program, the threads can better utilise the processing power. (Gonzalez 2016)

There are two ways to make a thread, the first is to extend the Thread class, and the second is to implement the Runnable interface. For more details about the Thread and Runnable class see Wellings 2004.

2.4.3 ThreadPool

The ThreadPool consists of a collection of all threads that are created when the application starts. It can be defined in the ThreadPool how many threads it should contain. It is important to keep in mind that the more threads in the ThreadPool, the more resources are used. When executing a runnable task, it is given a thread from the pool to execute the task. When the task is done, the thread will return to the pool for another task to use it. If there are no threads available the task will have to wait for another task to complete. (Gonzalez 2016)

2.4.4 Executor & Scheduler

Executor forms the basis for a flexible and powerful framework for asynchronous task execution, and are used for managing threads within an application. The executor is based on the producer-consumer principle, where producers submit tasks and consumers execute them.

Whenever Threads are in use one should consider replacing the standard single thread operation with the executor framework. This makes the multithreaded program more durable as the executor offers protection for:
(Gonzalez 2016)

1. Unforeseen events that make threads stop, executor starts a new thread if one dies unexpected.
2. Resource exhaustion, limits the number of active threads.
3. Prioritising thread to get run-time.
4. Actions before and after task execution.

Threads managed by the executor are thread-safe as long as the methods of the tasks are implemented correctly. Tasks can be scheduled to run one time or periodically using a scheduler. (Gonzalez 2016)

2.5 Control system

This section present the theory used to make and tune the control system for the ROV.

2.5.1 PID

A PID controller is the leading control method in the industry. A PID controller calculates the error between a setpoint and a feedback often containing a measured variable. The controller applies a correction output based on proportional, integral and derivative terms like shown in figure 2.8. The proportional term applies an output proportional to the current error value. The integral term provides an output based on the magnitude and duration of the error, or the integral of the error. Accelerating the process and eliminates steady state error, but can cause the value to overshoot. The derivative term applies an output based on how rapidly the error is currently changing. Additionally, the term predicts the system behaviour to reduce settling time and increase the system stability. The output of a PID controller is given by:
(Bye 2017)

$$u(t) = K_p \cdot e(t) + K_i \cdot \int_0^t e(t)\mathrm{d}t + K_d \cdot \frac{de}{dt} \tag{2.3}$$

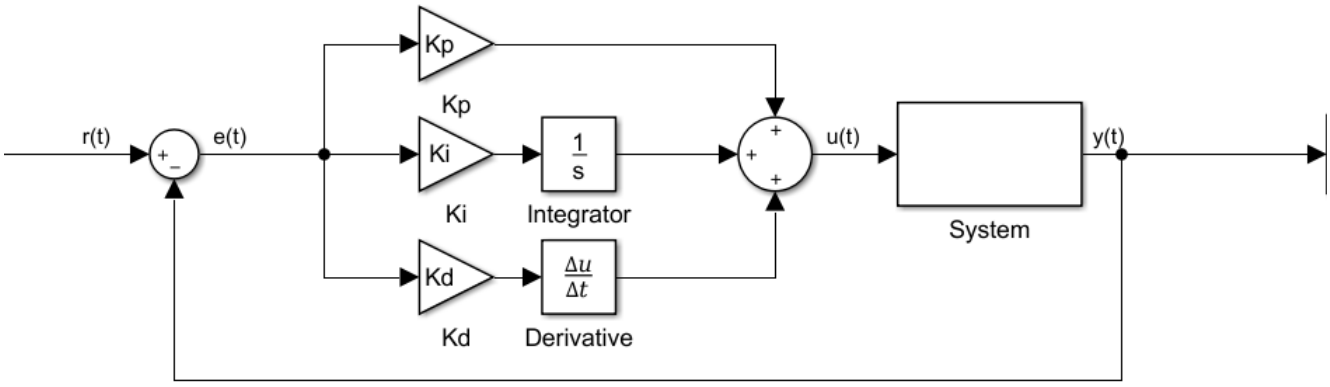


Figure 2.8: PID Controller

2.5.2 MPC

Model predictive control is an advanced controller often used in slow industrial processes. The basic principle of an MPC is to optimise output by predicting future events like illustrated in figure 2.9. The MPC predicts events within a predetermined time-horizon. This makes it possible to estimate future values and anticipate events.

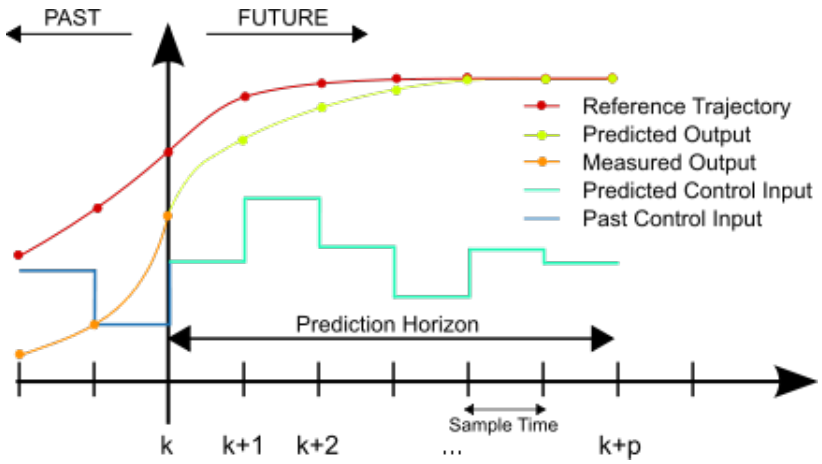


Figure 2.9: Discrete time MPC
Source: Robin T. Bye

(Bye 2017)

2.5.3 State-Space

State space representation is used in modern control system engineering to model a system in time domain. This easily implemented approach can be used to model nonlinear, time-varying systems with nonzero initial conditions. State space representation is also used to easily model MIMO systems. The state space representation of a continuous-time system is given by two equations:

State Equation:

$$\dot{x} = Ax + Bu \tag{2.4}$$

Output Equation:

$$y = Cx + Du \tag{2.5}$$

(Bye 2017)

2.5.4 S-plane

A system can be represented graphically in the complex S-plane while modelling in frequency domain. By using the Laplace transform to convert from time domain to s-plane makes it possible to use basic algebra to model the system. The s-plane is frequently used in modelling and makes it easy to analyse stability of a system.

A function f in time domain can be transformed to the s-plane using Laplace transform. The Laplace transform of a function is defined by equation 2.6

$$\mathscr{L}[f(t)] = F(s) = \int_{0-}^{\infty} f(t)e^{-st}dt \tag{2.6}$$

The inverse Laplace transform, allows us to find f(t) given F(s). This is defined by equation 2.7

$$\mathscr{L}^{-1}[F(s)] = f(t)u(t) = \frac{1}{2\pi j}\int_{\sigma-j\infty}^{\sigma+j\infty} F(s)e^{st}ds \tag{2.7}$$

where

$$u(t) = 1 \quad t > 0$$
$$u(t) = 0 \quad t < 0$$

(Nise 2011)

2.5.5 Ziegler–Nichols Closed Loop Method

The Ziegler-Nichols closed loop method (also known as Ziegler-Nichols 2nd. method) is a method for tuning a PID controller, with the use of trial and error experimenting. This method designs a fast reacting system with acceptable stability, where acceptable stability is defined as; "The ratio of decline to amplitudes subsequent to the first after a step input should be $\frac{1}{4}$." Which means that the second amplitude should be one fourth of the first. The criteria for using Ziegler-Nichols is that the system have a time delay and an order higher than 3. (Haugen 2010)

Steps in Ziegler-Nichols method:

1. Bring the process to the operating point of the control system to get a real represantation of the system.
2. Set integral and derivative gain to zero by setting $T_i = \infty$ and $T_d = 0$.
3. Set controller in automatic mode.
4. Set $K_p = 0$ and increase it gradually until the system reaches constant oscillation.

5. When the system reaches constant oscillation the $K_{CU} = K_P$ and the P_U = the period time.
6. Set these values into table 2.4 below for the desired controller and find the correct values for K_C, T_i and T_d.

Table 2.4: Ziegler–Nichols method

Controller	K_c	T_I	T_D
P	$\frac{K_{CU}}{2}$	∞	0
PI	$\frac{K_{CU}}{2.2}$	$\frac{P_U}{1.2}$	0
PID	$\frac{K_{CU}}{1.7}$	$\frac{P_U}{2}$	$\frac{P_U}{8}$

(Haugen 2010)

2.5.6 Fuzzy logic

Fuzzy logic or multi-valued logic have been around since 1930, but did not see commercial use until 1965 when Lotfi Zadeh promoted its use through his famous paper 'Fuzzy sets'. Fuzzy logic is not a logic that is fuzzy, but logic that can handle incomplete or uncertain values. Fuzzy logic attempts to model human's decision making process.(Negnevitsky 2011, p. 4.1)

In order to capture human knowledge or behaviour, fuzzy rules will be used. A fuzzy rule is in the form an IF-Then statement, meaning if something then a consequent will happen.

Example: If speed is fast then stopping distance is long (Negnevitsky 2011, p. 4.5)

There are two types of inference, Mamdani style and Sugeno style. Mamdani style is the original where we find the centroid by integrating over a continously varying function;

$$COG = \frac{\Sigma\, \mu(x) \cdot x}{\Sigma \mu(x)} \tag{2.8}$$

Sugeno style on the other hand uses singleton as the membership function of the rule consequent. This means that at a single point in

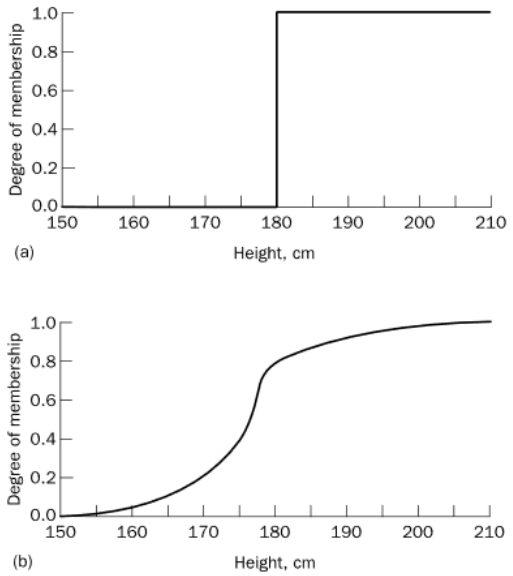


Figure 4.2 Crisp (a) and fuzzy (b) sets of 'tall men'

Figure 2.10: Degree of membership
Source: (Negnevitsky 2011, figure 4.2)

a function the value is "1" for the singleton and zero everywhere else.(Negnevitsky 2011, p. 4.6.2)

This changes the fuzzy set as shown in equation below 2.9:

$$\begin{aligned} &IF\ x\ is\ A \\ &AND\ y\ is\ B \\ &THEN\ z\ is\ f(x,y) \end{aligned} \tag{2.9}$$

From the equation 2.9 we observe that the output z is a function of x and y. The most common form of Sugeno style is the zero-order Sugeno fuzzy model, the output of the equation then changes from a function of x and y to a constant number. Leading to output of the rule consequent for all membership functions to be represented by Singleton spike 2.11.

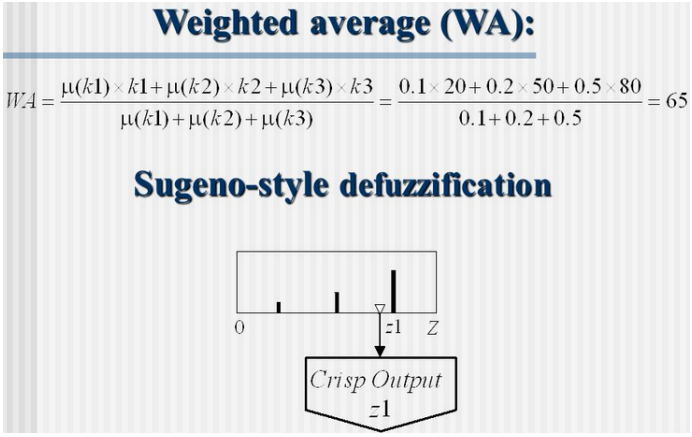


Figure 2.11: Sugeno-style inference
Source: Negnevitsky 2011

2.6 Communication

Communication protocols that were used in the project are explained in this section.

2.6.1 I2C

I^2C (Inter-integrated-Circuit) are used for intercommunication between several devices over two wires. Where one or more devices are masters and one or more other devices are slaves. These wires are SCL(synchronised clock) and SDA (Synchronised data) , which acts as half duplex. I^2C uses the clock to synchronise time up against each device on the bus using the

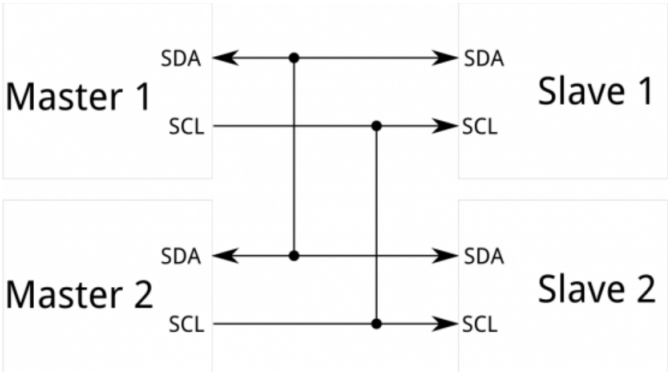


Figure 2.12: I^2C bus
Source:SparkFun.com

master device clock as reference. Data transfer is done through the second wire. I2C is capable of supporting 1008 slave devices on these two wires. However I2C has a relative slow bit transfer rate ranging from 100kHz to 5MHz, compared to modern Ethernet and other bus-protocols. I2C is also restricted to short distance communications.

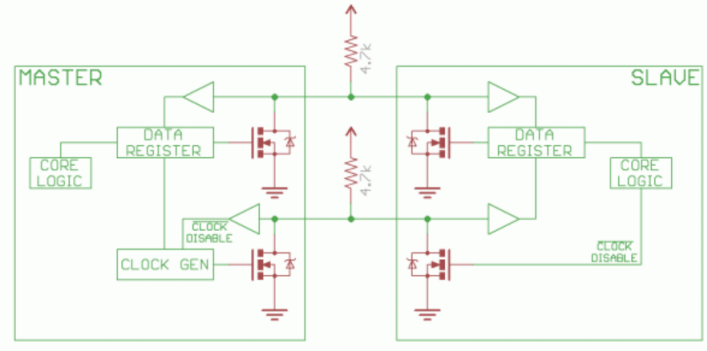


Figure 2.13: I^2C master-slave
Source: SparkFun.com

In order for the I2C to work the two wires (SCL, SDA) need to be connected to pull up-resistors, see figure 2.13. This must be done as I2C are "open drain" meaning they can pull a line low but not push it high. The resistors will thus push the line high again when no device is sending data.

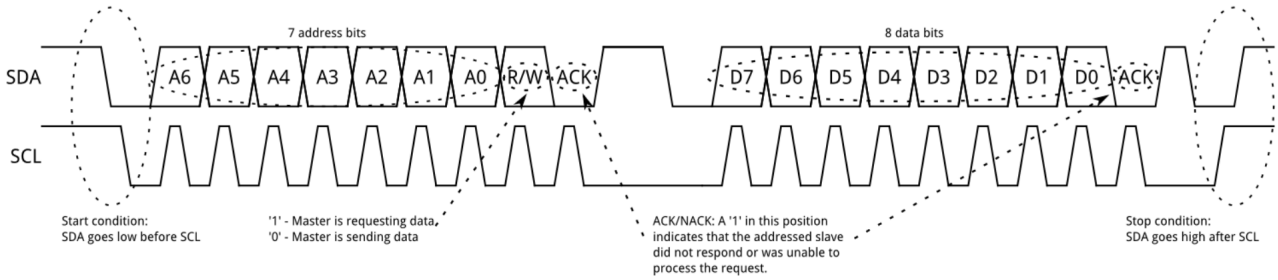


Figure 2.14: I^2C protocol
Source: SparkFun.com

When I2C is used messages are broken down into two types of frames. It is up to the master to indicate to which slave it wants to send/request data from, and how much data will be sent to the target in the form of 8-bit frames like shown in figure 2.14.

Clock Stretching: In I^2C communication the master determines the clock speed the bus will be operating in. This can lead to a master running a clock speed which can be too high for slave devices. When this happens the slave device try to slow down, this is called clock stretching. This enables the slave to reduce the bus speed in order for it to send or receive data. However this can significantly reduce the overall bandwidth of the bus (Telos n.d.).

2.6.2 Power-Line Communication

Power-Line communication (PLC) is a method of sending broad-band data over power lines. It modulates the data signal on top of the of the DC or AC power going through the cable which carries it to a receiver who demodulates it. The advantages of PLC are that no extra cables are required. This is the same technology in the old telephone wires to carry broad-band in the beginning. (Konark Sharma 2018)

2.6.3 SSH

Secure shell protocol, is a protocol developed to provided secure communication over an unsecured network, using the client-server principle. SSH is very similar to a password authentication where username and a password are needed in order to get access. The most recognisable feature of the SSH-protocol is that it enables remote login to computer systems using an Ethernet connection (Ylonen n.d.).

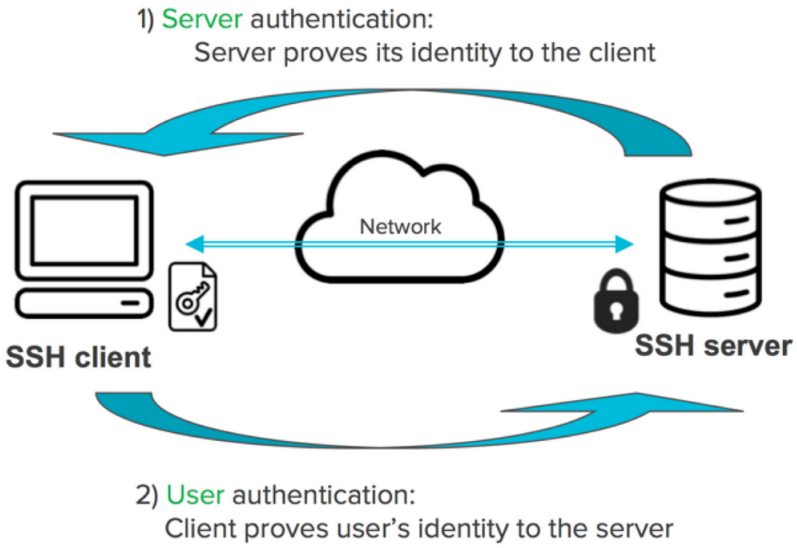


Figure 2.15: SSH protocol
Source: SSH.com

The process of SSH as shown in 2.15 is described below.

1. Server SSH provides its identity to the incoming client.

2. Client presents itself with username to the SSH server.

2.6.4 Serial communication

Serial communication is a way of transferring data over two lines. The data will be sent one bit at a time over the same line or bus. When using serial communications, there are two ways to define the transfer speed. Baud rate, which is how many times a second the line changes state. Bits per second, which is how many bits can be sent each second. The most widely used serial interface is the USB. USB supports full duplex communication, which means data can be sent and received at the same time. (taltech.com u.d.)

2.6.5 NMEA0183

NMEA0183 is a communication protocol created by the National Maritime Electronics Association. This protocol is widely used within the maritime industry for communication between GPS receivers, sonars, echo sounders, etc. It sends data as ASCII code over a serial line with a baud rate of 4800bps. Contrary to the NMEA2000 standard, NMEA0183 can only have one master which transmit data on a single line, but there can be several receivers.

Example; $SDDBT,32.8,f,10.0,M,5.47,F*hh<CR><LF>. This NMEA sentence shows that an echo

sounder is transmitting, the transmitted data is "depth below transducer", then the rest of the data is depth displayed in feet, meters and fathoms. (NMEA 2018; Raymond 2018)

2.6.6 UDP

User Datagram Protocol is a form of connectionless transport, this leads to no communication prior to sending. When writing to the UDP socket, the socket will send a “DatagramPacket” containing the data and a 8 byte header containing; source port, destination port, length and a checksum. UDP Contains no deliverance guarantee and the sender will never know if the package is received or not. If we encounter corrupt data in a package, the package will simply be discarded.(Kurose and Ross 2012)

2.6.7 TCP

Transmission Control Protocol is a protocol for connection-oriented communication. Connection is established through a three-way handshake prior to communication. This makes for fail safe communication between the client and the server. The TCP Header consists of 20 bytes containing source and destination port, sequence number, acknowledge number, etc. TCP guarantees deliverance of all packages and if a package is lost along the way or corrupted, the sender will resend the package. When a package is sent, the sender will wait for an acknowledgement before continuing the communication. (Kurose and Ross 2012)

2.7 Calculations

In this section the equations and formulas needed in the project are presented.

2.7.1 Buoyancy

The buoyancy is the upward force on an object immersed in a fluid. It is expressed by:

$$F_B = \rho \cdot V \cdot g \tag{2.10}$$

Where:
ρ = Density of fluid.
V = Volume.
g = Gravity.

2.7.2 Lift coefficient

The lift coefficient decides how big the lifting force of a submerged object is. In this case the lift force of the airfoils has to be greater than, or equal to the weight of the ROV. The lift coefficient is expressed by:

$$C_l = \frac{L}{\frac{1}{2}\rho \cdot S \cdot U^2} \tag{2.11}$$

Where:
L= Lift force.
ρ= Density of liquid.
S= Wing area.
U= Velocity.

(Aerotoolbox 2018)

2.7.3 Drag coefficient

The drag coefficient describes the force generated by the water resistance on the projected area of a moving submerged object. The drag coefficient is expressed by:

$$C_D = \frac{D}{\frac{1}{2}\rho \cdot S \cdot U^2} \tag{2.12}$$

Where:
D= Drag force
ρ= Mass density of liquid.
S= Projected area.
U= Velocity.

(NASA 2018)

2.7.4 Moment coefficient

The moment coefficient describes the torque applied to the airfoil. This is expressed by:

$$C_m = \frac{M}{\frac{1}{2}\rho \cdot S \cdot U^2} \tag{2.13}$$

Where:
M= Pitching moment.

ρ = Mass density of liquid.
S = Wing area.
U = Velocity.
C = Chord length.

(Aerotoolbox 2018)

2.7.5 Pressure

The pressure applied to a submerged object is dependant on mass density of water, gravity and depth:

$$P = \rho \cdot g \cdot h \tag{2.14}$$

Where:
ρ = Density of fluid.
g = Gravity.
h = Height/depth.

2.7.6 Reynolds number

The pressure applied to a submerged object is dependant on mass density of water, gravity and depth:

$$Re = \frac{v \cdot l}{\nu} \tag{2.15}$$

Where:
v = velocity of the fluid
l = Airfoil width
ν = The kinematic viscosity of the fluid.

(Airfoil-tools 2018[b])

2.7.7 Added mass derivatives for a Prolate ellipsoid

Ellipsoid totally submerged and with the origin at the centre of the ellipsoid are described:

$$\frac{x^2}{a^2} + \frac{y^2}{b^2} + \frac{z^2}{c^2} = 1 \tag{2.16}$$

Prolate spheroid is obtained by setting $b = c$ and $a > b$ The mass of the spheroid is defined by:

$$m = \frac{4}{3} \cdot \pi \cdot \rho \cdot a \cdot b^2 \tag{2.17}$$

$$e = 1 - \left(\frac{b}{a}\right)^2 \tag{2.18}$$

$$\alpha_0 = \frac{2(1-e^2)}{e^3} \cdot \left(\frac{1}{2} \cdot \ln\frac{1+e}{1-e} - e\right) \tag{2.19}$$

$$\beta_0 = \frac{1}{e^2} - \frac{1-e^2}{2 \cdot e^3} \cdot \ln\frac{1+e}{1-e} \tag{2.20}$$

Derivatives of the added mass coefficients

$$X_{\dot{u}} = -\frac{\alpha_0}{2-\alpha_0} \cdot m \tag{2.21}$$

$$Y_{\dot{v}} = Z_{\dot{w}} = -\frac{\beta_0}{2-\beta_0} \cdot m \tag{2.22}$$

$$K_{\dot{p}} = 0 \tag{2.23}$$

$$N_{\dot{r}} = M_{\dot{q}} = -\frac{1}{5} \cdot \frac{(b^2-a^2)(\alpha_0-\beta_0)}{2(b^2-a^2)+(b^2+a^2)(\beta_0-\alpha_0)} \cdot m \tag{2.24}$$

Alternative representation of the mass derivatives is using Lamb, who uses Lamb's k-factors. The k-factors defined for usage are:

$$K_1 = \frac{\alpha_0}{2-\alpha_0} \tag{2.25}$$

$$K_2 = \frac{\beta_0}{2-\beta_0} \tag{2.26}$$

$$K' = \frac{e^4 \cdot (\beta_0 - \alpha_0)}{(2-e^2) \cdot (2 \cdot e^2 - (2-e^2) \cdot (\beta_0 - \alpha_0))} \tag{2.27}$$

Added mass derivatives is by definition then:

$$X_{\dot{u}} = -K_1 \cdot m \tag{2.28}$$

$$Y_{\dot{v}} = Z_{\dot{w}} = -K_2 \cdot m \tag{2.29}$$

$$N_{\dot{r}} = M_{\dot{q}} = -K' \cdot I_y \tag{2.30}$$

Moment of Inertia defined as:

$$I_y = I_z = \frac{4}{15} \cdot \pi \cdot \rho \cdot a \cdot b^2 \cdot (a^2 + b^2) \tag{2.31}$$

Added inertia matrix $\boldsymbol{M}_A(v)$ containing the added mass coefficients:

$$-\begin{bmatrix} X_{\dot{u}} & X_{\dot{v}} & X_{\dot{w}} & X_{\dot{p}} & X_{\dot{q}} & X_{\dot{r}} \\ Y_{\dot{u}} & Y_{\dot{v}} & Y_{\dot{w}} & Y_{\dot{p}} & Y_{\dot{q}} & Y_{\dot{r}} \\ Z_{\dot{u}} & Z_{\dot{v}} & Z_{\dot{w}} & Z_{\dot{p}} & Z_{\dot{q}} & Z_{\dot{r}} \\ K_{\dot{u}} & K_{\dot{v}} & K_{\dot{w}} & K_{\dot{p}} & K_{\dot{q}} & K_{\dot{r}} \\ M_{\dot{u}} & M_{\dot{v}} & M_{\dot{w}} & M_{\dot{p}} & M_{\dot{q}} & M_{\dot{r}} \\ N_{\dot{u}} & N_{\dot{v}} & N_{\dot{w}} & N_{\dot{p}} & N_{\dot{q}} & N_{\dot{r}} \end{bmatrix}$$

Where $X_\dot{u}$ is added mass along X-axis due to an acceleration $_\dot{u}$ in X-direction and so on.
Coriolis and centripetal matrix $\boldsymbol{C}_A(v)$:

$$\begin{bmatrix} 0 & 0 & 0 & 0 & -Z_{\dot{w}} \cdot w & Y_{\dot{v}} \cdot v \\ 0 & 0 & 0 & Z_{\dot{w}} \cdot w & 0 & X_{\dot{u}} \cdot u \\ 0 & 0 & 0 & -Y_{\dot{v}} \cdot v & X_{\dot{u}} \cdot u & 0 \\ 0 & -Z_{\dot{w}} \cdot w & Y_{\dot{v}} \cdot v & 0 & -N_{\dot{r}} \cdot r & M_{\dot{q}} \cdot q \\ Z_{\dot{w}} \cdot w & 0 & -X_{\dot{u}} \cdot u & N_{\dot{r}} \cdot r & 0 & -K_{\dot{p}} \cdot p \\ Y_{\dot{v}} \cdot v & X_{\dot{u}} \cdot u & 0 & -M_{\dot{q}} \cdot q & K_{\dot{p}} \cdot p & 0 \end{bmatrix}$$

Added mass can then be summarised as:

$$F_A = M_A \cdot \dot{v} + C_A(v) \cdot v \tag{2.32}$$

Where:
F_A = Added mass.
M_A = Added mass matrix.
V = velocity.
C_A = Added Coriolis matrix.

(Fossen 1994)

2.7.8 Complementary Filter

The principle of the complimentary filter is to set restrictions on how much trust to be put into corresponding data, then sum these results together.

$$S = \alpha \cdot S_1 + (1 - \alpha) \cdot S_2 \tag{2.33}$$

Where:
α- The ratio of trust of data, the closer to 1 the more reliable the data will be.
S - The combined value
S_1 S_2 - The input sensors to the filter.
Additionally the sum of α and $1-\alpha$ need to be equal 1. (Pieter n.d.)

2.7.9 Depth Calculations

Calculating correct depth in seawater is based on the pressure of the surroundings and the gravity, in addition latitude should also be included to get correct values. Using the empirical formula from (*Algorithm for computation of fundamental properties of seawater* 1983), the following equations are defined.

Gravity variation with pressure is computed as:

$$g(m/s^2) = 9.780318 \cdot [1.0 + (5.2788 \cdot 10^{-3} + 2.36 \cdot 10^{-5} \cdot x) \cdot x] + 1.092 \cdot 10^{-6} \cdot p \tag{2.34}$$

Where variables are defined as:

$$x = \left(\sin \frac{latitude}{57.29578}\right)^2 \tag{2.35}$$

And depth is then calculated from the pressure.

$$depth = \frac{((-1.82 \cdot 10^{-15} \cdot p + 2.279 \cdot 10^{-10}) \cdot p - 2.2512 \cdot 10^{-5}) \cdot p + 9.72659) \cdot p}{g} \tag{2.36}$$

Where:
p= pressure(deciBar)
g=gravity (m/sec^2)

2.7.10 DC-cable resistance

DC resistance for a conductor calculated using formula:

$$R = \frac{\phi \cdot l}{A} \tag{2.37}$$

Where: R resistance of cable.
ϕ conductivity of copper ($1.678 \cdot 10^{-8} \Omega \cdot m$)
l is the length of conductor in m
A is the dimension of the cable in m^2.

2.7.11 Voltage Drop

Formula for calculating voltage drop in cable:

$$\Delta U = I \cdot R \tag{2.38}$$

Where:
I : the current through the object, measured in amperes
R : the resistance of the wires, measured in ohms
(Ormbostad 2014)

2.7.12 Cable Calculations

Formula for calculating cable dimension:

$$A = \frac{ib \cdot \rho \cdot \cos\phi \cdot l}{Un \cdot \Delta U} \tag{2.39}$$

Where:
A-Cable dimension
ib- Load current
ρ - Conductivity-Copper: $1.678x10^{-8}\Omega.m$
$\cos\phi$ - Angle of voltage
l - Length of cable
Un - Rated voltage
ΔU - Voltage drop
(Ormbostad 2014)

2.8 Version control

Version control is the management of changes done to documents, files and so on. These changes are associated with an identity time stamp in order to identify each revision made, and person responsible for the change.
Software development version control kits are essential for multi-group project work, where there are several people working on same code.
There are several types of version control; local version control,central version control and distributed version control. Local version control is where data is stored locally on an users computer, with several different versions of the data. Centralised version control data is stored on a single server, when used a user check out the single file and when done working, uploads the file to the server again. Distributed version control also stores files on an server, but users who will

be working on the files clones the whole project in order to work on it. This way every worker has a copy of the project, and even if the server crashes there will still be a functioning copy of the project on all the users personal databases. (Chacona and Straub 2014)

2.8.1 Git

Git is a distributed review control systems, developed by Linus Torvalds in 2005. Every Git clone is a full fledged repository stored locally, which enables the distributed structure.
In contrast to other version control systems git does not store information as a list of files. Instead we can think that Git think of the files as a series of "pictures", and every change being made is a new "picture". Then a reference of the time the change was made is referenced to that "picture". This way all files are seen by the Git system as a series of "pictures".

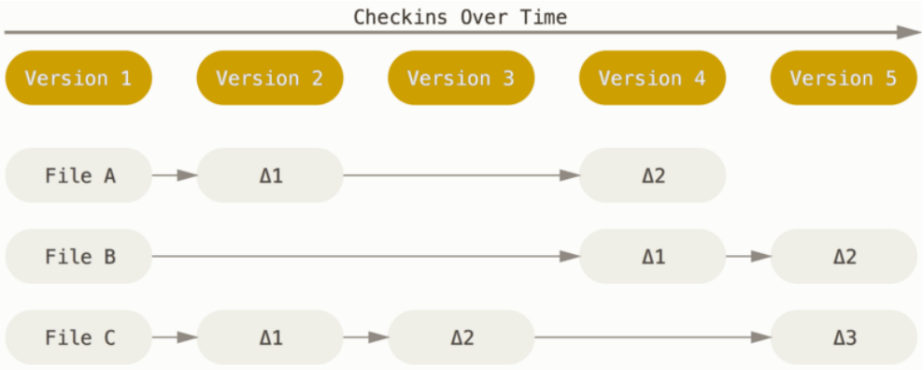


Figure 2.16: git Snapshot
Source:git-scm.com

(Chacona and Straub 2014, Chapter 1.3)

Chapter 3

MATERIAL AND METHOD

In this part of the thesis it is shown the used materials and methods that are applied.

3.1 Data

3.1.1 NMEA 0183

The NMEA0183 communication protocol is used to retrieve data from the GPS and echo sounder. Each of these devices is known as talkers and are connected to an Arduino, which is used to parse the NMEA sentences and send the required information to the on-board computer. There can only be one talker on each NMEA network, the reason for this is that there are no synchronisation in the NMEA0183 protocol and both talkers will try to send data at the same time if connected to the same network, this will again result in data corruption.

Used NMEA sentences

- SDDPT - Depth of Water
- SDDBT - Depth Below Transducer
- VWVHW - Water Speed and Heading
- YXMTW - Mean Temperature of Water
- GPGGA - Global Positioning System Fix Data
- GPRMC - Recommended Minimum Navigation Information

(Raymond 2018)

3.2 Project management

The project work is organised with a project leader and a secretary. The project leader's responsibilities is to ensure progress, being a contact person and delegate tasks . The secretary arranges meetings, evaluates the thesis and write summaries of the meetings. The project is structured on the Lean development principles. This means even though there is a project leader, every member is expected to work independently and ensure progress in their respective parts. During the course of the project the decision process will be collaborative, democratic, and every members opinion weighed equally.

Each week is started with a group meeting, where progress, problems and the plan moving forward is discussed. Meetings with the supervisors is held every other week, in these meetings the progress and solutions of the project are discussed, the supervisors provide useful inputs on the solution and makes sure the group sticks to the schedule.

During the pre-report phase (see apendix A), a feasibility study and development plan were created. These are used as a schedule for what needs to be done throughout the project phase. Asana is used to create a gannt diagram, from this diagram a schedule is made with the use of Instagantt as can be seen in appendix B. Google Drive is used to store and share documents, code, sources and other useful information during the course of the project. Additionally Sourcetree will be used for version control of Java code, and allowing multiple persons to work on same code.

3.3 Utilised Software

Software programs utilised in the project:

Table 3.1: Utilised Programs

Program	About
NetBeans	NetBeans is an IDE used to write Java code
MatLab	Computing environment with a wide range of possibilities.
Simulink	Simulation tool in MatLab
Visual Studio Code	General purpose IDE used to write python code
Arduino IDE	Arduino IDE lets you build and up-load code to the Arduino micro controller.
Visual paradigm	Software design tool
Eagle	Eagle is a PCB design and schematic program.
Simens NX	PLM software used for design, simulation and manufacturing.
AutoCad	CAD software used for drawing mechanical drawings and drafts.
RhinoZeros	Software used for creating sketches and transfer them to a laser cutter.
Excel	Spreadsheets where calculations can be made.
SourceTree	Git client development tool which is used to visualise and manage remote repositories.

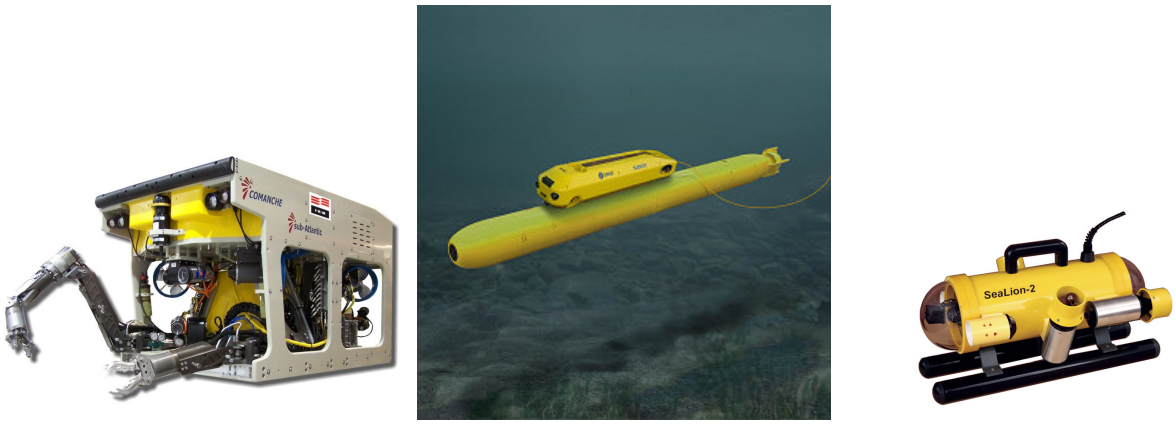
3.4 Concept Studies

3.4.1 Frame

When designing the frame for the ROV, several factors needs to be considered:

- Drag
- Simple production and assembly
- Cost
- Weight

The choice of the frame is heavily dependant on the operation area of the ROV. There are two main types of ROV frames: Work class ROVs, are usually used to perform sub sea operations at low speed. These are often equipped with a lot of sensors, equipment and usually thrusters. Due to the operation area of these vessels, they are usually not designed with hydrodynamics in mind. The other type of ROV has a more streamlined shape and is mostly used for high speed operations, usually with few or no sensors. These are often used to capture images of the sea bed. A combination of the two types would be optimal for the ROV made in this project.



(a) (b) (c)

Figure 3.1: Different ROVs
a: Work class ROV, b: Torpedo shaped ROV, c: EyeBall ROV.
Source: a: Seatrepid.com, b: Seaeye.com, c: inw.com.eg

3.4.2 Material Selection

One of the first things that has to be determined after designing a frame is the materials which is to be used for the different parts. ROV's are often made out of plastics and metals, as these have proven to be well suited for underwater operations.

Aluminium and titanium are often considered the best metal materials for these types of operations; aluminium because of cost and strength-to-weight ratio, and titanium because of its resistance to salt water corrosion and strength. Stainless steel, brass and bronze are also commonly used; bronze and brass for plumbing fittings, valves and fittings for screws and bolts, while stainless steel is used for hydraulic, pneumatic and electrical fittings, as well as fasteners.

There are many types of plastic materials fitted for sub sea operations. Some of the most common are PVC, ABS and acrylic. For operations in shallow water, PVC is widely used because of its low cost and availability. Even though it is a good material for shallow water, it is not as suitable

for operations at great depths, as it is very fragile. Like PVC, ABS is also used for shallow water operations. It often has bubbles in the structure to make it light weight. Acrylic is usually used as view ports for cameras because it can be completely transparent. Also, the strength-to-weight ratio is good, drawback being that it is quite brittle.

(Ch.4 Steven W. Moore 2010)

3.4.3 NACA profiles

NACA airfoil profiles are mostly used in the aircraft industry, but can also be used for subsea operations (Scott 2018). A NACA profile will generate a lift force by creating low pressure on the upper surface of the wing. This will lead to a upward force to help lift the vessel (Nakamura 2018).

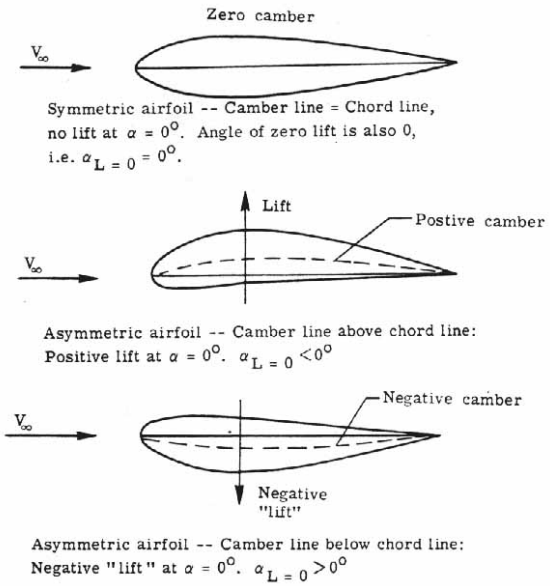


Figure 3.2: Symmetrical, positive camber and negative camber

Source: Slideplayer.com

There are no specific method for choosing a NACA profile, but by adding parameters, a profile can be generated. A 4-digit profile consists of 4 parameters; MPXX, where M is the maximum distance between the chordline and camber line multiplied with 0.01, P is distanse to the max camber in the horizontal axis multiplied with 0.1 and XX is the thickness of the airfoil relative to the chord divided by 100 (Airfoil-tools 2018[a]).

There are generally two types of NACA airfoils; symmetrical and asymmetrical profiles. A symmetrical airfoil will not generate any lift at an angle α=0° in which case an asymmetrical airfoil will (Airfoil-tools 2018[a]). Since the buoyancy of the vessel is supposed to be negative, it is desirable to use an asymmetrical airfoil, but this will have to be tested to get the best results.

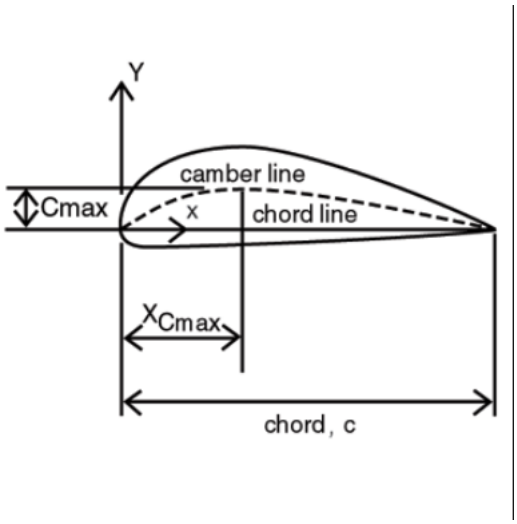


Figure 3.3: Illustration of camber and chord length
Source: akiti.ca

3.4.4 Rotation of Airfoils

To control the airfoils some sort of motors have to be used. There are several options to which type of motors that could be used; electrical actuators, stepper motors and servo motors. Stepper and servo motors are not as powerful as actuators, and since the airfoils will be exposed to drag and lift forces, our solution will be to use electrical actuators. As they combine torque and size to a reasonable price.

There are two types of electrical actuators: rotary and linear. Linear actuators can only move in one axis, while rotary actuators rotates around one axis.

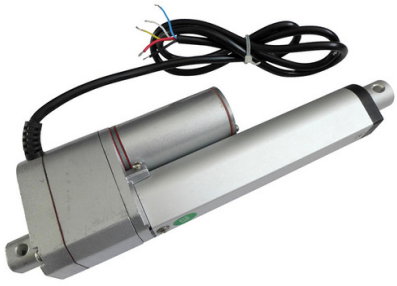


Figure 3.4: Linear actuator
Source: Gimsonrobotics.co.uk

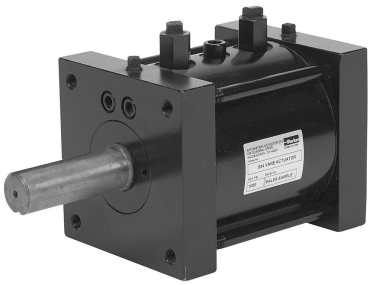


Figure 3.5: Rotary actuator
Source: Parker.com

By using a rotary actuator it may be easier to integrate the motors, as would be connected directly to the shaft going through the airfoils. Contrarily, by using linear actuators some sort of

joint would have to be designed to be able to transform the linear force to a torque. This would make it more complex than using a rotary actuator. On the other hand, rotary actuators tend to be more expensive than linear actuators, when comparing torque. Therefore a actuator have to be chosen based on the required torque and the budget available.

3.4.5 Housing

Most of the electronic parts in the ROV are not water resistant and has to be placed inside a waterproof housing. The most common solution to this is to use cylindrical pressure tubes. This is because the fact that cylinders can withstand pressure better than boxes and other shapes(Ch.5 Steven W. Moore 2010). These may be more difficult to implement in the construction of the ROV than boxes, because of their shape. Additionally, the ROV is not meant for depths greater than 50 meters, which means that the pressure applied to the vessel is aprroximately 0.5 MPa [2.14]. Since the pressure is not very high, it may be more beneficial to use 3D printed or metal boxes, which makes production and assembly easier, as well as being resistant to water.

3.4.6 Placement of wings

Since the wings will generate a lift force, it will be most practical to place the wings above the center of mass. This will make the ROV more stable than if the wings were to be placed lower. By placing the wings below the centre of mass it will make the ROV unstable and more difficult to control.

3.4.7 Power management

The ROV will need to have an cable for power transmission and data transfer. The cable need to be water resistant, withstanding the pull forces exerted on it and transmit power without a voltage drop that would endanger operation. Therefore power drawn by the ROV at 12V have to be calculated based on the data provided by manufacturer as shown in table 3.2 below.

Table 3.2: Calculated max consumption

Component	**Current**	**Power**
Actuator x2	6A/5.2A	72W
Raspberry Pi	1.05A	12.5W
LED Lights x4	2.2A	26.4W
Arduino Mega	250mA	3W
Arduino Mini x2	200mA	2.4W
Depth sensor	1.25mA	0.015W
Leak sensor	10mA	0.12W
Echo sounder	5A	60W
Total consumption	14,7A	176.5W

From this table, using the ampere with set efficiency rate of DC/DC converter to static 0.9 and dimension of cable set to $0.625mm^2$. Using the formula from equation 2.37, the resistance of the cable can be calculated to:

$$R = \frac{1.6780 \cdot 10^{-8} \cdot 200}{(\frac{0.625}{10^6})}$$

Where R = 5.47Ω. From this a new table with voltage drop indication can be generated for the target voltages (3.3). The table will be using voltage drop formula from equation 2.38.

Table 3.3: Voltage in cable

Supply Voltage	**A**	**ΔV**	**Effective Voltage**
24V	8.17	NA	0
36V	5.44	29.75	6.24
48V	4.08	22.31	25.68
72V	2.72	14.87	57.1

Using the specifications in table 3.3, the DC/DC converters range can be limited using the effective voltage and ampere as input.

3.4.8 Control of Actuators

To properly control the actuators, the actuators must be connected to a voltage source with reversible polarity. This can be implemented in multiple ways, but should be done in such a way that prevents the possibility of short circuit occurring. There are several options for motor controllers, but common for all are that they use a smaller control signal to turn on and off;

- High-current transistors
- Motor driver
- Relay

This can be done by connecting the motor to a H-bridge consisting of a number of different transistors. By using a H-bridge, the motors can easily be controlled by a low voltage signal, and be driven in both directions and makes it possible to stop the rotation when desired. Example of a H-bridge in figure 3.6.

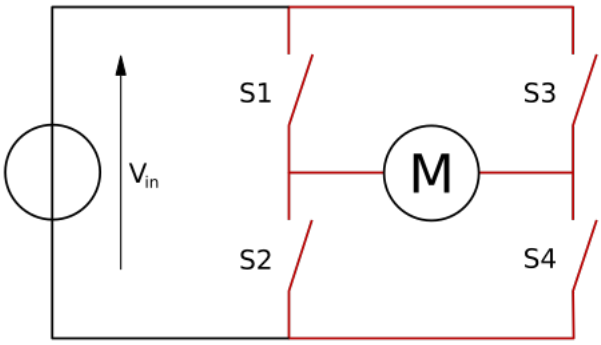


Figure 3.6: H-bridge

To allow proper control of the actuators, they must be equipped with a position feedback to ensure that the actuator provides the foils with the desired angle. Considering the ROV's depth control is supposed to be automatic a fitting control system should provide the actuators with the desired position.

3.5 Method

This section elaborates the methods used to create the ROV, control system and software applications.

3.5.1 Modelling

Obtaining the dynamic equations of the ROV is an important part of designing a control system and creating a realistic simulation. To make future advancement and development on the project easier, a model can be calculated. The model is not necessary to test the concept of this ROV, therefore the modelling will only be a priority after the concept is tested to lay a foundation for future work.

6 degrees of freedom (6DOF), will be used in order to get the equations although only 3 will be used actively by the ROV (heave, pitch and roll).

Table 3.4: Degrees of freedom

DOF	Motion Descriptions moments	Forces and moments	Positions and Orientations	Linear and Angular Velocities
1	Motions in the x-direction (surge)	X	x	u
2	Motions in the y-direction (sway)	Y	y	v
3	Motions in the z-direction (heave)	Z	z	w
4	Rotations about the x-axis (roll)	K	ϕ	p
5	Rotations about the y-axis (pitch)	M	θ	q
6	Rotations about the z-axis (yaw)	N	ψ	r

In order to create a model the following assumption had to be made.

- ROV is a rigid body and fully submerged in water
- Water assumed to be ideal fluid that is incompressible, inviscoid and irrational
- Wave disturbance can be neglected as vehicle is fully submerged

Motion of the surrounding body of fluid in response to the ROV's motion manifests itself as the hydrodynamic forces and moments resists the vehicles motion. This effect appears like the added mass and inertia.

Hydrodynamics forces and moments

Radiation-induced forces and moments can be identified by sum of three components

1. Added mass to inertia of surrounding fluid.
2. potential damping due to energy carried away by generated surface waves.
3. Restoring forces according to Archimedes principle (weight and buoyancy).

The sum of these factors can be expressed mathematically

$$\tau_r = -\boldsymbol{M}_A\dot{\nu} - \boldsymbol{C}_A(\nu)\nu - \boldsymbol{D}_p(\nu)\nu - g(n) \tag{3.1}$$

The added mass terms represents fluid surrounding a submerged body that accelerate with it. This means any motion on the vehicle will generate a movement on an otherwise stationary

fluid. For the vehicle to pass through the fluid will have to move aside and then fill the room left behind the vehicle.
(Fossen 1994)

3.5.2 Control System

There are several solutions when it comes to deciding what type of control system to use in the ROV. An option considered is the MPC controller (section: 2.5.2), which is a good option for predicting forward in time how to control the ROV. Alternative solutions are to use a more traditional PI, PD or PID controller, these are easier to implement than the MPC.

After advice from the supervisors and researching articles and published works, including Dr Peter Ridley n.d. and Nenad Popovich 2006, it is decided to use a PID controller for the ROV.
The PID is a versatile controller which is easy to tune with the help of Ziegler-Nichols method (2.5.5). The gain calculated using this method might need to be tweaked to obtain a desirable overshoot and settling time.

Considering the fact that the system changes according to the speed of the ROV, the ROV needs multiple controllers tuned differently based on speed. To handle this problem a fuzzy-logic system (2.5.6) will be implemented to calculate the different gains of the PID controller. In addition to giving a smooth transition between the different speed levels, it allows for a more accurate and adjustable system. Method on how to build fuzzy system is shown further down in 3.5.2.

PID

By implementing a PID controller, the ROV should be able to control the actuator positions based on the difference between the setpoint and measured depth. A properly tuned controller will give us a responsive and stable system with no steady-state error. By receiving a setpoint from the user and the measured depth, the controller can calculate the error and use it for the proportional and integral control. The derivative part of the controller should differentiate the depth to supply a more stable output than when differentiating the error. This is because when the setpoint is changed the derivative of the error becomes $\approx \infty$, hence making the output inaccurate.

A PID controller can easily be implemented on a microcontroller using existing libraries or writing an algorithm based on the principle explained in 2.5.1. By summarising all of the error values, we can determine the integral and by subtracting the previous input from the current input we can determine the derivative. Using these values along with the error, an appropriate output

value can be calculated.

Building fuzzy system

By following the principles from the book (Negnevitsky 2011) about building a fuzzy system, the steps in which to proceed are given as follows:

- Specify the problem and define linguistic variables.
  Our problem is to provide an output from our PID controller according to the varying speed. The range of these variables also has to be implemented.
- Determine fuzzy sets.
  As fuzzy sets can have a variety of shapes, but simple trapezoid or triangle are often adequate. What is important is to have enough overlap of adjacent fuzzy sets in order to have a smooth response for a system.
- Specify and construct fuzzy rules.
  Utilise if statements to generate output fuzzy number.
- Encode the fuzzy sets, fuzzy rules and procedures to perform fuzzy inference into the expert system.
- Evaluate and tune the system.
  The last and most laborious task, after building the fuzzy system is to tune the system. This is done due to see if the system meet the requirements set at the beginning.

3.5.3 Software development

The application program developed in this project is written in the languages Java, Python-2.7 and Arduino C. These languages have been a part of the core subjects during the bachelor studies for automation at NTNU and were therefore chosen. For a better overview, the software development is differentiated in two main coding projects; one for the ROV application and one for the graphical user interface on the surface vessel.

Java is an object oriented language that fit with most software, this makes it an ideal working tool for a developing project, as it supports real time programming as well. Python is a multipurpose programming language and has the necessary libraries for the raspberry camera, therefor it will be used for the video streaming solution. Arduino C will be used to control the micro controllers. Due to its multiple different I/O's, it will be a good solution for getting data from external sensors.

The program will be of a client-server architecture, with the client running on a computer on the surface vessel. This program will be the graphical user interface for the ROV and display relevant data, store video and data, and send commands to the ROV. The server will be a Raspberry Pi located inside the ROV, and programmed to receive sensor data, transfer data between the ROV and surface vessel, as-well as record video.

In the beginning of the project a plan to identify the software problems were conceived. In order to find out which classes are needed and what their responsibilities should be. Group members with past experience in programming assigned responsibility for the different tasks based on what they wished to work with and in some part on their competence.

The identified problems which needed to be solved are listed as follows

- Develop Graphical user interface
- Server-client communication
- Receiving and process sensor data
- Control movement based on received data
- Create fuzzy expert system
- Streaming camera feed
- Logging and storing data
- Develop expandable platform for connecting sensors and equipment

The concept drawing in figure 3.7 is designed from the bullet points above, to get a visual overview of the software solution:

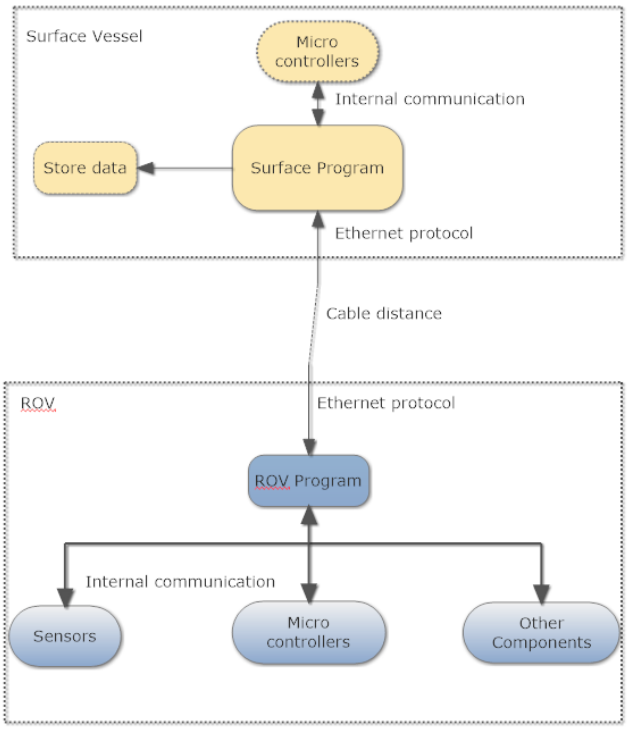


Figure 3.7: Software Sketch

During the development of the software there will be an emphasises on making all shared variables and objects thread safe, in accordance with concurrent programming principles (2.4.4). For communication between threads, shared-variables were used and producer-consumer pattern established see figure 3.8. The producer-consumer principle is that one or more thread will be writing data to a variable, and another thread will be reading. (Gonzalez 2016)

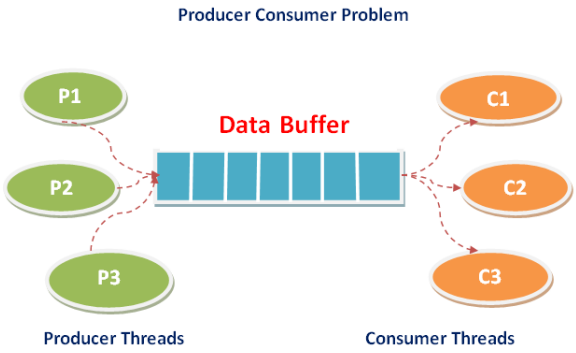


Figure 3.8: Producer consumer pattern
Source: Androidsrc.net

The data will therefore have to be protected to not get errors when reading or writing to them. There were a general agreement that at all times during development that the java application

should be based on the principles of high cohesion and low coupling. High cohesion to make sure a thread performed set task, at most times one task. Low coupling to make sure no class were dependant on other classes to work properly, as this would lead to an unresponsive application. (Kölling 2017, p. 8.3) The application running on the prototype will be a compressed .jar-file in a combination with a start up script which will start the application when powered on and SBC has finished booting up.

GUI

One common problem when creating a multi-threaded GUI, is making sure the different threads does not interfere with each other, and still making the user controls responsive and working correctly. The GUI design must also have a reasonable layout to make it easy to use and understand.

The software must be thread safe and retrieve the bytes of data in the correct order to avoid corrupt data. The correct values must be displayed for the user in a simple manner. User controls must be easy to understand and work correctly with quick response time and safe transfer of information to the ROV.

A video stream from the ROV should be displayed in real-time, and the relevant sensor values will be presented for the user. The user should also be able to control the ROV's depth with a simple and safe depth input. Additionally, a light-switch and emergency stop button should be implemented to optimise lighting and safety. Considering the use of electric actuators, the interface should also contain a warning whenever the usage of these actuators exceeds the duty cycle stated by the manufacturer. Features like I/O control, internet connection and graphic display must be implemented and easily accessible for the users. The framework should contain the ability to add extra features in later updates to enhance the user-experience.

Logging the data and encoding the video images into a video is an important part of this project. This can be done by running one thread for logging data in a specific time interval and one thread for encoding each video image while simultaneously displaying it for the users in real-time. This data can then later be used to observe the surroundings in the ocean, analyse the ROV's behaviour, analyse sensor data and simulate the ROV's movements.

Remote working setup

The server part of the application is to be on a SBC running inside the ROV when it is deployed. To create a better working platform which is more efficient and remove the need to work directly on the SBC, a remote SSH (2.6.3) solution is implemented. By using a remote solution, work can

be done directly on a working computer when using NetBean's integrated remote setup, a java program launched from computer will be sent over SSH to the SBC and there started as a jar program. This allows for simpler developing of the ROV applications as it removes the need to work directly on the SBC, only a power supply and Ethernet connection to make it work.

Libraries

Third party libraries that are used in this project:

Table 3.5: Java Libraries

Java libraries	**Appliance**
Pi4J	Enables full access to the raspberry Pi's I/O capabilities, this include the GPIO pins and I2C.
FuzzyLite	Enables fuzzy logic embedded methods into Netbeans IDE.
JCodec	Encodes a sequence of picture in to any video format.

Table 3.6: Python Libraries

Python libraries	**Appliance**
OpenCV2	Enables methods for image capturing and compression of images in the python code.
PiCamera	Enables accsess and control the Raspberry camera

Table 3.7: Arduino Libraries

Arduino libraries	**Appliance**
Adafruit_GPS.h	Enables NMEA format information to be extracted from the GPS unit.
Wire.h	Enables I2C communication for Arduino devices.
nmea.h	Enables parsing of NMEA sentences on Arduino
Arduino PID library	Calculates an appropriate output based on an error between a setpoint and feedback from a system.
MS5837.h	calculates depth, pressure, temperature

3.5.4 Ethernet Communication

Ethernet will be used as communication between the ROV and the surface vessel. Power-Line Communication (2.6.2) is a simple method for Ethernet over two wires and is a common method of Ethernet communication in cabled ROVs since it uses fewer wires than regular Ethernet cables. The PLC module which is decided to use is the Fanthom-X-3 tether interface, which is better explained in section 3.6.5

The Power-Line Communication is easy to implement, there are two circuit boards, one in the ROV and one in the surface vessel. The circuit boards have an RJ45 socket to connect the Ethernet cable from the computer, a 7-28V input for the voltage which the Ethernet signal is carried upon, and the tether interface itself where the two wires from the ROV cable is connected. Another advantage with the Fanthom-X-3 is that there is no need for a TP-cable, any cable will do.

3.5.5 Internal communication

There will be different hardware in use for the project internally on both platforms, and uniformity needs be achieved. The use of lean, easy to implement and reliable protocols is thus of utmost importance. Serial communication and the I2C protocol were considered, the former due to familiarity as the group have been working with it earlier. In addition serial communication was also needed for parsing the NMEA sentences (3.1.1) from the echo sounder and the GPS. The latter I2C protocol is also a viable option as it is simple to implement (2.6.1) using only two wires for communication, and the majority of sensors selected supports I2C communication. Another argument is that the Arduinos supports I2C with the Wire.h library.

A compromise were made, I2C as internal communication for the ROV, and Serial communication for the surface vessel. This was split up to minimise the number of signal cables in the ROV with the usage of I2C. However, as I2C only supports short distance communication, whereas Serial communication supports greater distances. Serial communication will therefore be used for the surface vessel where the selected hardware could be further apart from the client computer.

The disadvantages of using these two methods is that they are relative slow compared to other protocols, such as Ethernet. However, for parsing NMEA sentences the maximum baudrate is 9600. removing the necessity for faster protocol on this part. I2C is also sensitive to noise and an effort must be made to reduce this vulnerability. (Telos n.d.)
Action to reduce noise include;

- Twisted pair cable
- Shielded cable
- Filtering capacitor's
- soldered connections
- increase distance to noise sources

How to implement I2C is described in detail in 3.5.5

I2C devices working method

I2C works as mentioned in 2.6.1, each I2C devices has an internal address which the master has to poll in order establish a link between them.

The first step of I2C communication is to read the data sheet of the registers, according to what I2C device you will be working with. In these devices, the internal registers that will be used to read/write are described. The registers use hexadecimals as identifiers, but the commands that are sent is 8-bit binary (3.9).

CTRL_REG4_A (23h)

Table 26. CTRL_REG4_A register

BDU	BLE	FS1	FS0	HR	0(1)	0(1)	SIM

1. This bit must be set to '0' for correct working of the device.

Table 27. CTRL_REG4_A description

BDU	Block data update. Default value: 0 (0: continuos update, 1: output registers not updated until MSB and LSB reading
BLE	Big/little endian data selection. Default value 0. (0: data LSB @ lower address, 1: data MSB @ lower address)
FS1-FS0	Full-scale selection. Default value: 00 (00: +/- 2G, 01: +/- 4G, 10: +/- 8G, 11: +/- 16G)
HR	High resolution output mode: Default value: 0 (0: high resolution disable, 1: high resolution enable)
SIM	SPI serial interface mode selection. Default value: 0 (0: 4-wire interface, 1: 3-wire interface).

Figure 3.9: I2C register
Source:LSM303 datasheet.

The master writes to the devices address, then choose which register to access and finally what command what to write to that specific register. This register can be everything from reading values of an I2C sensor or accessing a method of an Arduino. The master then sends "end transmission" when the process is complete in order to disconnect and the bus will be idle. See figure 3.10 below for a illustration of this.

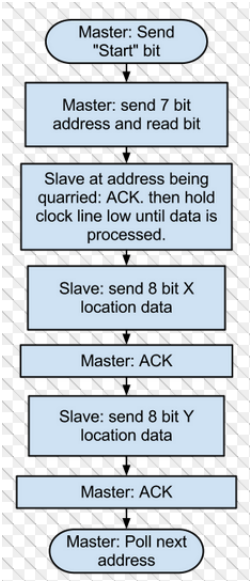


Figure 3.10: I2C init
Source:rit.edu

On synchronised transmissions like the I2C bus, the situation is much more relaxed. The clock is transmitted by the sender and the receiver is always able to synchronise with that clock. I2C defines several speed levels, which is; standard mode: 100 kbit/s, full speed: 400 kbit/s, fast mode: 1 Mbit/s, high speed: 3,2 Mbit/s, are maximum ratings. Compliant hardware guaranties that it can handle transmission speed up to the maximum clock rate specified by the mode. The speed is thus limited by the capability of the connected devices as mentioned in (sec:2.6.1)

3.5.6 Calculations

To predict the physical behaviour of the ROV, several calculations have to be made, like lift, drag, buoyancy and pressure.

Lift force

As described in chapter 2.7.2, the lift coefficient can be expressed by:

$$C_l = \frac{L}{\frac{1}{2}\rho \cdot S \cdot U^2} \tag{3.2}$$

By knowing the lift coefficient you can find the required pitch angle of the wing by looking at the graph in figure 3.11

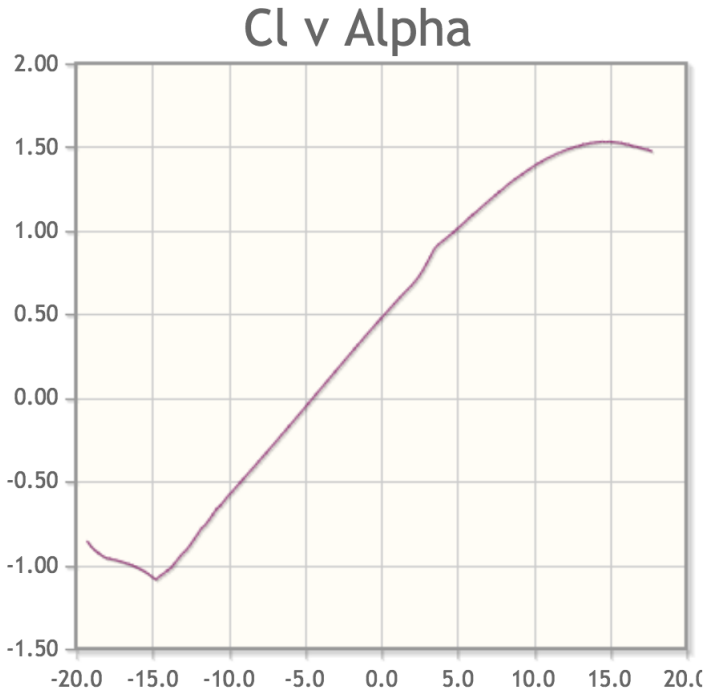


Figure 3.11: Dependancy of angle compared to lift coefficient
Source: Airfoiltools.com

With a speed of 6 knots, a chord length of 150mm and the viscosity of water at 10°C, the Reynolds number was estimated at 367 009 by using the formula in 2.15. The graph used is for a Reynolds number of 500 000 as this was the closest to the results.

Drag force

The drag force is the force from the water that generates water resistance to the front of the ROV [eq: 2.12]. To make the vessel hydrodynamic, it is necessary to design the frame with a shape that will generate as little drag as possible.

Moment coefficient

The moment coefficient can be used to find the torque applied to the wings [2.13]. By knowing this, it is possible to find the required moment to counteract the torque from the wings. This will help determine the required force of the actuators that is to be used.

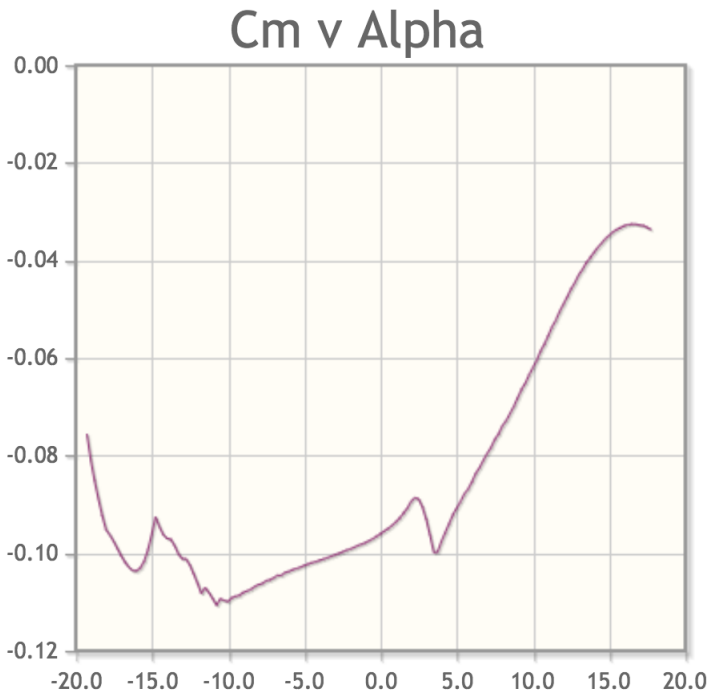


Figure 3.12: Dependancy of angle compared to moment coefficient
Source: Airfoiltools.com

Estimation of distance to ROV

The distance from the surface vessel to the ROV in the horizontal plane can be estimated based on depth and cable length, with the use of the Pythagorean theorem. The ROV cable makes a triangle as shown in figure 3.13, where A and B are perpendicular to each other. The Pythagorean theorem is given by $A^2 + B^2 = C^2$, where A is the horizontal leg, B is the vertical leg, and C is the hypotenuse.

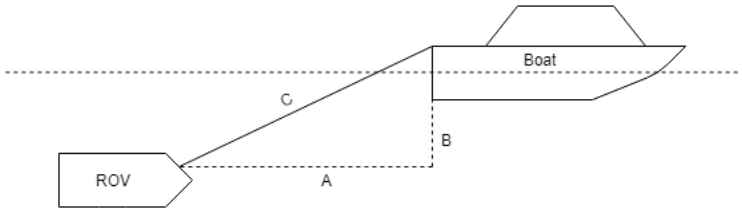


Figure 3.13: Estimating distance to ROV

In this figure, A will be the distance from the boat to the ROV, B is the depth of the ROV, and C is the cable length. The distance is then given by:

$$Distance = \sqrt{C^2 - B^2} \tag{3.3}$$

Pressure

The encapsulations mounted to the ROV will have to withstand a significant amount of pressure. To manufacture them accordingly, it is necessary to calculate the pressure at the depth which it will be operating [eq: 2.14]. This makes it possible to determine the wall thickness and strength of the boxes.

Since the ROV will operate at a maximum depth of 50 meters, the pressure is calculated accordingly. The calculations show that the pressure can be expressed by a linear curve as shown in figure 3.14

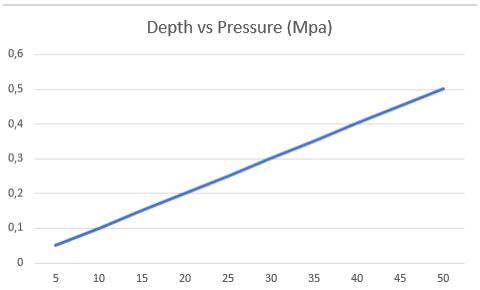


Figure 3.14: Depth/pressure relationship

3.5.7 FEM-Analysis

FEM analysis is commonly used to calculate the strain and displacement of an object. This is done by dividing the 3D model in to a mesh, shown in figure 3.14, and applying constraints and forces which the component is exposed to. It can be beneficial to simplify the model as much as possible, as this will reduce the complexity of the mesh, as well as shorten the time required to perform the analysis. A FEM analysis will have to be done to see if the components will withstand the pressure from the water.

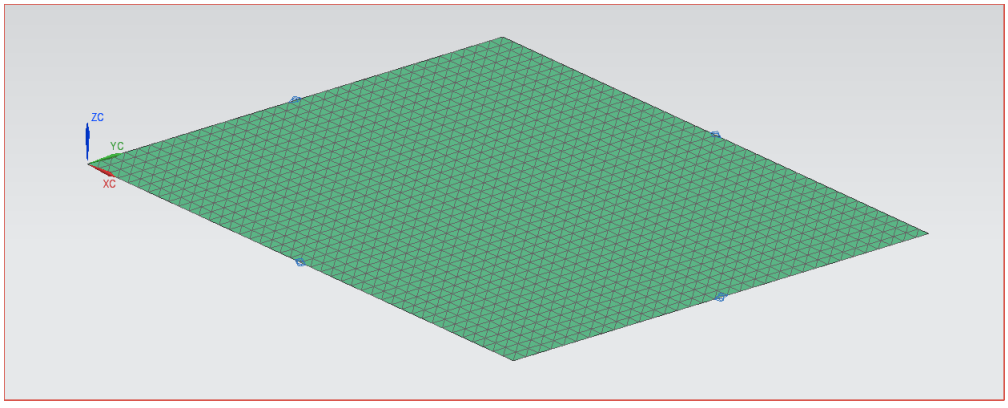


Figure 3.15: Sample of a mesh

3.6 Material

In this section the materials and parts used in the ROV will be presented, why they are chosen, the cost of them, and the usage area. Components without price are parts that the university already had in stock.

Table 3.8: Part and price list

Parts	**Use**	**Quantity**	**Price in NOK**
Plexiglass	Used as the ROV frame		-
PLA	Used for every printed part		-
Aluminium	Used as mounting brackets, lids, stiffeners and legs		-
Stainless steal	Used as bottom plate and weight		-
Nuts, screws and bolts	Used to mount all parts together		-
Threaded rods	Used to mount wing modules	2	-
Shaft	Used for rotating wings	2	-
cable gland	Used for waterproof cable entry		-
Rubber gaskets	Used to create a waterproof seal between boxes and lids		-
Pi Battery Power Board	5V converter for Raspberry	1	180
SOS Leak Sensor	Used to detect leaks in the camera house	1	243
Airmar-DST800 x2	NMEA0183 enabled echo sounder used to measure depth, speed and temperature.	2	5000
Depth sensor	Depth and pressure sensor capable for measuring depths of 300m or 30bar, and temperature.	1	552
Adafruit Ultimate GPS	NMEA0183 enabled GPS breakout board from adafruit with external antenna.	1	325

Table 3.8 continued from previous page

Parts	Use	Quantity	Price in NOK
Adafruit 9-DOF IMU	IMU breakout board from Adafruit with gyroscope, accelerometer and magnetic compass with 3 DOF each. I2C enabled and supports interrupts.	1	162
12V GE DC/DC conv	120W DC/DC converter with 18-75V input and 12V output	1	480
5V DC/DC conv	12 to 5V converter for Raspberry, Arduino and sensors.	1	960
Traco power DC/DC conv	Traco DC/DC Converter with 9-36V input and 12 V output, used for actuators	1	1000
LED Lights	Used to illuminate the view of the ROV	4	208
Capacitors	Used to smooth out voltage levels and filter out noise		-
5V Relay	Used for controlling the lights	1	-
Diodes			-
Transistors	Used to supply signal circuits with necessary voltage		-
Fuses	Securing the circuit	2	-
12V Batteries	3 12V batteries to creat 36VDC for the ROV	3	3000
Ferrite cores	Used to reduce noise	2	-
Mineral oil	Used to prohibit leaks and waterproof components	10l	400
Silicone grease	Used on gaskets to prevent lekage	4pkg	1200
Rubber paint	Used to paint all boxes, to better withstand water	5cans	750
Vulcanised tape	Used to vulcanise all cable glands	2 rolls	110
Arduino Mega	Used for controlsystm and I/O on boat	2	-
Arduino Pro Mini	Used for Eecho sounder parisn in ROV and extra I/O unit	2	-

Table 3.8 continued from previous page

Parts	Use	Quantity	Price in NOK
Raspberry pi fisheye camera	Used to capture video	1	270
Fanthom-X tether interface	Used for Ethernet to the ROV	2	1290
Tether cable	Tow, supply power and signal	100m	5000
SKF6000 ball bearing	Bearings for the wing shafts	2	-
LGLA750 linear actuator	Control movement for the wings	2	3000
Total Price in NOK			**24130**

3.6.1 Raspberry Pi 3 model B

The Raspberry Pi single board computer is used as the main computational hub with in the ROV. It houses a wide array of input and outputs, and runs on the Raspberrian shell on the linux OS. Chosen also because it supports direct input of camera without the usage of USB-port 3.6.4. The raspberry pi supports java library Pi4J 3.5, which enables I2C and GPIO pins control.

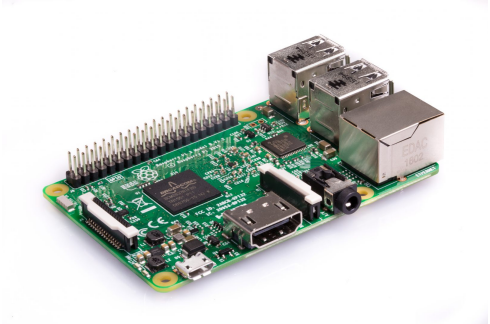


Figure 3.16: Raspberry Pi 3 model B
Source: Raspberrypi.org

3.6.2 Arduino Mega

Arduino Mega 2560 is an Arduino microcontroller that is more powerful, has multiple serial ports, more memory and more I/O ports than the Arduino UNO. The extra I/O also includes more PWM pins that can be used. This Arduino board enables more flexibility and allow for heavier code to be run. This microcontroller is therefore chosen for running the control system and as the parser for the NMEA devices on the boat.

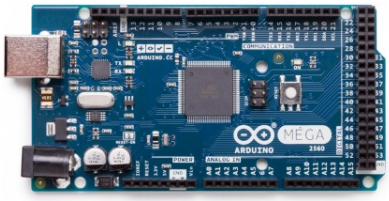


Figure 3.17: Arduino Mega 2560
Source: Arduino.cc

3.6.3 Arduino Pro Mini

The Arduino Pro Mini is a small microcontroller measuring 18x32mm; the small size makes it perfect for the ROV, where there is not much free space. It supports a variety of input and outputs, including an I2C bus and serial communication. This is the reason for using the Pro Mini as an extra I/O card.

3.6.4 Raspberry pi fisheye camera

The Raspberry pi fisheye camera has a horizontal angle of 170°and a resolution of 1080p with a frame rate of 30fps.
The camera connects to the Raspberry via internal headers, which is one of the reasons for choosing this camera. Since it saves room regarding the USB connection, and there is no chance the cable can fall out. Price is also a deciding factor, and the raspberry camera is generally cheap.

Figure 3.18: Raspberry Pi Camera used in the ROV
Source: modmyPi.com

3.6.5 Fathom-X Tether Interface

The Fathom-X-3 tether interface is a Power-Line Communication module which is used in order to get a smooth and reliable data transfer between the ROV and surface vessel. It uses two conductors to send and receive an Ethernet signal, and suport a speed of up to 80 Mbps over 2000m (Bluerobotics 2018). For more details about PLC see section 2.6.2. This part was necessary when it was discovered that normal Ethernet communication through the cable was not possible to achieve, due to it not being revolved.

Figure 3.19: Fathom-X-3 tether interface
Source: BlueRobotics.com

3.6.6 Umbilical neutral tether ROV cable

100m $14 \times 0.5 mm^2$ Kevlar reinforced neutral buoyant copper cable., with a breaking strain of 200KgF. This cable was chosen over similar designs due to:

- Waterproof
- Kevlar-reinforced enables towing with the cable

- 14 conductors enables flexibility for transfer of data/power.
- Neutral buoyant does not add weight to the ROV.

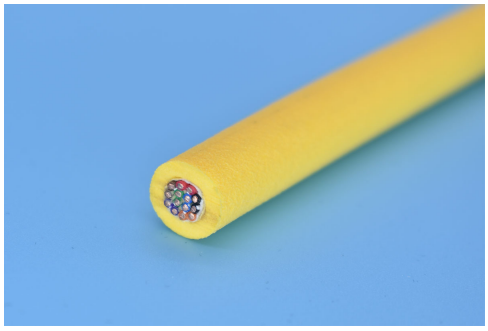


Figure 3.20: Umbilical neutral tether
Source Derucable.com

3.7 Test Setup

In order to test the prototype, a PC with the GUI-application is needed. As well as a RJ45 port, and 3 12V batteries to provide power. The PC needs to be connected to the ROV using the Ethernet-cable over a local LAN-network, due to the use of static IP-addresses being used. The static IP allows for hardcoding into the software with IP addresses to connect to for the ROV-application and the GUI-application. The steps for connecting to the ROV-application is described as follows:

1. Fasten the cable to the surface vessel.
2. Start Client PC.
3. Set IP-address of client PC to: 192.168.0.20
4. Connect USB for the onboard microcontroller.
5. Start client application when ready.
6. Deploy ROV to the water surface.
7. Switch on ROV power supply (36VDC).
8. Wait 2 minutes for the SBC to boot and sensors to calibrate.
9. Video will start when the ROV has booted.
10. Use the “file” option in the GUI-application and click “connect”

11. Enter 192.168.0.10 as the IP-address and click “Ok”.
12. Values will be updating, video is showing and commands can be sent.
13. Return to step 5 if client fails to connect.
14. Release the ROV into the water and set wanted depth.
15. Check that the IP-address is: 192.168.0.10, set it to this if not.
16. Click OK.
17. Values will be updating, video is showing and commands can be sent.

A script located on the ROV-SBC will be auto-started when booting has finished 4.4.7. The script starts both the video stream server and the server-application. After a connection has been made, control of the ROV can ensue, including controlling lights and I/O connected. For more information see result 4.52

3.7.1 Systems Tests

In this section simple tests performed to components and small systems are described.

Voltage drop, cable, DC/DC-konverter

Cable will be tested for voltage drop by rolling out cable and couple conductors to (+ -) supply voltage. This will indicate voltage drop at idle load, medium load and heavy douty load.

GPS

The GPS is tested with the Arduino Mega used for the GPS and echo sounder on the surface vessel. This Arduino is connected to the Adafruit GPS breakout board, supplying it with 5V from the Arduino and connecting the NMEA + and NMEA - wires. Additionally the microcontroller is loaded with the code meant to parse the GPS sentences and prints out the parsed data. An external antenna is connected to the GPS module and placed outside a window. The GPS test includes but is not limited to testing longitude, latitude and heading.

Echo sounder

The echo sounder test is carried out in a test tube with a height of approximately 140cm and 15cm in diameter. The echo sounder is mounted inside an orange box, and is the same as used on the ROV and surface vessel, an Airmar-DST800. This is connected to a 12V source and the NMEA data wires are connected to an Arduino Pro Mini through an optoisolator, which is the same circuit as used in the ROV and surface vessel. The arduino was loaded with the same code as when it is inside the ROV, expect form it printing the data instead of sending it. The test tube is filled to about the same height as it is, and the echo sounder is placed resting on top of the tube, as can be seen in figure 3.21 to the right.

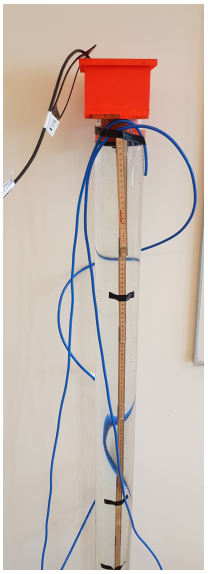


Figure 3.21: Echo Sounder Test

Inertial measurement unit (IMU)

The IMU using the accelerometer reacts to g-forces acted upon it, and the gyroscope reacts to a change in position. As the IMU is used to calculate the angle of the ROV, it is important to test if the device is working properly and give the correct values as outputs. The IMU data the accelerometer and gyroscope also has to be processed and the calculations (eq: 2.33) used has to be tested to produce the correct answer.

Depth pressure sensor

The pressure sensor is used as one of the main sensors for setting the depth of the ROV 3.5.6 and therefore has to give the correct data. The raw data from the pressure sensor has to be calibrated and processed, consequently it is essential to test that every part of the process works.

I2C-bus

The I2C bus is the internal communication method used inside the ROV, in order for everything to be working together and having an operational ROV, this communication has to work flawlessly. By testing the I2C bus for every possible data generation method, any fault detected will be accessed. Then methods for safe communication can then be applied to the protocol, hence making sure no important data is lost.

Leakage sensor

The ROV will be operating underwater and therefore a big challenge is waterproofing all boxes used to house the electrical components. As a mean of further securing the electrical components, a leak sensors will be used. These sensors will be placed in critical sections of the ROV and whose job is to detect if a leak occurs. As these sensors will be the indicators in the event of a plausible leak, which means they have to be tested and approved.

Led-Light GPIO output

The Led will be used to brighten up the surroundings of the ROV, as it submerges deeper into the ocean. The Led will be controlled from the user interface and will have to be tested several ways.

3.7.2 Water test

To assure that components are waterresistant, several water tests will have to be conducted. This is crucial to ensure that none of the electrical components inside will take damage from water leakage. The tests will have to be executed in shallow water first, following more testing at greater depths to make sure the water pressure will not cause leakage.

Stability test

The ROV have to be tested for stability to ensure it will not roll over or become unstable, and also to see if it needs extra weight. This test will be conducted in the tank test and in the ocean.

Chapter 4

RESULT

This chapter will present the achieved results from the project. The main objective in the project was the development of a prototype of a ROV, including design, control system and software. In addition to this, equipment on a surface vessel were needed.

The ROV was tested in water at the end of the project. This was done by towing the ROV after a small boat and testing the different systems such as camera, lights, sensors and control system. It behaved well in the water and was stable, the control system did what is was supposed to and the software worked as intended.

Figure 4.1: Finished prototype

In this chapter the design, construction and how the ROV worked will be described in detail. It is split in five parts: prototype, surface vessel, control system, software solution and test results.

4.1 Prototype

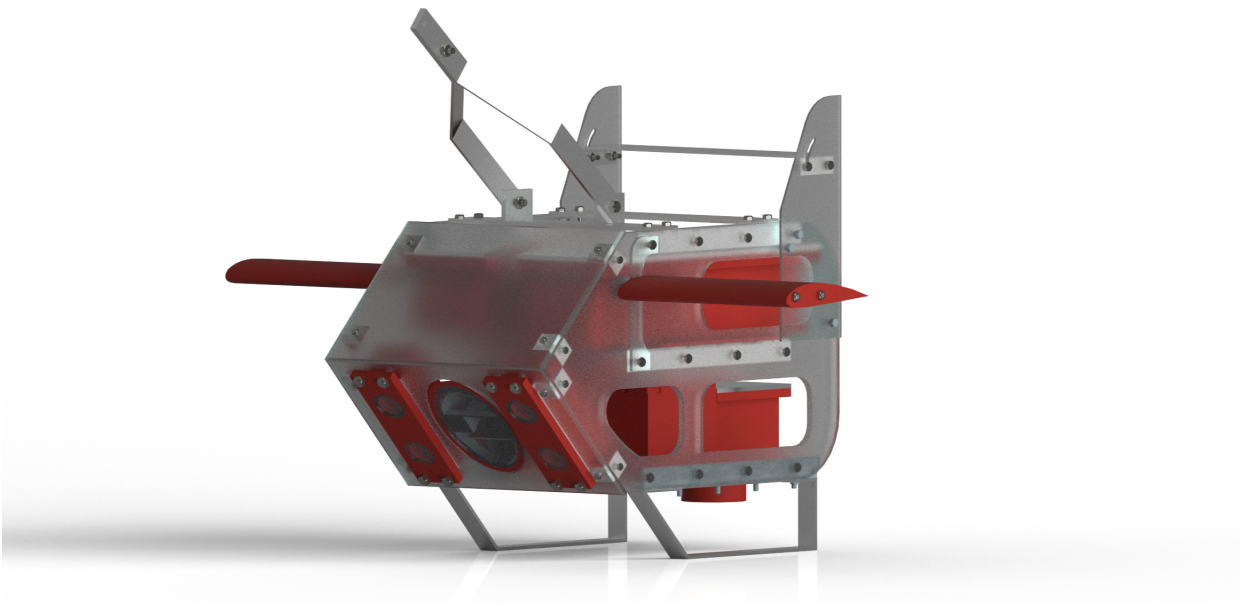


Figure 4.2: Final design of prototype

This section focuses on the process of building the prototype, and the chosen equipment will be described. The main goal during the construction have been to build a functional and low cost prototype. The objective of the prototype is to test of the DCS (Depth Control System). The prototype should be small enough to be handled by a single operator, but still have enough space to fit extra sensors. The size had to be calculated according to the volume of the actuators, different hardware and lift capacity of the connected wings.

4.1.1 Frame

The main frame of the ROV is made out of aluminium, stainless steel and acrylic. It has a box shaped body, and a front cover. The reason for this was to make enough room for all the equipment and to reduce the drag as much as possible with the materials and methods available.

The frame was designed to avoid welding, mainly because the design changed rapidly, and this made it easier to modify the different parts. This also made the assembly of the frame simpler.

It consists of two side plates, a front cover and aluminium plates in the middle. The side plates are made out of 8mm thick Plexiglas, because it is cheap and easy to work with. Since this is not a finished product, cutting out the plates in Plexiglas made it easier to replace them if the design

was to be changed or the components damaged. They are made with cutouts to reduce weight and to add and remove parts without having to remove the plates.

The aluminium plates' main function is to strengthen the frame, as well as for mounting of equipment. They were going to be made out of stainless steel at first to avoid corrosion, but were made out of aluminium because it was available at the lab, and it is much lighter than steel. They were mounted to the side plates by bolting them to aluminium angles which again are bolted to the side plates.

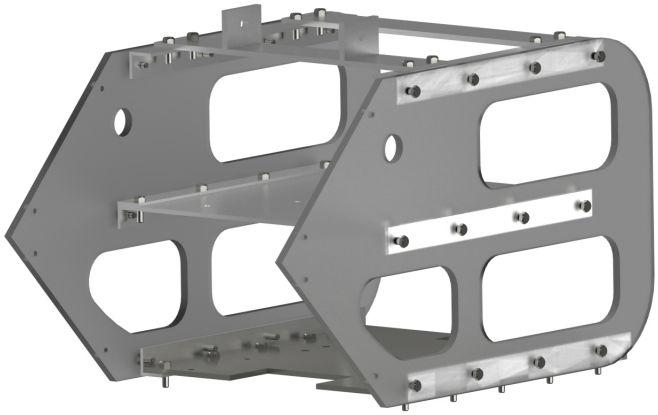


Figure 4.3: The main frame of the ROV

4.1.2 Wings/ profiles

To decide which NACA profile that was to be used, a spreadsheet was made, this can be seen in appendix G. By considering several factors like weight, speed, water density and viscosity, it was possible to narrow down which profiles to use.

Several different profiles were tested and the NACA 4418 was selected. This profile have a max camber of 4% located at 40% , and has a thickness of 18% of the camber line (section: 3.4.3). By deciding a lift force based on the weight and desired speed of the ROV it was possible to find the necessary lift coefficient. The lift force has to be equal to or greater than the submerged weight of the ROV in order to obtain an upward speed. The weight of the ROV itself is approximately 20 kg, which means that each wing has to lift 10 kg. From NX, it is possible to find the surface area of the NACA profiles which is measured at 78999 mm2 for each wing. If a speed of 6 knots=3.1 m/s and a water density of 1025 kg/m3 is used, the lift coefficient is:

$$C_l = \frac{10kg \cdot 9.81m/s^2}{\frac{1}{2} \cdot 1025kg/m^2 \cdot 0.079m^2 \cdot 3.1m/s^2} = 0.2 \tag{4.1}$$

The wings on the ROV are also modular, mainly because the required length of the wings changes depending on the speed of the vessel. By using modular wings, the lift force can be changed by adding or removing modules. Figure 4.3 shows how the lift changes depending on how many modules that are mounted to the ROV.

This is solved by making parts of wing-shaft fit into each other. The first part is mounted to a shaft connected to the actuator while the rest are connected to the first one by using two threaded rods which goes through the whole wing. This provides the option to modify the length of the wings without having to make entirely new ones. The modular foils are demonstrated in the picture below 4.5.

Number of modules on each side	Lift at 0°		Lift at 15°	
0	15,31242333	N	57,421587	N
1	30,62484665	N	114,84317	N
2	45,93726998	N	172,26476	N
3	61,24969331	N	229,68635	N
4	76,56211664	N	287,10794	N

Figure 4.4: Lift force per module at 6 knots

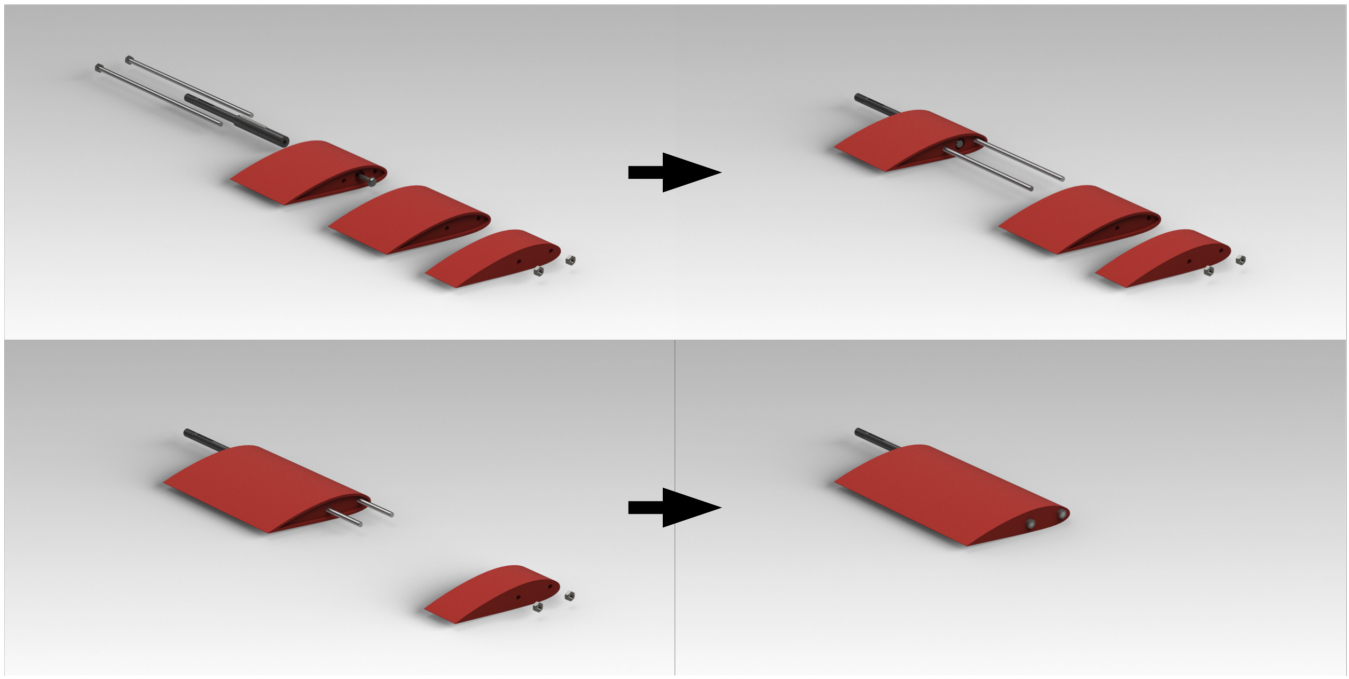


Figure 4.5: Demonstration of wing assembly

4.1.3 Front cover

To make the ROV as hydrodynamic as possible, it is important to make a shape that will generate the least possible drag [3.5.6]. This was done by making an angled profile in the front of the ROV made out of acrylic with two plates placed 45°from the vertical axis, with a 90°angle to each

other. The lower plate has cutouts for the camera housing and the lights. A drag coefficient is needed to calculate the drag [2.7.3]. This information was found in DNVGL Recommended Practice C205 (*DNV-RP-C205* 2010). There are no given drag coefficient for the shape of the vessel, so the coefficient for an isosceles triangle was used. By using this drag coefficient it is possible to find the drag force that is applied to the ROV. The drag coefficient for the isosceles triangle with an angle of 90°is defined as 1.6.

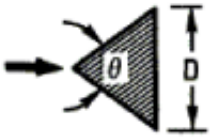
15. Isosceles triangle	θ	C_D ($R_e \sim 10^4$)
	30	1.1
	60	1.4
	90	1.6
	120	1.75

Figure 4.6: Drag coefficient of isosceles triangle

By using a drag coefficient of 1.6 you get the drag force:

$$1.6 = \frac{D}{\frac{1}{2} \cdot 1025 kg/m^2 \cdot 0.3m * 0.3m \cdot 3.1 m/s^2} \tag{4.2}$$

$$D = 1.6 \cdot \frac{1}{2} \cdot 1025 kg/m^2 \cdot 0.3m * 0.3m \cdot 3.1^2 = 709.2N = 72.3kg \tag{4.3}$$

This means that the cable will be capable of handling approximately 2.76 times the drag force of the ROV at max intended speed. [3.20]

4.1.4 Wing control

Electrical linear actuators were used to control the wings. When calculating how much force the actuators had to use, the torque from the airfoils had to be found. It is complicated to calculate the force necessary to rotate them, therefore the torque required to stall the wings was calculated. The graph in figure 3.12 shows that the highest moment coefficient possible is -0.12. By using this, the necessary moment was found to be 6.9 Nm. With a distance from the wing shaft to the actuator of 33 mm, the required linear force is 208.8 N. While this seems high, it is important to recognise that this is the maximum force, and normal use will not require this much. Since this is the required moment to stall the wings, an estimate was made and the required force from the actuators were doubled, which gave a force of 417.6 N. The linear actuators that is used can handle 750 N, therefore the results show that the actuators will be able to rotate the wings properly.

To make the wings rotate, a joint is connecting the actuators to the rods going through the wings. By using a bolt, the actuator is connected to a square piece of aluminium, which is also connected to the wing shaft. This makes the wings rotate when the actuator moves in a linear motion. This can be seen in figure 4.7

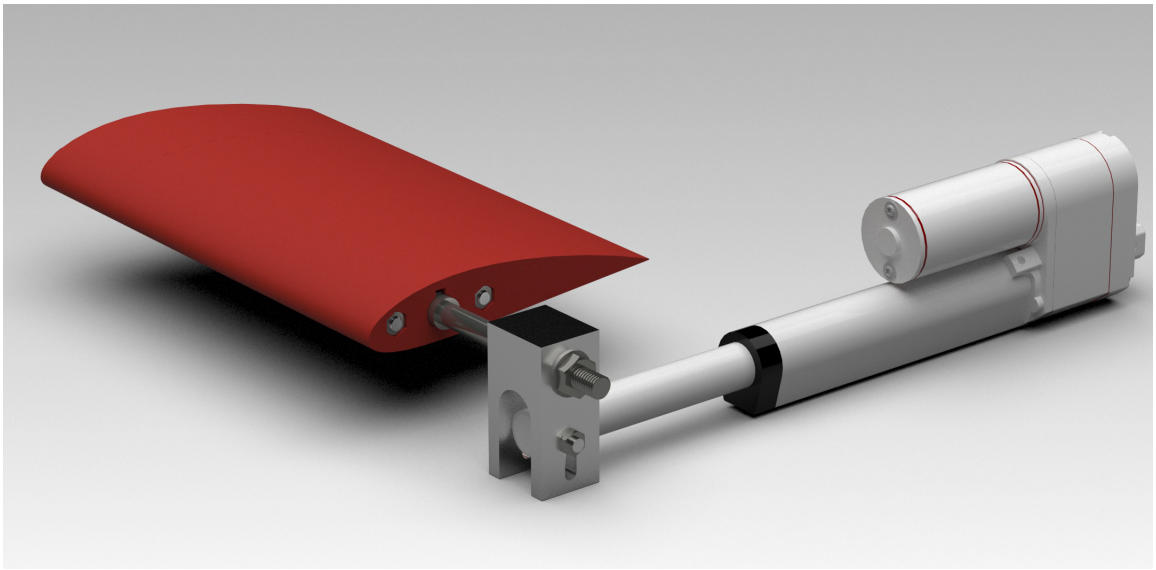


Figure 4.7: Joint connecting the foil to the actuator

There were also drilled holes in the enclosure of the piston to the actuators. It was done since the compartment containing the actuators will be filled with oil, and when the pistons move, they change the occupied volume. The holes allows the oil to enter the capsule and float behind the pistons when it moves forward, resulting in no volume change.

4.1.5 Camera Housing

Figure 4.8: 3D printed camera housing

The front of the camera housing is made out of plexiglas to make it transparent. This is placed at an angle of 45°to make it possible to capture video of the bottom as well as the front. This was glued to the housing using Ess Tack.

There is also a frame inside the housing to make it easier to remove and place the camera and the Raspberry pi. The components are first placed to the frame, before the frame is put inside the housing.

Both the housing and the frame are 3D printed. The housing was printed in 60% infill, since it could not be filled with oil. The frame was printed in 20% as it does not have to withstand pressure when placed inside the housing.

To enclose the housing, a lid was screwed onto the back of it. Cutouts are made on the opposite side to hold the nuts in place when tightening the lid as shown in figure 4.10

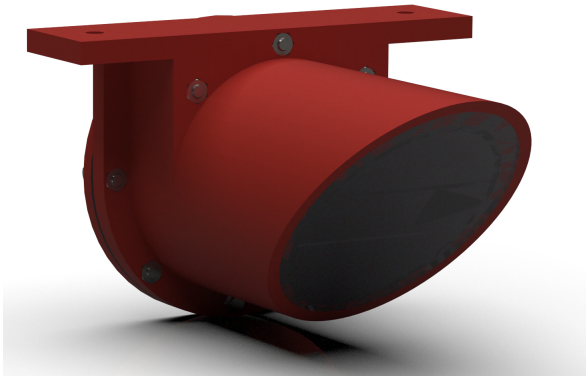


Figure 4.9: Camera housing

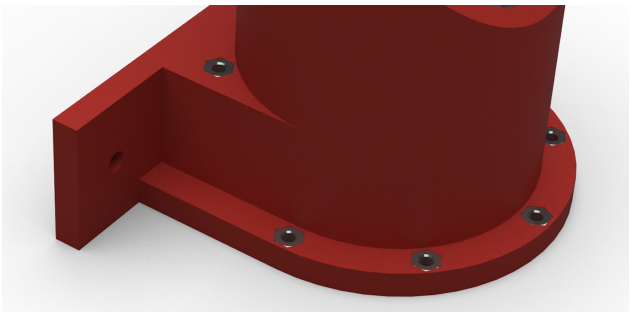


Figure 4.10: Cutouts for nuts on the camera housing

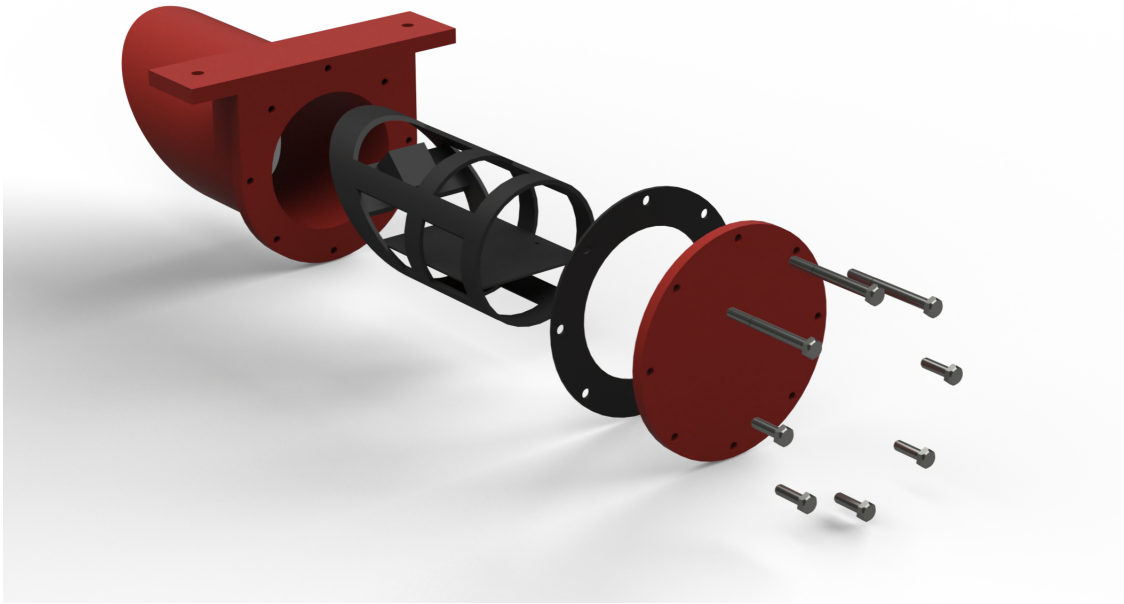


Figure 4.11: Demonstration of camera house assembly

4.1.6 Trim spoiler

A spoiler was placed at the rear of the ROV as shown in figure 4.12. Its purpose is to adjust the pitch of the vessel. The reason this needs to be adjusted is because the wire connected to the boat is placed in the front of the ROV which will cause an upward tilt. By generating an upward lift in the back as well, it is possible to stabilise the ROV.

The spoiler is attached to the frame by two plates on the sides in the back. A spoiler is placed between these plates and screwed in place. This spoiler can be modified, however it needs to be

adjusted manually. That means that the necessary angle needs to be determined before placing the ROV in water.

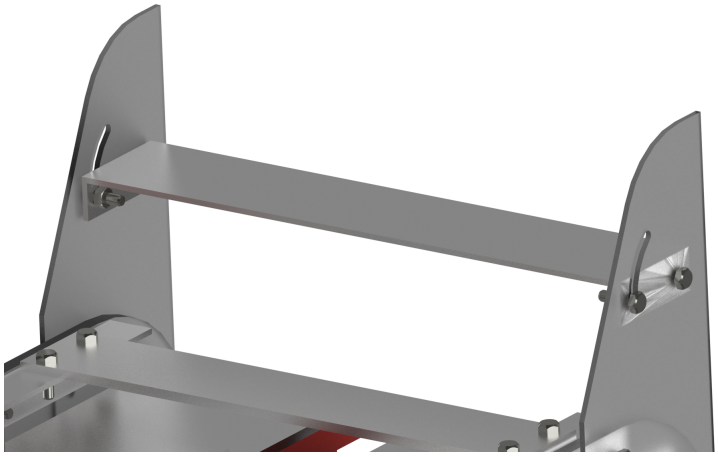


Figure 4.12: Trim spoiler placed in the back of the ROV

4.1.7 Boxes and sealing

The ROV is equipped with different sensors and apparatuses. Most of this equipment is not water resistant and is therefore placed inside waterproof boxes and installed into the ROV. This makes it easier to remove and add the sensors that fits the desired operation area. The boxes are all 3D printed with 20% infill, and should withstand pressure up to 50m of depth (section: 3.5.6). One of the problems with 3D printed materials is that the object is printed in layers, which may lead to leakage of water between them. This turned out to be a problem during the water test in the tank, which can be seen in section 4.5.1. The boxes are therefore coated in rubber spray paint and all joints and corners glued with Ess Tack to prevent water from penetrating. The orange colour also makes the ROV easier to spot underwater.

To avoid further leakage, the boxes were filled with mineral oil as can be seen in figures 4.13 and 4.14. Since oil is not conductive and does not compress, it is well suited for keeping water from leaking in.

All the boxes also have lids made out of aluminium plates. These are screwed on with a rubber gasket between the box and the plate. These were greased with silicon to also prevent leakage, which can be seen in figure 4.13 and 4.14. The lids were made out of aluminium instead of 3D printed material, since this makes it easier to tighten them without breaking or cracking.

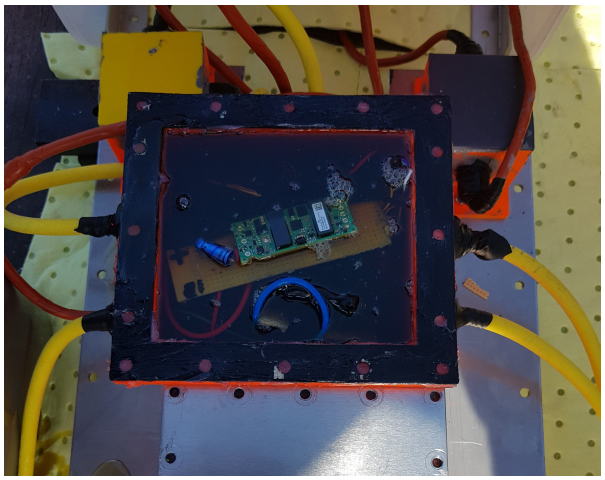


Figure 4.13: Electronicsbox oil filled

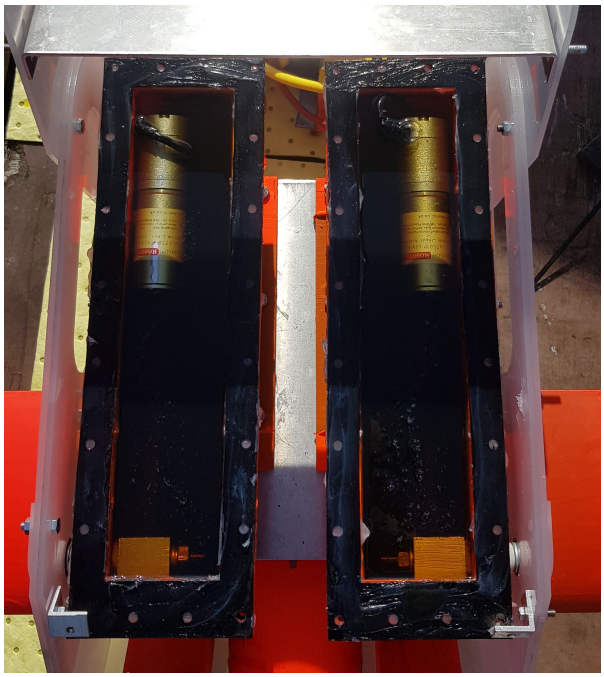


Figure 4.14: Actuatorbox oil filled

A FEM-Analysis was done on the boxes to assure it was going to withstand the pressure. It was done to the electronics box, the actuator box and the echo sounder box. The models used in the analysis are simplified as it is more efficient when the program is running. The results of the simulation are shown in figure 4.15 - 4.17. The results show that the boxes will withstand the pressure as the yield strength of PLA is 37 MPa (Giang 2018).

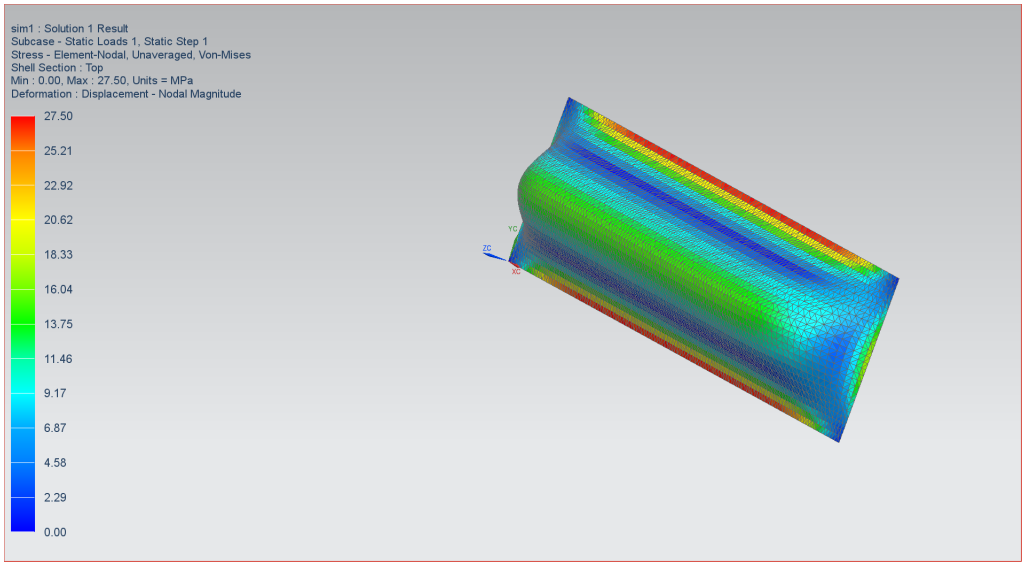


Figure 4.15: FEM analysis of actuator box wall

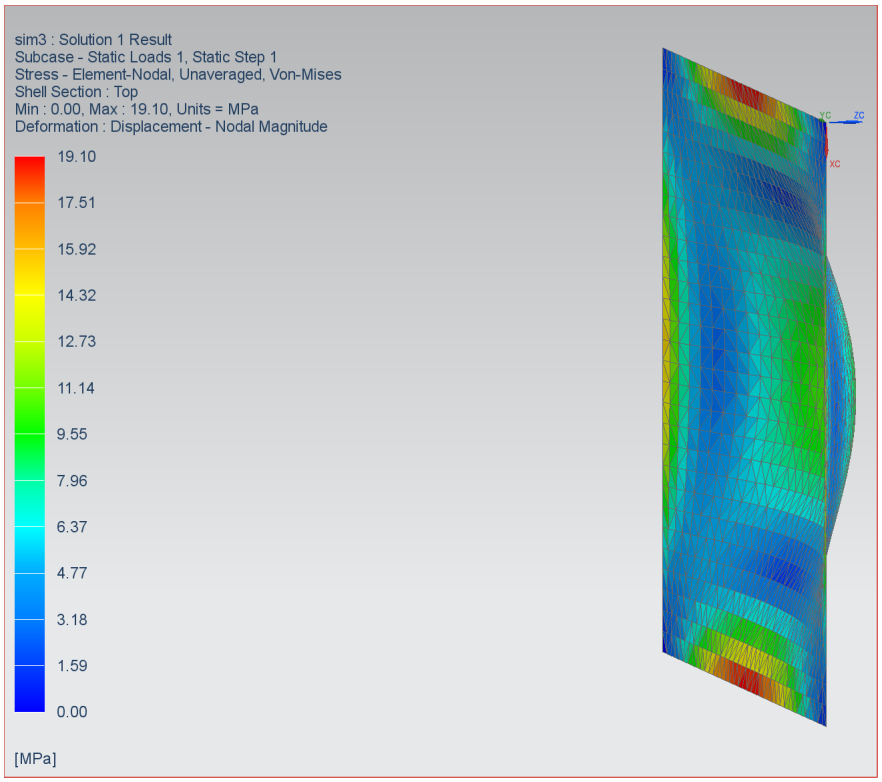


Figure 4.16: FEM analysis of echo sounder box wall

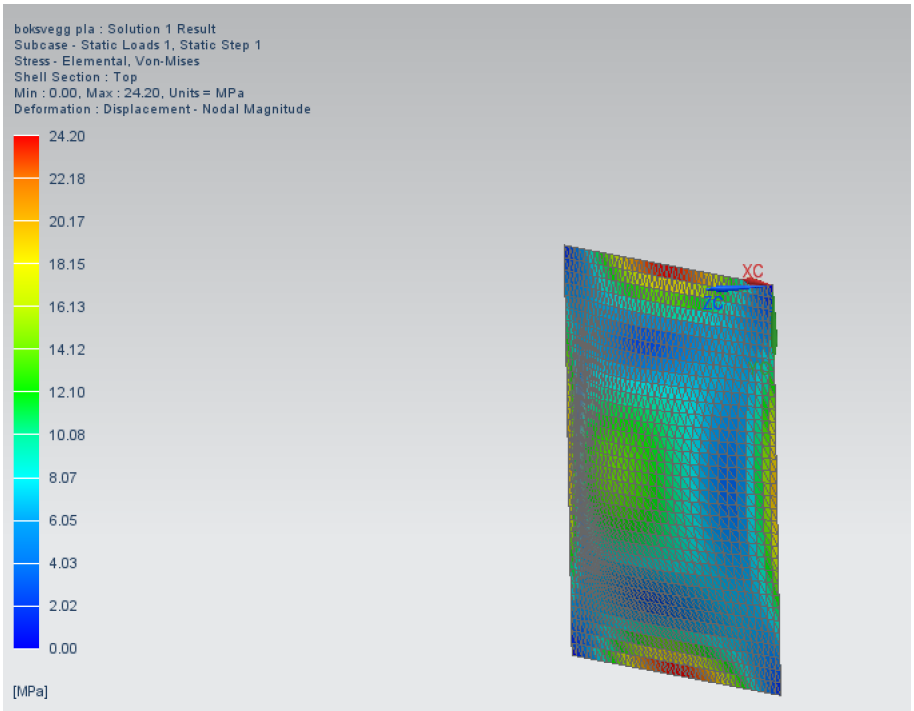


Figure 4.17: FEM analysis of electronic box wall

The echo sounder box had a hole to mount the echo sounder through. This hole was thoroughly greased with universal grease and it has its own sealing as well. The echo sounder entered into the hole and was then tightened with a nut. The hardware of the echo sounder is also cast in epoxy from the manufacturer, enclosed it and protecting it from foreign intrusion. The pressure sensor were also placed in this box, it was cast into the box using two component epoxy.

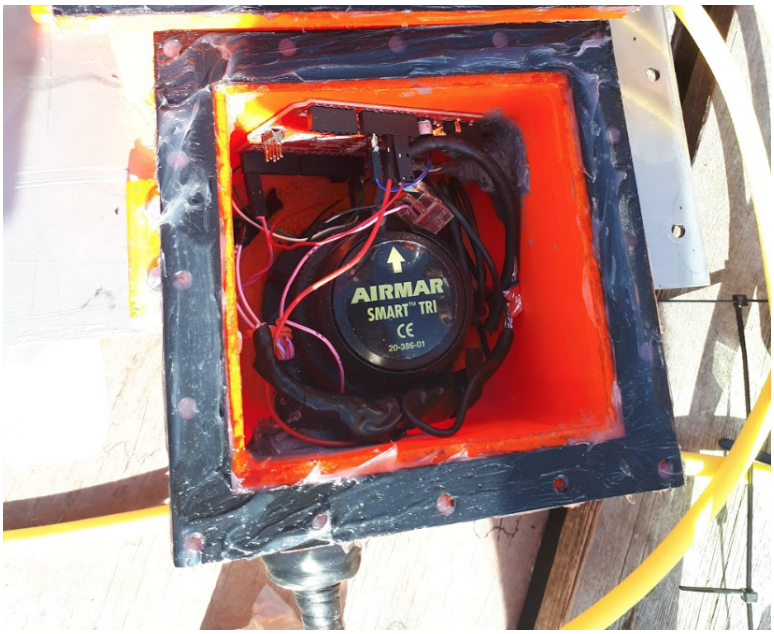


Figure 4.18: Echo sounder box

4.1.8 Mounting of cable

To mount the cable to the ROV, a mounting solution had to be made. This is done by bolting two metal plates to the upper front of the frame. The cable is then mounted to the plates by using a cable thimble and a shackle. This is demonstrated in figure

Figure 4.19: Mounting of cable

4.1.9 Instruments

Numerous sensors was installed in the ROV to control the depth, analyse the ROV behaviour and orientation. These sensors are the foundation of the control system and are necessary to make the ROV work as intended.

IMU

The Adafruit IMU using the accelerometer and gyroscope is used to calculate roll and pitch of the ROV. The necessity arises to use sensor fusion to perform the calculations as a gyroscope by itself will drift over time 2.5, and an accelerometer is sensitive to noise and vibrations 2.4.
Based on the objective of the task to have automatic control of an underwater vehicle, it is essential to have an overview of the pitch and roll of the ROV.
The IMU is powered by 5 volt supply and connected to the main I2C bus and accessed by the Raspberry Pi directly as explained in 3.9, hence eliminating the need for "intermediates" and enables faster readings.
The rate of readings/g-forces performed by/on the module can be adjusted using the registers of the target sensor. For more information see the datasheet of module (*Adafruit 9-DOF IMU Breakout* 2018).

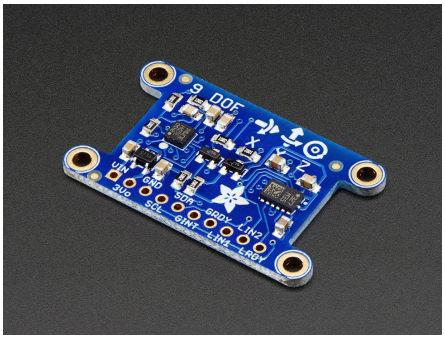


Figure 4.20: Adafruit breakout board 9DOF
Source: Adafruit.com

Pressure sensor

The pressure sensor shown in figure 4.21, is used to measure pressure and calculate the depth of the ROV with the use of equation 2.36. Furthermore, it measures the temperature of the surrounding water. It is necessary for the control system to know the depth of the ROV to use it as feedback for the controller.

The sensor operates on 3.3VDC and communicates over I2C. It were connected to an intermediate Arduino over I2C, implementing the MS5837 library see(sec:3.7). The sensor had to be calibrated before use in order to get correct readings. The sensor is calibrated by writing to the sensors I2C register as seen in 3.9. The correct placement in the register and bit-order was found in the datasheet from the manufacturer. (*MS5837-30BA* 2015)

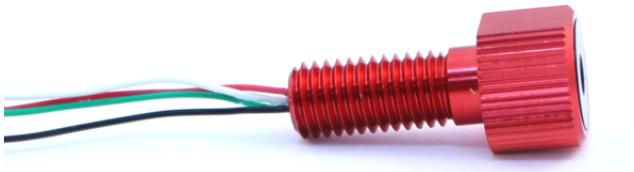


Figure 4.21: 30Bar High-Resolution Pressure Sensor

Source: BlueRobotics.com

Echo sounder

Airmar-DST800 NMEA0183 is the echo sounder used on the ROV and the surface vessel. This echo sounder can measure speed, water temperature and depth. It can measure up to 100 meters of depth below it with a measuring angle of 22°, and has a update rate of 1Hz. In the ROV this echo sounder is used for the following two things: finding the depth beneath the ROV and finding the speed of which the ROV is travelling at. This data is then used in the control system of the ROV, feeding it with the ROV's speed and depth below it.

The echo sounder is connected to an Arduino pro mini through an opto-isolator. The opto-isolator is a requirement for connecting a NMEA device to any computer or microcontoller and help avoid creating any ground loops in the NMEA network. The Arduino then gets NMEA sentences from the echo sounder and deciphers the data with the NMEA library for Arduino 2.6.5. (*Airmar-DST800* 2013; *The NMEA 0183 Information sheet* 2015)

Figure 4.22: Airmar-DST800

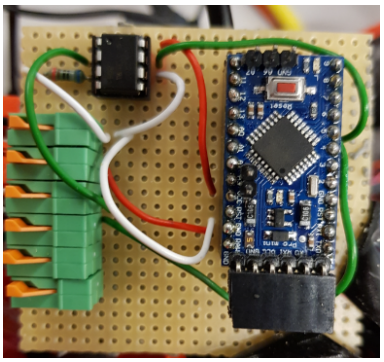


Figure 4.23: ES Arduino and optoisolator

Leakage Sensor

The leakage sensor were implemented in the camera housing as a safety measure, as this box is not filled with oil like the rest. It is connected with a 5 volt supply ground and signal and is required to have connections to four sponge probes.
The host board uses three sponge-tipped probes to check for leakage and if one is touched by a liquid, a signal will appear. (*SOS Leak Sensor* 2018)

Figure 4.24: SOS Leak Sensor
Source: Bluerobotics.com

Camera

A Raspberry Pi wide angle camera is used for capturing video images in front of the ROV. The camera is a type of fish eye raspberry camera and is connected directly into the raspberry pi. The fish eye lens enables a very wide angle to be recorded which makes the camera ideal for covering a great area in front of the ROV.

The camera is more compact than a standard web camera and does not need additional drivers to function. The small size of the camera can also be seen in figure 4.25 with a thumb as a reference.

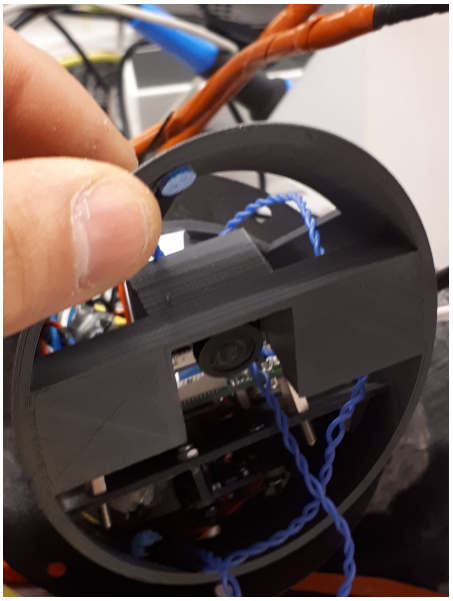


Figure 4.25: Fish eye camera

Circuit Boards

Several circuit boards were made to hold small components such as DC/DC-converters, connectors and etc. These were cut and soldered by hand and are used different places throughout the ROV. The circuit boards were covered in non-conductive epoxy on the underside to prevent short circuiting if they were to come in contact with other components.

4.1.10 Instrument disposition

The instruments placed in the ROV were divided into several housings. This as to maintain a modular design which was set as an objective; this also makes the prototype more redundant since if a leak occurs, it wont affect the whole ROV.

One of the housings is for the Raspberry Pi, the camera, IMU and the leak sensor. This housing is in the front of the ROV with a acrylic plate with a 45 degree angle. This is because it makes it easier for the camera to be able to capture images both forward and downward. It also has a frame inside which the raspberry, IMU, leak sensor and camera is mounted on. This makes it easier to take out the components as all you need to do is take out the frame.

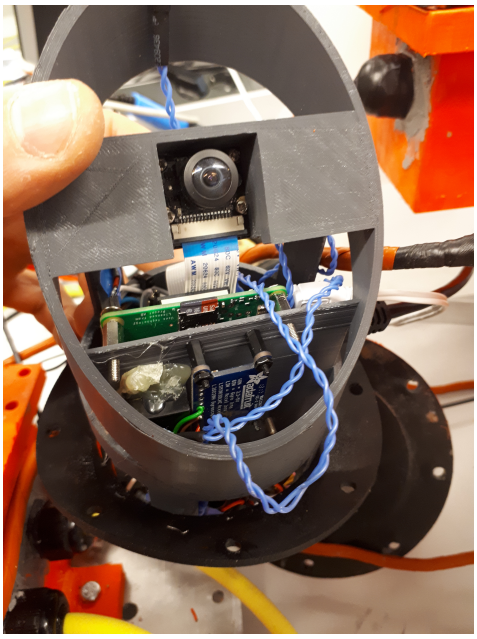


Figure 4.26: Camera houseing front

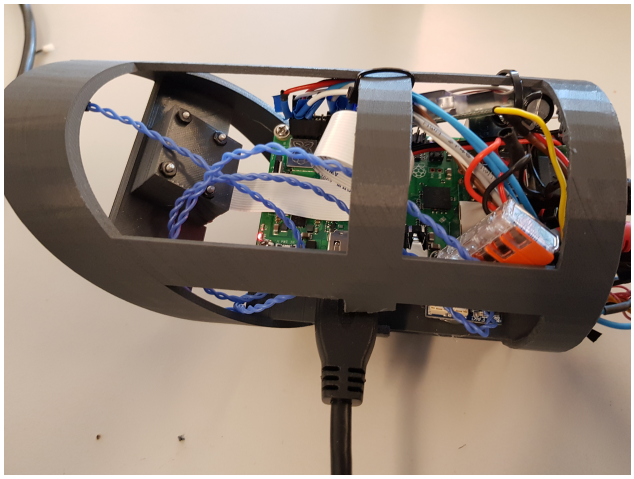


Figure 4.27: Camera housing

There is one box for most of the electrical hardware. This contains all the circuit boards of the ROV and Arduinos. It is placed in the lower centre of the ROV.

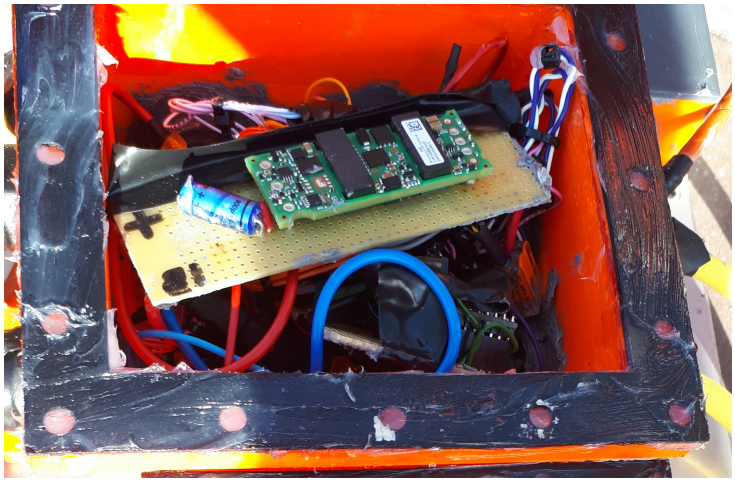


Figure 4.28: Electrcial box

In the lower back, the echo sounder box is located. The box has a hole in the bottom of it to mount the echo sounder and to two holes in the side for the connecting cable and the pressure sensor.

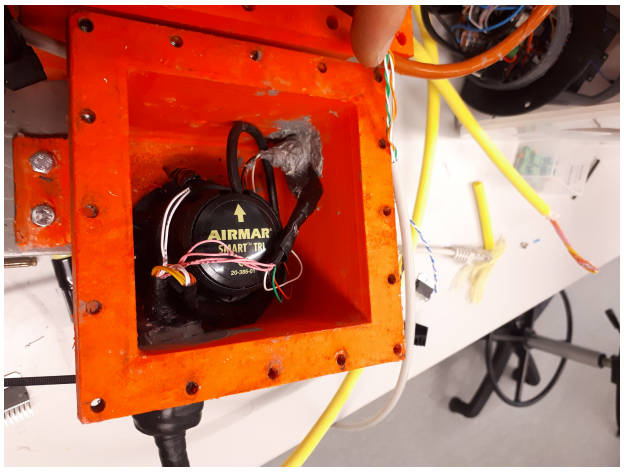


Figure 4.29: Echo Sounder box

There is also two identical boxes for each of the actuators. These are placed in the upper part of the ROV.

4.1.11 Microcontrollers

The prototype utilises three different kind of microcontrollers, the Arduino Mega 2560, Arduino Uno and the Arduino Pro Mini. They have the same clock speed of 16MHz, and supports digital, analogue and PWM inputs and outputs, but the Mega has more memory and RAM. The Arduinos were chosen since they were available at the university, and because the group was experienced with using them.

The Arduinos have different tasks, one of the Megas were dedicated as the PID-controller, while the other did the interpreting from the echo sounder and GPS on the surface vessel. The Arduino Uno was used to read data from the depth sensor. The two Pro-minis served as NMEA parser for the ROVs internal echo sounder and the other was used as an input/output device, for easily adding extra I/O devices such as extra sensors.

Several microcontrollers were used as the sensors are placed in different compartments and there was a focus on minimising cables between the boxes. Each microcontroller served, which ensures high cohesion, hence resulting in a fast update speed for values, giving a good real-time system. Having one task pr. Arduino did also make troubleshooting easier, since the Arduinos had fewer code lines and modules executing. Lastly there are also fewer devices and wires connected to each Arduino.

4.1.12 Onboard Computer

The Raspberry Pi 3 Model B+ was chosen as the ROVs onboard computer because the ROV don't have a lot of space and the Raspberry Pi have a small footprint. Another reason for choosing the Raspberry was the possibility to integrate the small sized raspberry camera and the amount of I/O pins, supporting digital, analogue and several I2C connections. The Raspberry Pi Single-Board Computer (SBC) was provided by the university which was in stock.

This Raspberry runs on the newest version of the Raspbian OS (Stretch). It is connected to Ethernet through a cabled connection and is powered through the 5V converter in the ROV. These specifications together with the raspberry's small size and combined with the camera input provides a good setup for the onboard computer.

Figure 4.30: Raspberry Pi-Model 3 B +
Source: Raspberrypi.org

4.1.13 Cable fitting

Cables are running to all of the boxes, this is the same type of cable that is running between the ROV and the surface vessel. There were drilled holes in the boxes, and PG cable glands (figure: 4.31) were screwed into the holes to prevent leaks where the cable enters the compartment. Additionally, the glands were cemented with epoxy on the inside.

The glands were greased with silicone on the rubber seal, before entering the cable. The cable was stripped, vulcanised and greased with silicone before entering, then fastened using the tightening nut. Afterwards the cable were glued into the cable glands from the inside, and the glands were vulcanised on the outside to prevent water entering as a last measure. Every box except the camera house was filled with mineral oil to prevent water from entering. The high viscosity of the oil will also prevent it from leaking out.

Figure 4.31: PG Cable gland
Source: Ahlsell.no

As the linear actuators were not water resistant, they were placed inside water resistant compartments. To connect the actuators to the air-foils, the actuators had to be attached to shafts exiting the compartment through PG cable glands to the air-foils. These glands were greased with silicone on the inside, and tightened around the axle to prevent water from leaking in.

Figure 4.32: Actuator Boxes

4.1.14 Power management

To ensure that the ROV were supplied with enough power, the maximum power consumption of each component was found in the data sheets and added together (3.4.7). This showed that the absolute maximum consumption would be 14.7A at 12V. The actuators and the echo sounder were the most power consuming components, according to the data sheets (*Airmar-DST800* 2013; *GLA750-P 12V* 2018) the actuators could use up to 3A each, and the echo sounder 5A. This was the worst case scenario and were accounted for.

The fluctuating consumption of the actuators and echo sounder had to be considered. The actuators draws a different amounts depending on the load and speed, and the echo sounder will differ depending on the depth. In table 4.1 it shows that the echo sounder only uses 200mA,

however this is only the result from one test (4.5.4) where the depth was at 1.4m. It also shows that the LED lights only used 200mA, which is much lower then calculated.

Table 4.1: effective consumption 12V

Component	**Current**	**Power**
Actuator x2	2.4A/3A	28.8W/ 36W
Raspberry Pi	400mA	4.8W
LED Lights x4	200mA	2.4W
Arduino Mega	50mA	0.6W
Arduino Mini x2	100mA	1.2W
Depth sensor	1.25mA	0.015W
Leak sensor	10mA	0.12WW
Echo sounder	200mA	2.4W
Total effective consumption	3.36A/4A	40.3W/ 37.5W

On the background of the calculations and tests seen in the tables above, a 120W DC/DC converter was chosen as the voltage converter in the ROV. This could be done as the power consumption were lower than expected when tested, even though resistance were higher (6,1Ω). This lead to an updated voltage drop table being drawn.

Table 4.2: Voltage in cable

Supply Voltage	**A**	**ΔV**	**Effective Voltage**
24V	2	12.2	11.8
36V	1.33	8.11	27.9
48V	1	6.1	41.9

The converter had an input voltage ranging from 18-72VDC, which was useful due to this voltage drop. 36V were therefore chosen as supply voltage as it met the criteria as well as it were simple to implement. Three car batteries coupled in series were needed for this.

There are two different voltage levels operating within the ROV, these are 5V and 12V. Where the 5V supplies are Raspberry, Arduinos and different sensors, and the 12V supplies are actuators, echo sounder and LED lights. Capacitors were placed in the circuit at the input of the DC/DC converter and before each component to filter out noise and avoid sudden voltage surges. In addition to this, the input of DC/DC-converter is protected by 5A blade-fuses. Reverse voltage

from the actuators was a problem which resulted in other components losing power for a second and made the raspberry reboot; to counter this, a zener-barrier which lead the reverse voltage from ground to plus were placed at the input of the motor drivers. Additionally, the power circuit for the actuator were separated into its own circuit with an extra 12V DC/DC converter, resulting in any remaining spikes would not effect the other components. The full power circuit of the ROV can be seen in appendix E.

4.1.15 Motor Controllers

In accordance to section 3.4.4 in method, motor controllers were needed to drive the actuators. First homemade motor controllers were made with the use of positive and negative MOSFET-transitors, coupled in the way of a H-Bridge, seen in figure below 4.33

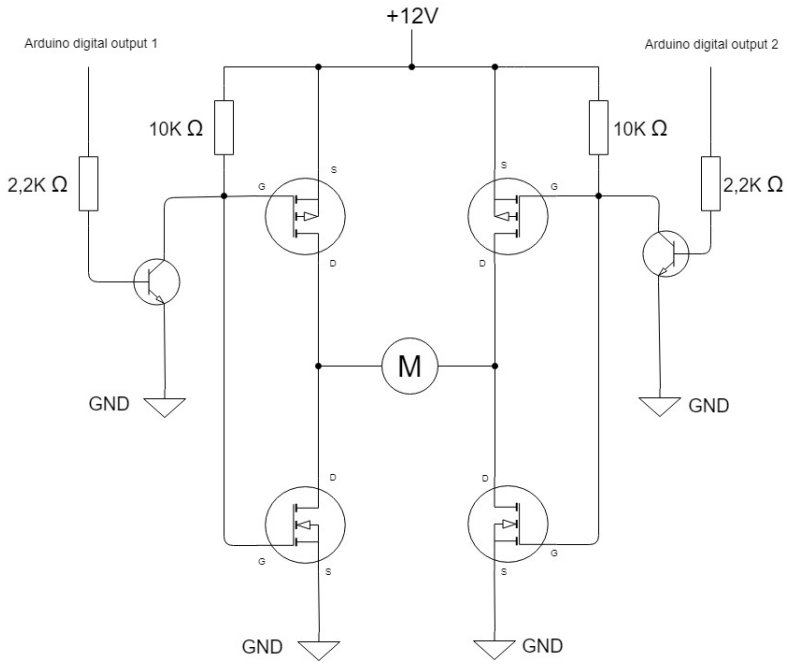


Figure 4.33: MOSFET H-bridge

The H-bridge were connected using a circuit-board and components were soldered by hand. The h-bridge worked as intended and could control direction and PWM voltage which adjusted the speed. The problem arose when stopping or changing polarity, as it created a reversed power spikes (mentioned in 4.1.14) that restarted the DC/DC converter.

The self made drivers were replaced with the L298 motor driver (shown in figure 4.34) and zener-barriers were added. However, this did not take care of the reverse power from the actuators, and continued to make the 12V DC/DC converter restart each time a reversed voltage spike

occurred. This resulted in all the hardware losing power and restart. Counter measures where tried such as common ground for the 5 and 12V, designing a low-pass filter and discharge capacitors. Afterwards, the H-bridges were changed in favour of a motor driver with built-in reverse power protection. Additionally, the power circuit for the actuators were connected to a separated DC/DC converter as not to affect the other electronic components.

In a final attempt to solve this problem, 2 JRK 21v3 motor drivers were purchased. These motor controllers are equipped with reverse voltage protection, a PID interface and several communication protocols including USB, logic-level serial and analogue voltage. To fit the rest of the setup, the motor controller PID was disabled and controlled the direction of the actuators using logic-level serial communication. By utilising the implemented features that allow the users to limit the motor controllers duty cycle and max acceleration, the actuators were secured from overheating and smoothly accelerated and braked. This solution removed the reversed voltage spikes, making the actuators work as intended.

Figure 4.34: L298 Motor driver
Source: Robotshop.com

Figure 4.35: Jrk 21 motor driver
Source: Pololu.com

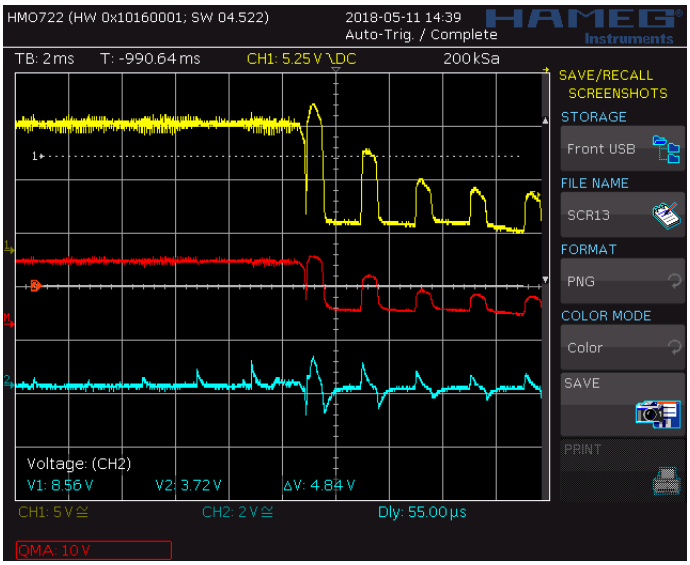


Figure 4.36: Voltage spike

4.1.16 Ethernet

Using four of the wires in the tether for Ethernet between the boat and the ROV as a 10/100 Mbps Ethernet solution did not work (*100 Mbps Ethernet / IEEE 802.3u including 100 Base-T* 2018). It did not work due to the cable not being a twisted-pair cable, resulting in interference from the DC power, and crosstalk between the RX and TX pairs (*Ethernet Cable Types, Performance & Pinout: Cat 5, 5e, 6, 6a, 7, 8 etc* 2018).

To actually get Ethernet to the ROV the Fantom-X-3 tether interface (3.6.5) was used, this interface carries the signal on top of DC power and managed to make a high speed connection between the ROV and the surface vessel, by using only one pair in the cable, and two Fanthom-X3 modules with a RJ45 plug and power input.

4.1.17 Mounting of Equipment in Prototype

Having a user friendly and easy to use prototype was important, with no need of any expert background to operate, maintain it or install extra sensors. This proved difficult in some places with many components and limited space. Effort were made to keep the overview as good as possible. This include twin power cables, cable ties, fastening of hardware, stacking of equipment and retractable frame. The mounting solution can be seen in figures: 4.37, 4.38, 4.39 and 4.40 below.

As an underwater ROV needs to be stable, the centre of mass needed to be as low as possible and

as centred as possible. The boxes are therefore mounted furthest down, with weight distributed along x-axis, as they are filled with oil they provide some ballast for the ROV. The actuators weigh 1 kg each and placed as low as they can be, but still too high as they provided instability. Therefore weights in the form of limb parts has been added at the bottom, which can be adjusted with more load for stability and ballast.

Figure 4.37: Equipment in boxes

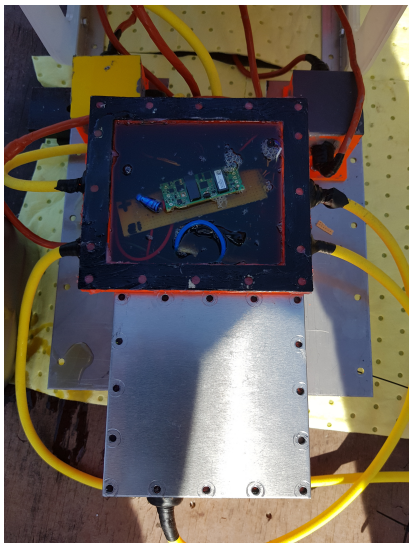


Figure 4.38: Boxes filled with oil

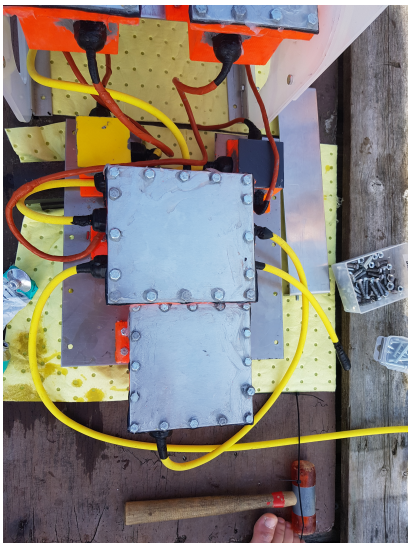


Figure 4.39: Boxes sealed shut

Figure 4.40: Boxes with equipment placed in ROV

4.2 Surface vessel

In addition to the prototype, equipment and housing is needed on the surface vessel, these have to be securely mounted as well. The solutions for these problems are described in this chapter.

4.2.1 Instruments

On the surface vessel there is a Adafruite Ultimate GPS Brakeout Board which is used to track the position of the boat and the ROV. It has an accuracy of about 3 meters which is standard for commercial GPS technology, velocity accuracy of 0.1 meters/s, and the update rate can be chosen between 1 - 10Hz. In addition to this, it supports both 3.3 and 5V input voltage. The GPS is used to log the position of where video and sensor data is recorded, because of this a high update rate is not required and the update frequency is set to 1Hz. It can display a wide variety of GPS sentences, but since only location data is needed, it is set to display RMC and GGA sentences (more about NMEA sentences in section 3.1.1). The GPS unit has a build-in antenna, but for better reception and the possibility to have the module in a box, an external GPS antnna was connected to the module. The GPS unit is connected to the same circuit board as the echo sounder on the surface vessel, and they are served by the same Arduino mega. The GPS sends NMEA sentences to the Arduino which then get deciphered. (*Adafruit Ultimate GPS* 2018)

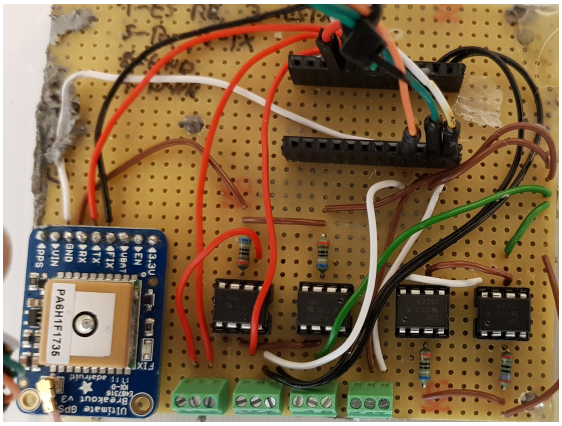


Figure 4.41: Adafruit Ultimate GPS

There were an echo sounder on the surface vessel, as mentioned in section 4.1.9. This was an Airmar-DST800 NMEA0183. This echo sounder is connected to the same Arduino Mega as the GPS, which parses the data from both devices and sends it to the GUI via a serial line. This echo sounder is fastened to the echo sounder bracket mentioned in section 4.2.4. If the boat towing the ROV have an NMEA0183 enabled echo sounder, this can be used instead of the second Airmar-DST800.

4.2.2 Surface Computer

On board the surface vessel there will be a client-PC. Any kind of computer would do, as long as it have one available USB and Ethernet-port, although a laptop is preferred. The computer will have the GUI installed and placed in a secure location away from any water.

4.2.3 Battery

Three 12V lead batteries was used for power of the ROV and surface equipment. Each of these batteries had a capacity of 75Ah. These three batteries were connected together in series to generate an operating voltage of 36VDC that were needed for the ROV. The echo sounder on the surface vessel (section: 4.2.1) needed 12V to operate, this was delivered by connecting it to one of the batteries.

4.2.4 Mounting Echo Sounder to boat

An adapter was made for fixing the second echo sounder to the boat. It was made from a flat double plate with a hole in, and square tubes in aluminium. The echo sounder was then mounted through the hole in the plate, and the adapter is fastened to the boat with clamps as seen in figure 4.42. There is no waterproof compartment protecting the echo sounder from water. However, this is not a problem since the echo sounder is waterproof itself, and it is not exposed to any pressure. This adapter is universally made, and is meant to be able to fit any small boats.

Figure 4.42: Mounting echo sounder to boat

4.3 Control System

This chapter describes how the control system is made, and goes into details about the modelling, PID-controller and the fuzzy logic.

4.3.1 Modelling

By using the estimated height and length of the vehicle and divide these by 2. The vehicle measurements then becomes: $a = \frac{0.54m}{2} = 0.27m$ $b = \frac{0.30m}{2} = 0.15m$ We assume that the volumetric mass density of the vessel is 1 until the design is finalised: $\rho \approx 1kg/m^3$
Using the equations from 2.17-2.31 we can obtain the necessary coefficients for the added mass derivatives:

$$m = \frac{4}{3} \cdot \pi \cdot 1 \cdot 0.27 \cdot 0.15^2 = 0.025447$$

$$e = 1 - \left(\frac{0.15}{0.27}\right)^2 = 0.69136$$

$$\alpha_0 = \frac{2(1-0.69136^2)}{0.69136^3} \cdot \left(\frac{1}{2} \cdot \ln\frac{1+0.69136}{1-0.69136} - 0.69136\right) = 0.50300$$

$$\beta_0 = \frac{1}{0.69136^2} - \frac{1-0.69136^2}{2\cdot 0.69136^3}\cdot \ln\frac{1+0.69136}{1-0.69136} = 0.74852$$

$$K_1 = \frac{0.50300}{2-0.50300} = 0.33601$$

$$K_2 = \frac{0.74852}{2-0.74852} = 0.59811$$

$$K' = \frac{0.69136^4\cdot(0.74852-0.50300)}{(2-0.69136^2)\cdot(2\cdot 0.69136^2-(2-0.69136^2)\cdot(0.74852-0.50300))} = 0.07361$$

$$I_y = I_z = \frac{4}{15}\cdot\pi\cdot 1\cdot 0.27\cdot 0.15^2\cdot(0.27^2+0.15^2) = 0.00049767$$

$$X_{\dot{u}} = -0.33601\cdot 0.025447 = -0.0085504$$

$$Y_{\dot{v}} = Z_{\dot{\omega}} = -0.59811\cdot 0.025447 = -0.015220$$

$$N_{\dot{r}} = M_{\dot{q}} = -0.07361\cdot 0.00049767 = 0.000036633$$

The added mass of the mathematical model was calculated, but after consulting with supervisors, advancement in the modelling process was abandoned. A more simple trail and error approach was chosen to design the controller and manage to complete the ROV in time.

To complete a simulation and get an indication of a proper tuning for the controller, the missing hydrodynamic coefficients were filled in with coefficients from another autonomous depth control system (Sujay D. Kadam n.d. Jan Petrich 2009). These values were altered to match the speed of the system, and restrictions like maximum wing angle and maximum actuator speed were added to make the system more accurate.

This resulted in a initial model for the ROV, which were simulated in Simulink shown in figure 4.43 and 4.44. A controller was tuned to match the requirements for this project and was implemented into the controller as a starting point. This would lay a foundation for further testing in the project.

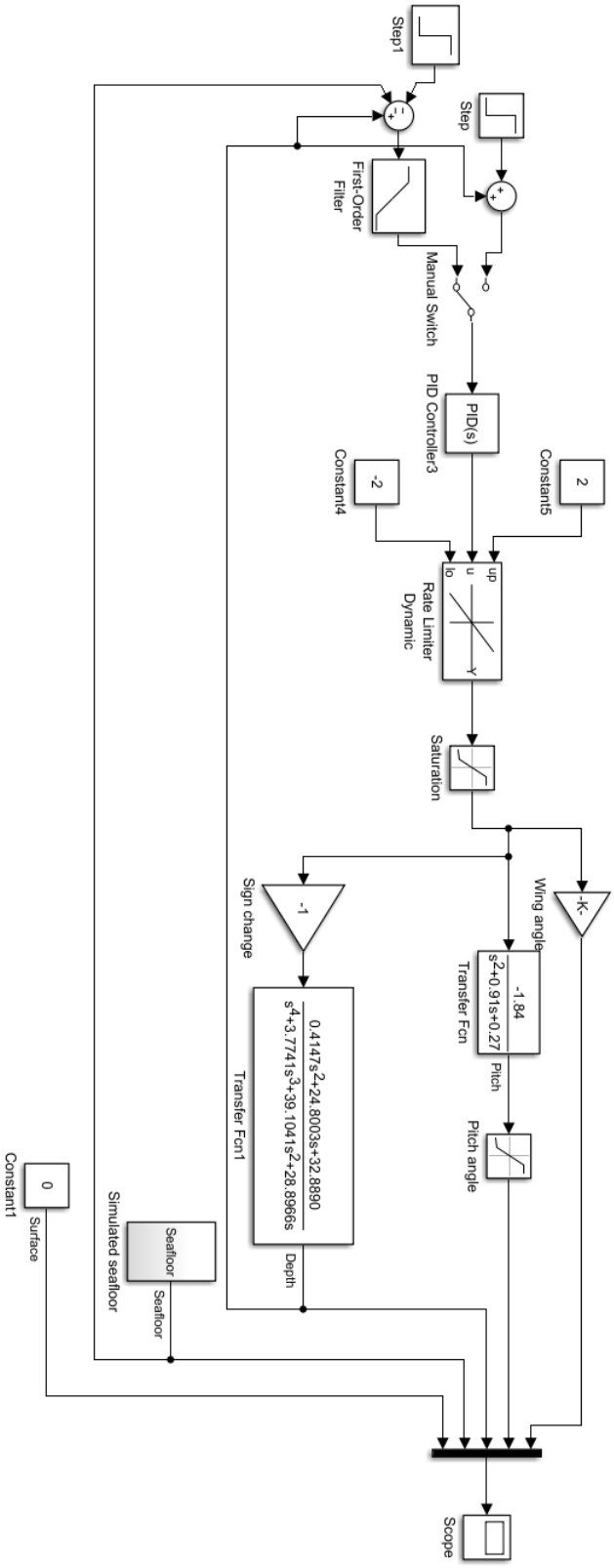


Figure 4.43: Simulink model

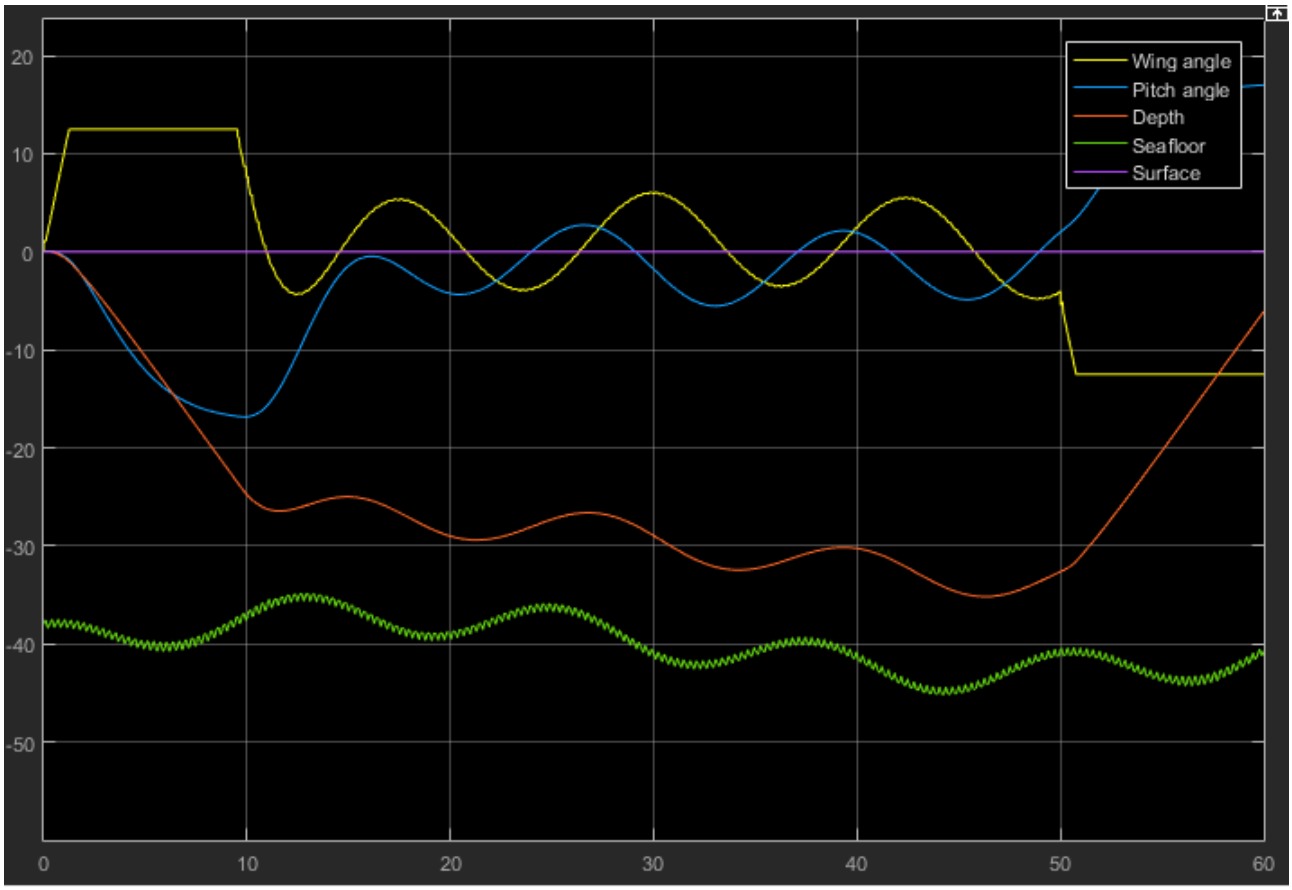


Figure 4.44: Simulink simulation

4.3.2 PID

The PID controller successfully calculated a position for the actuators based on the setpoint in the GUI and measured depth from the pressure sensor. An output is calculated based on the error values and the microcontroller runs the actuator to this desired position at full speed.

A hidden feature was added in the options to provide the fuzzy logic with appropriate gain values from the surface. This makes it possible to tune the controller without dismantling the ROV and uploading a new source code to the Raspberry pi. 9 gain values can be entered in the options file to change the low, medium and high speed values of the 3 gain values in a PID controller

4.3.3 Fuzzy logic

Fuzzy logic were implemented as a mean of providing varying degrees of gain to the PID controller, as such removing the need for multiple controllers when the speed varies. The fuzzy logic were implemented on the raspberry Pi using the FuzzyLite library for Java (tabel: 3.5). The

method used where of the Sugeno style inference, this were decided based on principles from Negnevitsky 2011, p. 4.6.2 that says Sugeno style inference is mostly used for dynamic non-linear systems. This inference works in the way of singletons and weighted average are obtained by using formula 2.11. For more information on Sugeno see section 2.5.6 The input parameters are classified using trapezoid and triangular functions as shown in figure 4.45.

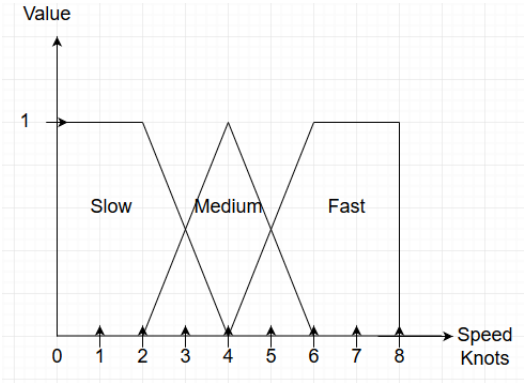


Figure 4.45: Input classification Fuzzy

The inputs are classified according to which speed the ROV is moving. This is the fuzzification process, the whole fuzzy inference system is illustrated in figure 4.46

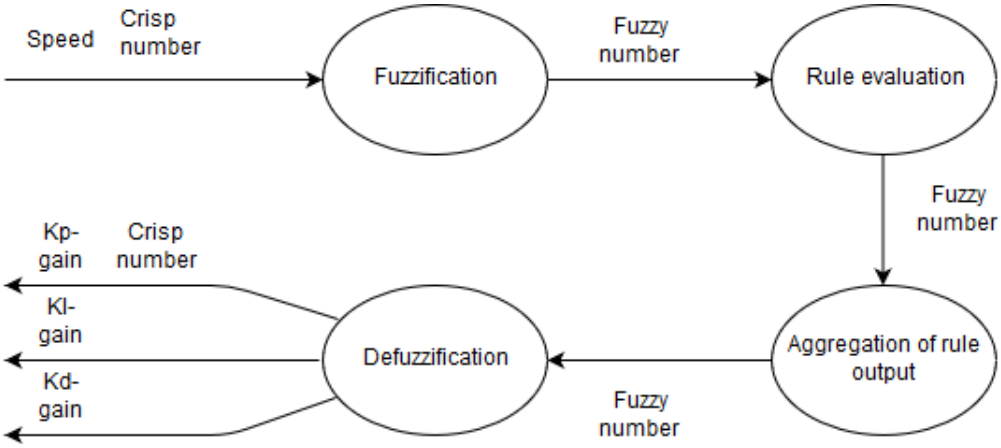


Figure 4.46: Fuzzy logic Sugeno

Speed is provided as an input to the fuzzy system, using the Sugeno style inference engine a fuzzy number is generated.
This number is then evaluated using the rule bases for kp, ki and kd given in equation 2.9. Each of these rule-bases will give an output, these outputs will then be subject to a defuzzification (equation: 2.11) process in which we obtain three output values. These values are the different gains for the PID controller.

4.4 Software design/solution

The solution for the software development for the ROV will be displayed and explained in this chapter. As mentioned in the method chapter, the software solution is split into two parts; the server application and the client (also known as the GUI), both these parts are described in this section. These solutions were implemented using Java, while every micro controller in use had Arduino C. The class diagrams and source code for the project can be found in appendix H and I

4.4.1 Dataflow diagram /dataflyt

Communication between the client-server, internal and corresponding components are shown in figure 4.47.

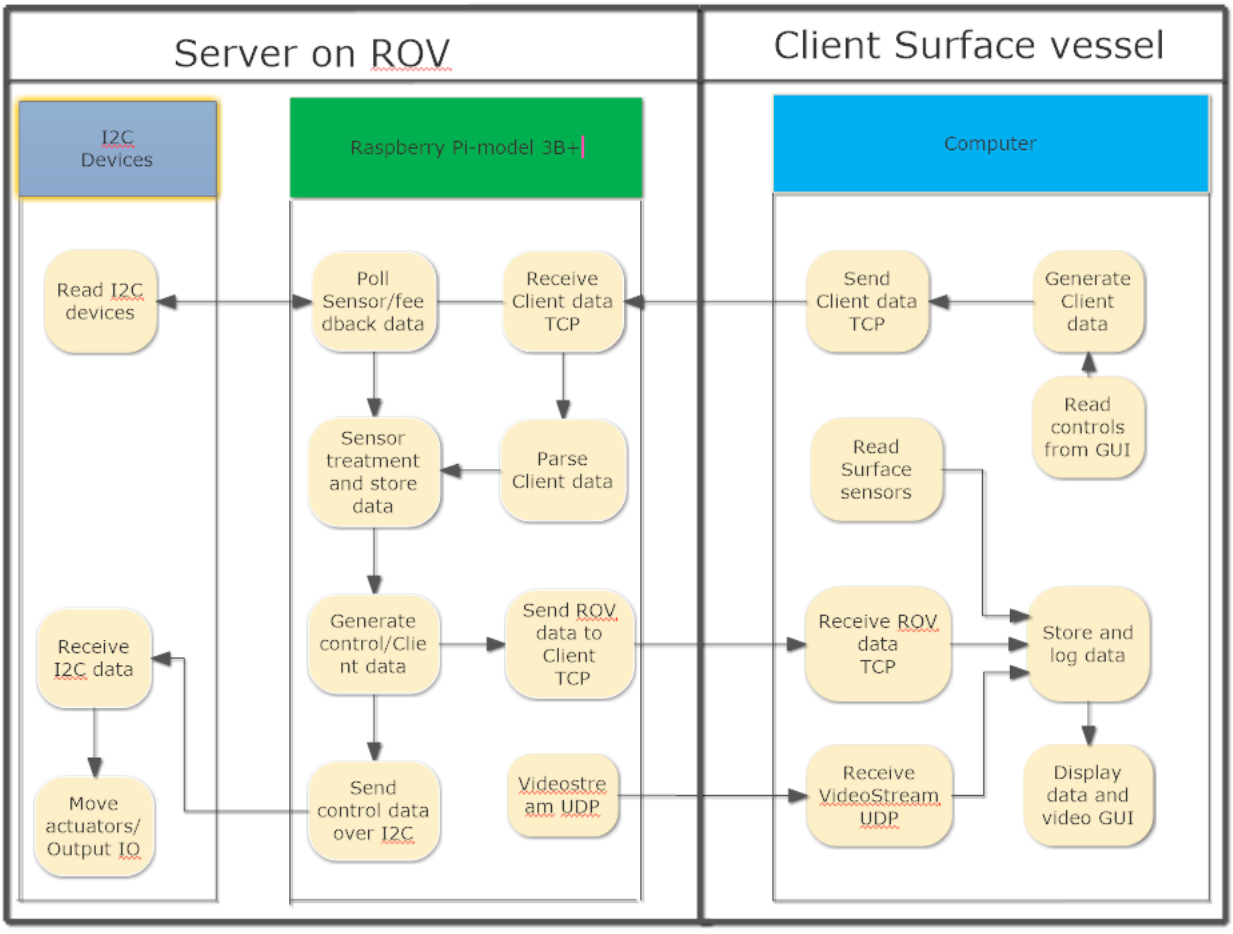


Figure 4.47: DataFlow

4.4.2 I2C and Serial Communication

As specified in section 3.5.5, two different communication protocols were used between the Arduinos and the Java applications. The first protocol being I2C were used within the ROV, as most of the sensors supported I2C and the distances were short. The other protocol serial communication was used both on the surface vessel and in the ROV. It was used as a parser for NMEA sentences using Arduino, and for the Arduino controlling depth. On board the surface vessel serial lines were used between an Arduino and the client computer. Transferring GPS and echo sounder data, this due to longer distances between the equipment, so I2C would not have been a sustainable option.

Internal communication in the ROV were solved using the I2C communication protocol (section: 2.6.1). As many of the sensors used integrated I2C protocol to transfer data these were connected directly to the Raspberry pi. The remaining sensors were installed with an Arduino as intermediate, and transferred to the Raspberry using I2C. This made hardware connections between components easy as they only needed to be connected using two wires of copper for the clock (SCL) and data transfer (SDA), and the required power (5V/3.3V) (section: 2.6.1).

All devices that were used had integrated pull-up resistors which made setup only point to point, instead of using external resistors on the two wires 2.12. Raspberry pi were set as master and would poll all I2C devices connected in turn, for retrieving and send values to them. As the pi were master it also set the clock speed, which were set static to 400kHz using command line in linux. The speed were set to a point all devices supported, and removed problems with clock-stretching (section: 2.6.1). By using I2C as internal communication all addresses had to be known to the raspberry pi, and link established between master and slave devices on the selected bus. By using the Pi4J library (table: 3.5) a bus instance would be created and an object link to the different devices set in the Java program. The GPIO pins would then be activated for I2C communication.

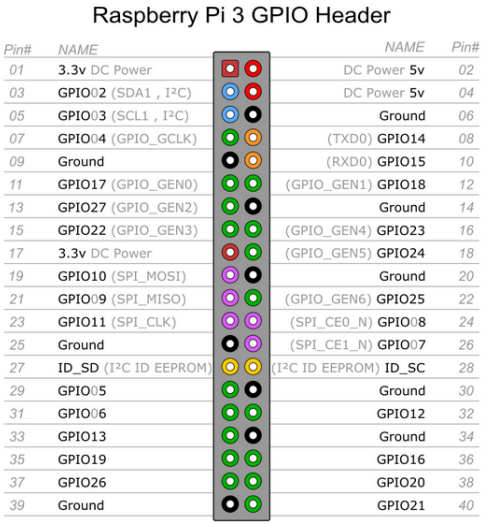


Figure 4.48: Raspberry GPIO
Source: Myelectronicslab.com

After the bus and devices were created communication can start, master polls all devices in turn and receives a set amount of bytes, either as an array or as single byte. The master (Pi) sends request for data and receives data, or sends data to the i2c devices.

Arduino uses the built in standard library Wire.h to communicate over I2C, by defining what wire to communicate over and set the slave address of the Arduino. By adding event based methods in the setup, these methods would be activated when master (Pi) polls the slave, who would either read/write data.

I2C only works with sending bytes, therefore all floats values in use had to be converted to byteArray before being sent. This where done using Arduino function called Union explained in section: 4.49. Where one float value and 4 bytes shared the same memory space.

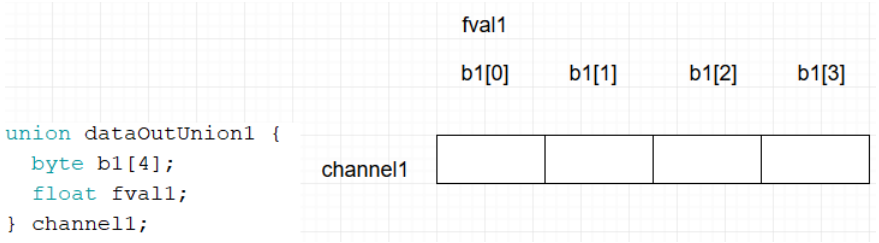


Figure 4.49: Union Arduino

There were problems using I2C as it were weak against noise when distance increased. This led to the echo sounder having noise that interfered with the connection between the depth sensor and the I2C bus. As this first started to have an impact late in the project, the noise were tried

to be reduced or filtered. These measures include; Low pass filter, shielded device and cable internal in box. The effort did not work and another solution had to be found for depth sensor.

The depth sensor were instrumental for the project and had to be working. Arduino with Bluerobotics library for I2C data retrieval (method: 3.7) were therefore placed beside depth sensors, this gave the desired values without any noise disturbing the connection. After this change values had to be transferred from Arduino to GUI and PID-control micro controller, this were done using serial communication. Sending float values as bytes (see 4.49) to the PID-control Arduino, then adding a float value to the regulator feedback that sent data to the SBC. From the SBC depth-data were transferred to the GUI for logging and visualisation, this is shown below in figure 4.50 .

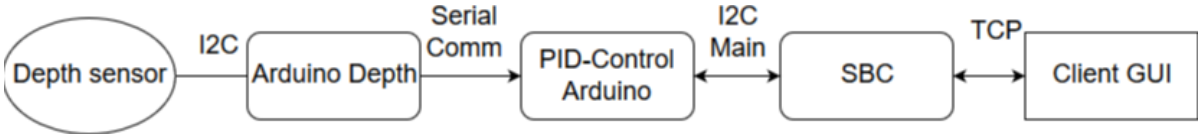


Figure 4.50: Depth sensor solution diagram

4.4.3 Client-Server Communication

The client-server communication is done through Ethernet via a cable running from the ROV to the surface vessel. Two different protocols are used for communication, one for video streaming and the other for data transfer. UDP is used to stream video footage from the ROV to the GUI on the surface vessel, and gave a fast video stream. Additionally it was not crucial if some video frames were lost. TCP is used for sending commands from the surface vessel to the ROV, and receive ROV data the other way. The data which is sent and received is crucial that are correct and in the right order, therefore a reliable communication protocol is used for this purpose. In figure 4.51, the implementations of TCP and UDP communications in this project is showed.

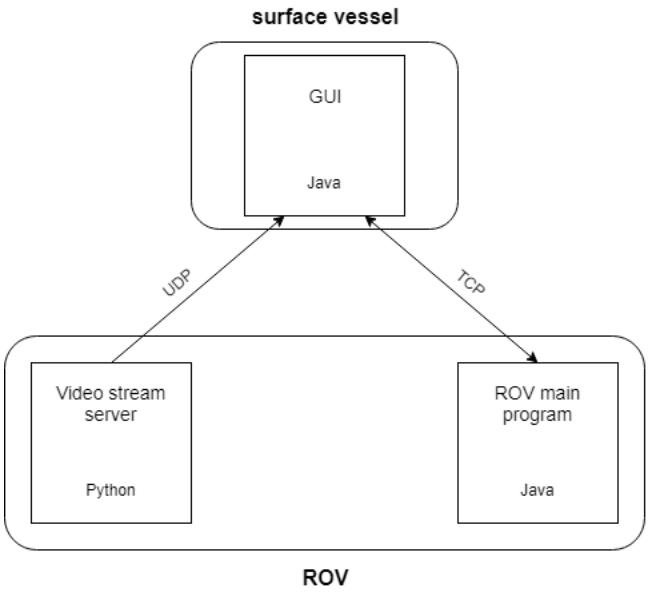


Figure 4.51: TCP and UDP communication

UDP communication

The UDP sender is implemented within the videoStream server on the Raspberry Pi for transferring images to the GUI, and makes for a smooth and stable video transfer. The function of the UDP stream is to stream video in real time to the GUI. A DatagramSocket is opened by the UDP server, and datagram packets containing the newest images are sent.

In the GUI it is implemented a UDP client for receiving the datagram packet containing the bufferedImage. This UDP client implements the Runnable interface, making it a thread which runs continuously and receives images from the Raspberry. The images are received through the UDP socket as a byte array and then converted back into a bufferedImage. The image can then be re-sized, making it larger, however, this can make the image blurry. This result in a solution which can give a large image frame and use as little resources as possible. The image can then be retrieved by anyone who needs it.

TCP communication

The TCP server is located within the main Java program on the Raspberry. This means that the external client must connect to ROV before communication between the two can commence.

The server on the Raspberry consists of a *connectionServer* and a *clientHandler*. Any request for TCP connections will be handled by the *connectionServer* which is a thread who always is listening for incoming connection from a client. When a client sends a request for connection,

the this class will create an instance of the *clientListener*, and pas along the needed parameters. The *clientHandler* is implementing the Runnable interface and is responsible for handling the TCP-clientsocket for the client connected to it. It monitors the connection and keeps it open as long as the external client is still connected, if not it will close the socket. It is this class which reads and writes to the client socket by using a BufferedReader and a PrintWriter.

The external TCP-client is implemented in the GUI. Through this clients commands and data are sent and received from the ROV. The client is left open until the program is terminated or the user shuts it down manually.

4.4.4 GUI-Graphical user interface

The graphical user interface is programmed in java using swing, and uses multiple threads to utilise the processor capacity properly. The GUI components works as intended like explained in section 3.5.3, and transmits commands safely using the TCP connection.

Users will be met with a screen for viewing the video feed from the ROV with a full screen button and an estimate of the horizontal distance to the ROV calculated with equation 3.3. The main frame also contains a text field to enter desired depth, buttons for selecting if depth is meters from seafloor or surface, light switch and an emergency button. On the right side of the main frame, the different sensor values are displayed along with the actuator duty cycle that notifies the user if the actuators may be overheating. The toolbar is equipped with connection buttons, other optional tools like echo sounder graphs, I/O controls and options. Additional safety features are implemented, like preventing the user from entering an invalid depth. A screenshot of the main frame is showed in figure 4.52.

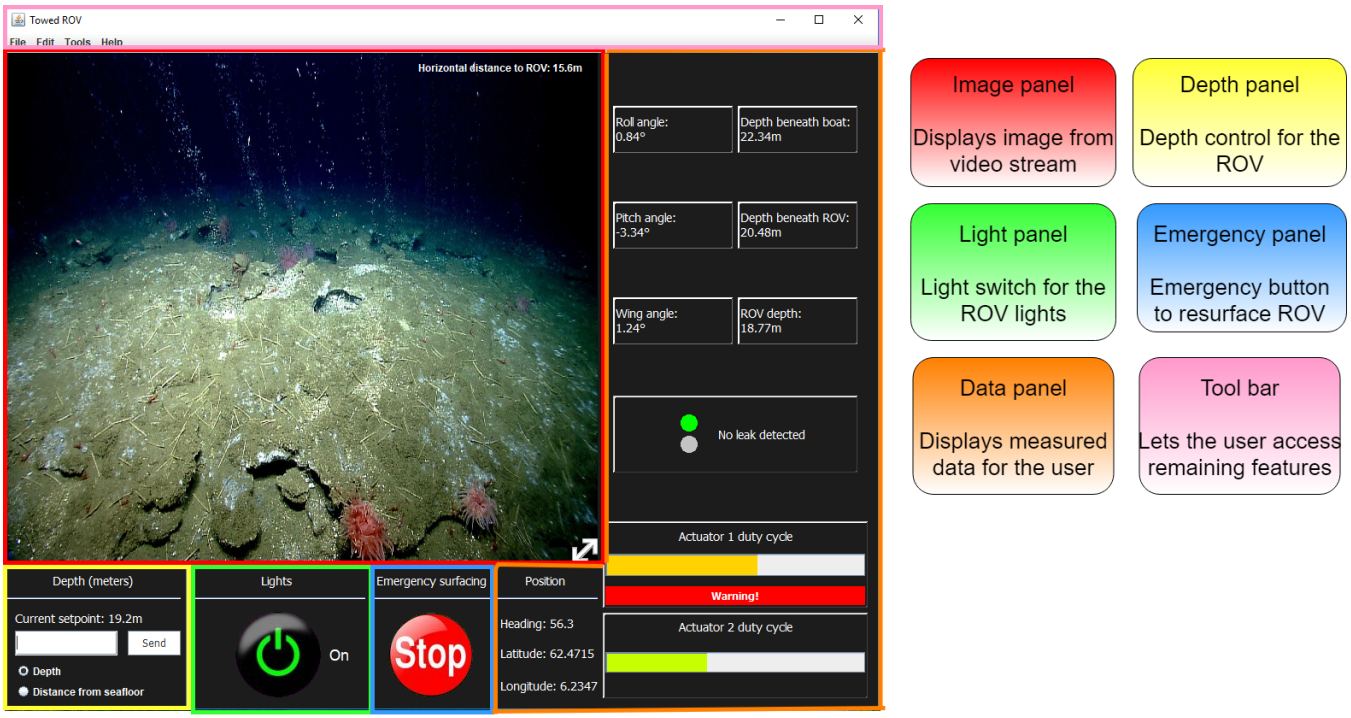


Figure 4.52: Towed ROV GUI
Video image source: Wikipedia.org

An echo sounder frame (shown in figure 4.53) can be opened from the tool bar, that displays the seafloor and depth of the ROV graphically. Calibrating the graph in the x-axis to match the exact position is not yet implemented, rendering the position of the ROV graph inaccurate.

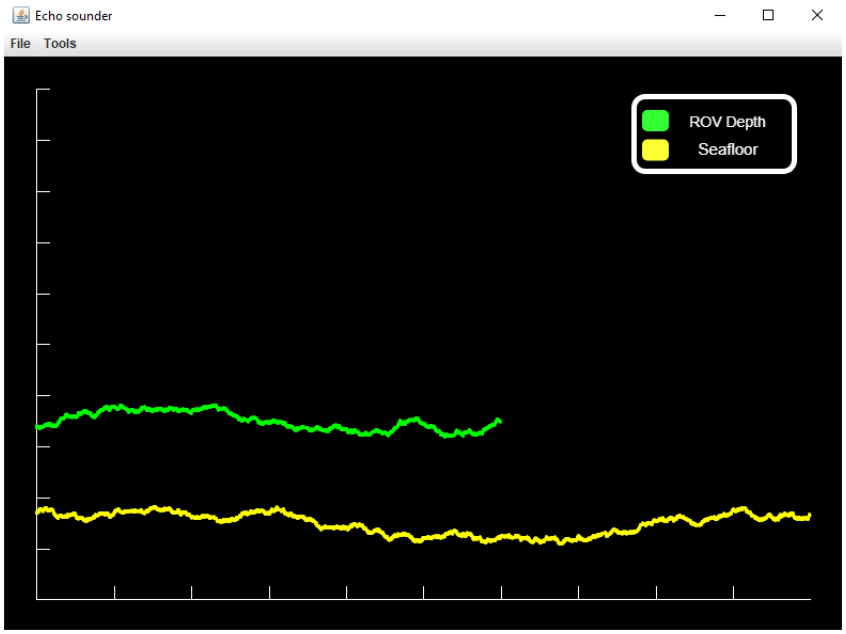


Figure 4.53: Towed ROV echo sounder

As one of the main objectives for the ROV was to create a multipurpose remote vessel platform, the option to mount sensors and different instruments was implemented. The ROV has the ability to implement new sensors in an efficient and easy way, without the need of an engineer. To prevent floating values when the sensor is disconnected and to control the digital outputs, the IO Arduino receives one byte of data where each bit indicates which sensors are active shown in figure 4.54 where 1 indicates on and 0 indicates off.

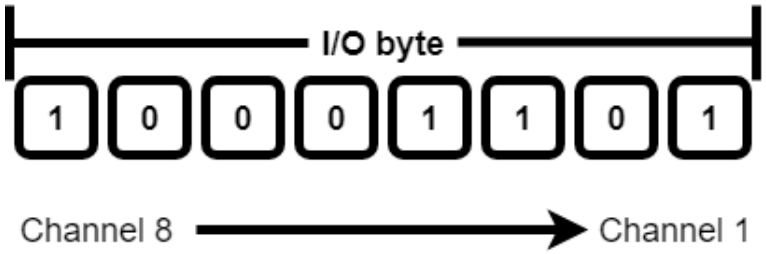


Figure 4.54: I/O Byte

The I/O controller frame (shown in figure 4.55) lets the user control 8 flexible inputs and outputs. The user can turn on and off 4 analogue inputs, taking into account which channels that have connected sensors. The corresponding sensor value will be displayed when the channel is turned on, and saved in the data log. 4 digital outputs can also be controlled in a similar manner to let the user control additional equipment. The channels can also be labeled according to which sensor is connected in the options frame. The

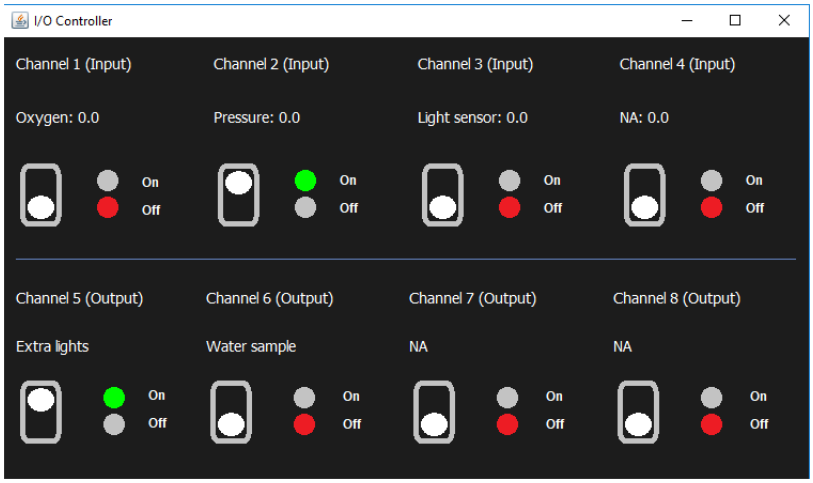


Figure 4.55: Towed ROV I/O

While the GUI is running, the software logs all the incoming data in a text file containing the start date and time of the measurement (Example of data log in figure 4.56). This makes it easy

for anyone to search and find data that might be considered useful later. Sensors can simply be removed and added to the log in the code of the dataLogger class. The video frames are also encoded into an MP4 file to make it possible to re-watch the video on a later occasion.

```
----- New session! -----

-----23.04.2018-----

              | Depth      | Heading      | Latitude      | Longitude      | Channel 1    | Channel 2     | Channel 3          | Channel 4
Time: 12:01:55 | Depth: 0.0 | Heading: 0.0 | Latitude: 0.0 | Longitude: 0.0 | Oxygen: 0.0 | Pressure: 0.0 | Light sensor: 0.0 | NA: 0.0
Time: 12:01:56 | Depth: 0.0 | Heading: 0.0 | Latitude: 0.0 | Longitude: 0.0 | Oxygen: 0.0 | Pressure: 0.0 | Light sensor: 0.0 | NA: 0.0
Time: 12:01:57 | Depth: 0.0 | Heading: 0.0 | Latitude: 0.0 | Longitude: 0.0 | Oxygen: 0.0 | Pressure: 0.0 | Light sensor: 0.0 | NA: 0.0
Time: 12:01:58 | Depth: 0.0 | Heading: 0.0 | Latitude: 0.0 | Longitude: 0.0 | Oxygen: 0.0 | Pressure: 0.0 | Light sensor: 0.0 | NA: 0.0
Time: 12:01:59 | Depth: 0.0 | Heading: 0.0 | Latitude: 0.0 | Longitude: 0.0 | Oxygen: 0.0 | Pressure: 0.0 | Light sensor: 0.0 | NA: 0.0
Time: 12:02:00 | Depth: 0.0 | Heading: 0.0 | Latitude: 0.0 | Longitude: 0.0 | Oxygen: 0.0 | Pressure: 0.0 | Light sensor: 0.0 | NA: 0.0
Time: 12:02:01 | Depth: 0.0 | Heading: 0.0 | Latitude: 0.0 | Longitude: 0.0 | Oxygen: 0.0 | Pressure: 0.0 | Light sensor: 0.0 | NA: 0.0
Time: 12:02:02 | Depth: 0.0 | Heading: 0.0 | Latitude: 0.0 | Longitude: 0.0 | Oxygen: 0.0 | Pressure: 0.0 | Light sensor: 0.0 | NA: 0.0
Time: 12:02:03 | Depth: 0.0 | Heading: 0.0 | Latitude: 0.0 | Longitude: 0.0 | Oxygen: 0.0 | Pressure: 0.0 | Light sensor: 0.0 | NA: 0.0
Time: 12:02:04 | Depth: 0.0 | Heading: 0.0 | Latitude: 0.0 | Longitude: 0.0 | Oxygen: 0.0 | Pressure: 0.0 | Light sensor: 0.0 | NA: 0.0
Time: 12:02:05 | Depth: 0.0 | Heading: 0.0 | Latitude: 0.0 | Longitude: 0.0 | Oxygen: 0.0 | Pressure: 0.0 | Light sensor: 0.0 | NA: 0.0
```

Figure 4.56: Towed ROV Data log example

The options frame displayed in figure 4.57 allows the user to customise the labels of each of the input and output channels. This makes it easy for the user to understand and remember what equipment is attached to the different channels while using the application and reading previous data logs. Additionally the user can specify the default IP to make it quick and easy to establish connection with the ROV. These options are saved on the hard drive and the current options to prevent the options from resetting each time the program is opened.

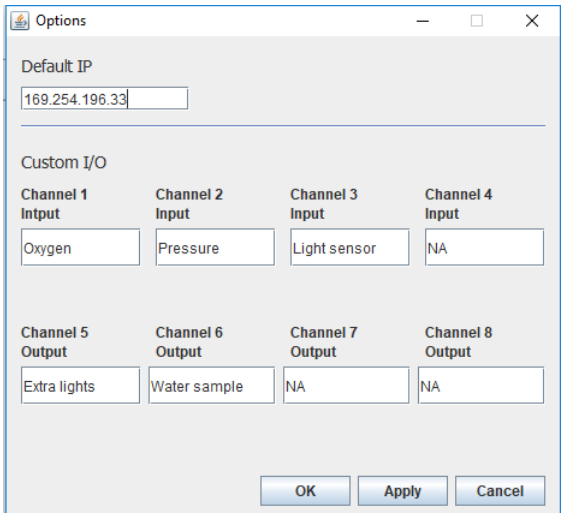


Figure 4.57: Towed ROV Options frame

The main problem encountered while programming the GUI, was the storing the video file on the hard drive. The library used for this could not keep up with the frame rate of the camera.

This resulted in the image buffer filling up and using increased amount of memory for each image arriving to the GUI. After running a while, the application would freeze because of a lack of memory. This problem was analysed using the Netbeans profiler tool to determine which variable types were increasing, and observing the memory using while commenting different variables and objects. The bug was fixed by removing the buffer and encoding images continuously as fast as the library could manage.

4.4.5 Server Application

The server application is the main program running on a SBC in the ROV, and consists of two sub-programs; a Java program and a Python program. The server application serves as the brain of the ROV. The java program consists of a TCP server, I2C-hub, Sugeno controller, treatment of data, controlling I/O's, and intermediate storing of data. The Python program have a single task, to stream video images from the ROV to the GUI. A class diagram over Java-ROV-application can be found in H.1 and a short summary of the classes are described below:

The ROV java application runs on multiple threads handled by the executor framework. The threads are set to run at specified time intervals, which releases processing power when threads are idle. The program starts upon booting of the SBC, and an object of all classes are created except *clienthandler*. The *ConnectionServer* listens on a serversocket for incoming connections on a specified port and creates the *clienthandler* upon client connection. The *clienthandler* is responsible for exchanging data between the client (GUI) and the server (ROV), see (4.4.3) for more information, principle figure below (4.58).

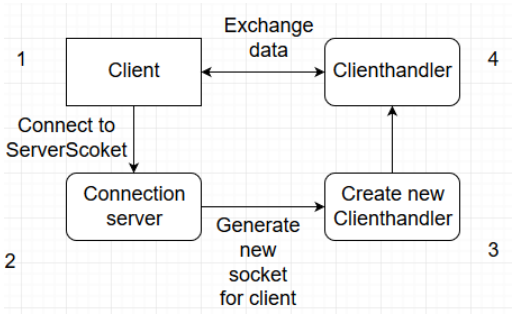


Figure 4.58: Client connection:Creating Clienthandler

When data has been received at *clienthandler*, it is transferred to *Parser* for deciphering. The Parser takes the input data and use a split function limited by control signs (<>), where first parameter is checked up against a switch case. Second parameter is the data which transferred

from the GUI, then placed in *StorageBox* from switch case. Figure below illustrates work-method (4.59).

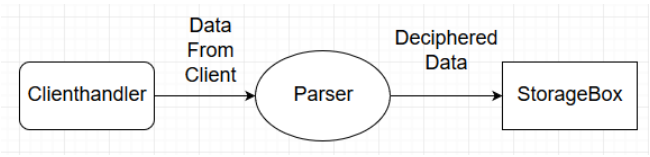


Figure 4.59: Parsing of data from client

The *StorageBox* uses set and get methods to distribute data, and only holds values and no tasks are performed except when called by other classes. Therefore the methods had to be protected using synchronisation, as multiple threads might try to access same methods. Principle of *StorageBox* is show in (4.60).

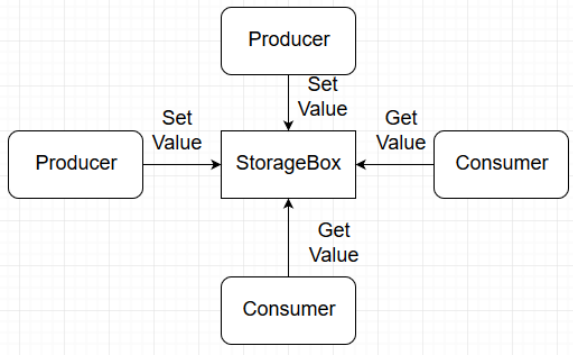


Figure 4.60: StorageBox:Producer consumer principle

FuzzyLiteController class is responsible for the implementing of fuzzy logic using the FuzzyLite library see (table:3.5). As the interference engine is created upon start of the server program for the ROV. The input parameters are set static as shown earlier in section: (4.45), and output parameters are retrieved from *StorageBox* when the client has sent them. Before this the class waits, after retrieved values are obtained, the output and rule-bases are generated into the fuzzy system. The system then runs as a thread with set intervals and stores the output gain values in the *StorageBox*. For more information on fuzzy logic see section: (4.3.3), for principle drawing see figure (4.61).

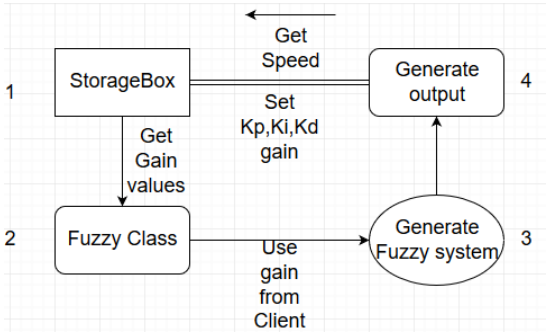


Figure 4.61: Fuzzy class overview

GPIO controls the lights and checks for leakage, the class runs as a thread and implements Observer pattern. The Observer changes it's output if the *StorageBox* get a new value from the connected client. The thread runs with set intervals and check if the leakage sensor input is changed, and send these values to the *StorageBox*. Interpretation shown in (4.62).

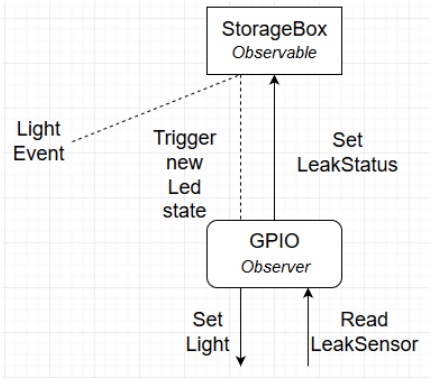


Figure 4.62: GPIO Overview with Observer

The *I2CBusComm* is responsible for the control of the main I2C bus, it creates a link for the bus and all connected devices, and the thread is set to run at 50 ms intervals. The class polls all devices for sending/receiving data and distributes data to preset destinations. This class extends Observer for notifying classes implementing the Observer pattern that changes has been made, creating event based action. For more information on I2C see section: 4.4.2. Explanatory illustration below (4.63).

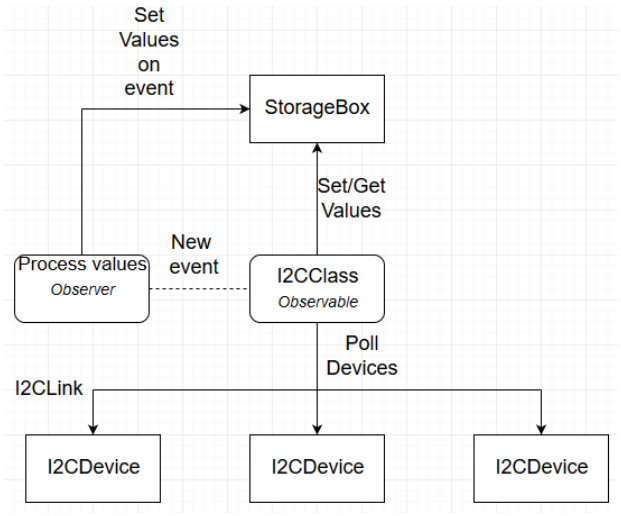


Figure 4.63: I2C class overview

For processing data from the accelerometer and gyroscope the *IMUValues* is used. Implementing Observer pattern as to remove the need of a running thread, notified only when a new value is retrieved from sensors. The class then process the data to generate roll and pitch, then store these values in the *StorageBox*. For more information on generating roll and pitch see section 4.4.6.

The *PressureValues* class was used for processing data from the pressure sensor, this class were not in due to any noise from the echo sounder described in section: 4.4.2. The class however is valid as it works and can be used if pressure sensor is moved away from noise. The class processed data from the pressure sensor, retrieved by *I2CBusComm* and implemented the Observer pattern. Acting on events and generating depth,pressure and temperature, then sending values to the *StorageBox*.

Video Streaming

The video streaming was achieved with a separate Python program. First it was intended to be done within the main Java code, but Java does not support filming with the Raspberry camera due to lack of libraries. The Python program named VideoStreamServer, captures images and streams it to the GUI through the UDP connection explained in section 4.4.3. A transfer speed of approximately 15 frames per second where achieved with this solution, and the available resources.

The program utilises the OpenCV2 and PiCamera libraries (table:3.6) to capture and modify the images. The PiCamera library contains functions for capturing the images and the OpenCV2

library contains functions for changing the images afterwards. The resolution, orientation and image compression can easily be changed within the python code. The images is being captured in HD quality, before being re-sized to 420p then sent to the GUI, this saves bandwidth in the Ethernet communication. The VideoStreamServer is located in the same folder as the Java code, and is also started during the start-up process by the start-up script explained in section 4.4.7.

To connect to the VideoStreamServer it is important to use the correct IP-address on the external client, the IP-address should be set to 192.168.0.20 for the GUI. This address is hard-coded into the program and therefore no video will be displayed if the IP-address is set to something else than this.

4.4.6 Sensor data treatment

In order to get relevant data from some of the sensors, the raw data had to be processed and filtered. The sensors that this is valid for are the IMU (accleromter,Gyroscope) and the pressure sensor. The original plan were to use the built in libraries from adafruit and bluerobotics, then use an Arduino as an intermediate to transfer the data to the SBC. This however failed as having Master-Master communication with Raspberry Pi is difficult. An attempt was made to use two I2C busses with the Teensy 3.5 as intermediate, where there were one bus from Teensy to sensor , then another from Teensy to raspberry. This communication worked, but were unstable and crashed after some minutes.

Solution were to implement parts of written source-code for java and use part of Arduino sensor libraries to create own application for sensor treatment.

Accelerometer

The accelerometer works as mentioned in section 2.4, the raw x,y,z values are then multiplied by the gravity of the earth and the sensitivity the accelerometer has been set to by the accessing correct I2C-register, see section 3.5.5. The raw values can after this be processed through filters, first being a low pass filter;

$$\begin{aligned} accel &= raw \cdot Weight + (oldAccel \cdot (1 - Weight)) \\ oldAccel &= accel \end{aligned} \tag{4.4}$$

Where:
accel is the vector value that will be used for calculations.

raw g-value in set direction from accelerometer
Weight degree of trust in new values from accelerometer.

The function of the low pass filter is to remove noise and sudden vibrations that can affect the accelormeter. Which would have made it unreliable and impractical to use for our application.

The second part of the accelerometer data process is to convert from x,y,z in g-forces to roll* and pitch* in degrees. This were done using the formula specified in equation 2.1 as shown below;

$$pitch* = \frac{(\arctan\left(\frac{accelX}{\sqrt{accelZ^2+accelY^2}}\right)) \cdot 180}{\pi} \tag{4.5}$$

Where:
pitch* is the pitch generated by the accelormeter
accelX/Y/Z are the accel vectors

Gyroscope

Gyroscope works as mentioned in section 2.3.5, the Gyro is initialised by the I2C master shown in section: 3.5.5, and calibration then ensues. The calibration calculate the min, max and mean value of the gyro which will be used to filter out irrelevant readings. The values are then retrieved in fixed time Intervals, where the raw values are multiplied by the set gyro sensitivity for correct values.

The gyro values are then processed through a low pass filter in the same manner as accelerometer was in section 4.4. The noise would then be further reduced in order for as accurate as possible readings. The next part were to use a complimentary filter to merge the two result for best accuracy as explained in section 4.4.6.

Complementary filter/Sensor Fusion

A complementary filter were implemented in order get accurate readings from the IMU, it is described in detail in (theory: 2.7.8). How we implemented this filter were in a similar manner;

$$\begin{aligned} pitch &= \alpha \cdot (pitch + GyroY \cdot (dT + \tau)) \; + (1-\alpha) \cdot pitch* \\ roll &= \alpha \cdot (roll + GyroX \cdot (dT + \tau)) \; + (1-\alpha) \cdot roll* \end{aligned} \tag{4.6}$$

Where:
α is the trust to be put into new data.
pitch/roll generated values
GyroX/Y generated degrees/s for each angle.
pitch*/roll* accelerometers pitch and roll.
dT is the time from retrieving data to calculating the result.
tau constant set to same time as gyroscope reading rate.

This lead to the system being resistant to noise and vibrations, while still being fast and keeping accurate calculations over time.

4.4.7 start-up scripts

A start up script is used in the ROV to start all necessary software for running, when the Raspberry is powered on. The programs initialised by the start-up script are the main ROV application and the video stream server, these are the two programs required to run on the ROV. The script was made as a shell script (.sh) and then added to the raspberry's /etc/init.d folder, where the start-up scrips are located.
Start-up script:

```
# !/bin/bash
cd /home/pi/Desktop/ROVApplication/dist/
java -jar TowedROV.jar &
cd /home/pi/Desktop/ROVApplication/VideostreamServer
python VideoStreamServer.py &
exit 0
```

Line two and four enters the correct folder containing the programs that shall be executed, and line three and five executes the programs. The ampersand most be added for the program to be run continuously. To make the script to run at start-up it had to be added to the raspberry's start-up sequence, this was done through entering the /etc/rc.local file from the terminal and adding the path of the script to the end of this file.

4.5 Test Results

This section shows the results from the various tests conducted. The first tests presented is of the ROV as a whole, and the latter tests of different equipment.

4.5.1 Testing in Tank

A test were done in the towing tank at NTNU Ålesund to check for leakages and the ROV's stability in water. It was performed by dragging the ROV by hand through the tank several times.

The test showed that the ROV was positively buoyant in water without the electronics and oil filling, so that extra weight had to be added underneath to decrease buoyancy and increase stability. Afterwards, the ROV manoeuvred fine in the water and achieved good stability; however, it also revealed leaks in every box except from the camera house (which was the only box that could not be filled with oil).

4.5.2 Dry Test

A dry test was conducted in the lab, testing all systems for functionality before the test in the ocean to see if the software and control system worked as intended. Additionally, the load on the wings were gradually increased by applying pressure against the direction they were driven, to simulate water resistance.

All implemented functions and features were first tested separately, before testing the whole system. Communication were examined by sending commands and receive sensor data, different gains were sent to the PID controller from the GUI, and the video from the raspberry successfully streamed to the GUI.

Autonomous depth control were evaluated by setting a depth in the GUI, then applying different pressure to the pressure sensor to simulate depth. This would make the wings turn upwards or downwards depending on the depth set and the pressure applied. The emergency surfacing button was also tested, and when pushed, it turned the wings to move downwards to generate lift.

The functionality for extra payload with inputs and outputs were examined by connecting different sensors to the the inputs and measuring the voltage on the outputs of the I/O Arduino. When sensors were connected and switched on in the I/O controller menu they showed up in the GUI and the values were logged in the data log. Additionally, the names of the sensors could be changed in the options frame and an indicator light would tell if they were on or off.

The video recorded throughout the test and was streamed to the GUI, where it was presented in a video frame in addition to being converted to a .mp4 and stored on the computer. The video was later watched and it has revealed a smooth and reliable video transfer without any

noticeable latency or the video freezing.

The permanently installed sensors were also tested: echo sounders, depth sensor, leakage sensor and GPS. This were done by placing the echo sounders in water, with a depth deeper than one meter, placing the GPS antenna in the window, put the pressure to different depths and triggering the leakage sensor. The echo sounders would then display the correct depth which was 1.4m. The GPS gave the correct coordinates, the depth sensor displayed the correct pressure and depth, and the leakage sensor gave a warning in the graphical user interface.

4.5.3 Ocean Test

The full test in the ocean was the final test and a milestone for the project. This test was done in Storfjorden, Glomset which at its deepest is approximately 680m. The ROV was assembled and filled with oil on-site; pads were used to absorb spilled oil, these were placed underneath the ROV. Silicone grease were applied to the gaskets and the boxes were sealed. The boat used for the test were a 17ft pleasure craft with an outboard motor, and it was conducted at speeds between 2-6knots.

When the ROV was assembled and ready for testing, a dry test on the docks were conducted to ensure all the systems were working as designed. The test revealed that everything was working, and nothing had been compromised when filling the ROV with oil.

After this, the ROV was lowered into the water from the dock to check its buoyancy and stability. It showed that with the electronics and oil it was negative buoyant, as intended to be, and the stability when laying still was good. No oil spills were detected, nor any leakage.

Finally, it was time for the main test from the boat. The ROV was transported out on approximately 15 meters deep waters. It was then deployed from the side of the boat and started, so that the sensors could be calibrated at the surface. The test was preformed at the surface by setting a low gain and the setpoint to 0m, and later a few meters under the surface; this was done to prevent collision before the controller was tuned. The settling time of the system proved to be approximately 14 seconds with an overshoot of 0%. The depth data from this test is shown in figure 4.64. This proves that the control system can be further optimised by increasing the gain to provide a faster settling time. Images showing the ROV and pictures from the ROV during the test can be seen in figure 4.65 and 4.66 below the graph.

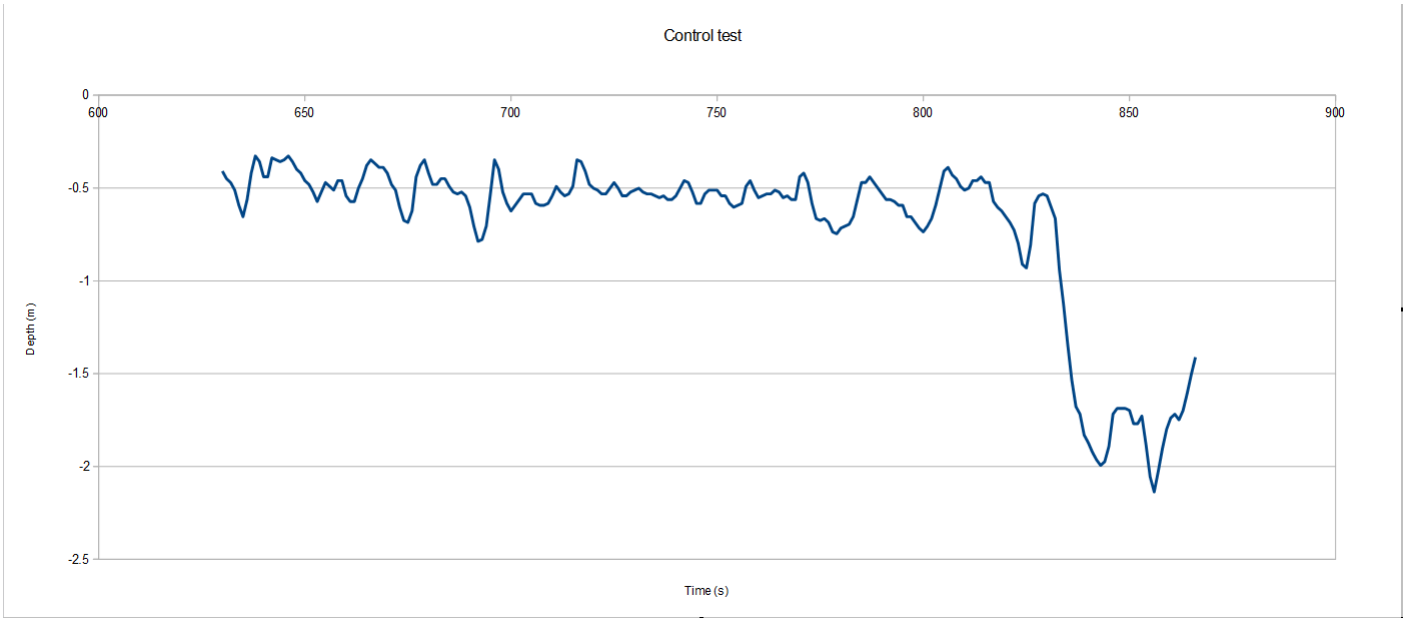


Figure 4.64: Depth test data

Figure 4.65: The ROV behind the boat

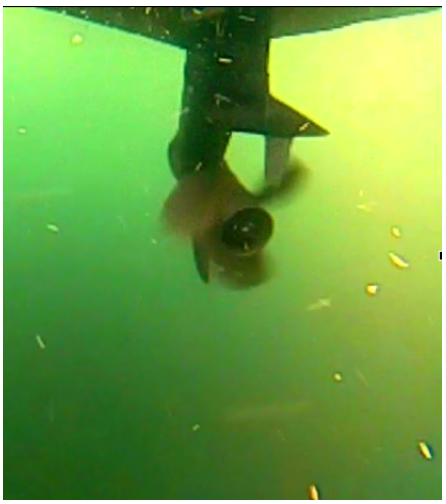


Figure 4.66: ROV camera capturing propeller

At this point in the test some problem were encountered, the video stream froze and we lost communication with the sensors. After getting the ROV onboard the boat again, video transmission was regained and it was tried to redeploy. However, the I2C bus did not come back online, and the leakage sensor in the camera house was triggered. After the ROV was back on the boat and a closer inspection was done, it was revealed that there was condensation in the

camera house, however, no leaks were found. There were no further testing after this since I2C is crucial for communication and get sensor data from the ROV and it was too late in the evening to start with troubleshooting.

As a additional note, the ROV maintained good stability in the water. With a maximum roll of 25°. Most of this roll were caused by surface disturbances and motor turbulence, but also by slack in the wings, the actuators not being entirely synchronised, and the shackle not being perfectly aligned.

4.5.4 Component tests

Test of different sensors and hardware.

GPS

Results from the GPS test in figure 4.67 shows that both the GPS module and the Arduino code for parsing the data works as intended. At latitude and longitude it displays 62.4715N and 6.2347E which is the location of NTNU Ålesund, the heading was controlled against a compass and also displays the correct values. Additionally, the time and date is correct. Speed and altitude is not correct in this image, but corrected itself after the GPS had been running for a while. It is not crucial that speed is correct since this is retrieved from the echo sounder and altitude is not used.

```
$PGTOP,11,3*6F
$GPGGA,102401.000,6228.2903,N,00614.0809,E,1,05,1.61,97.0,M,43.3,M,,*57
$GPRMC,102401.000,A,6228.2903,N,00614.0809,E,0.14,339.40,270218,,,A*6A

Time: 10:24:1.0
Date: 27/2/2018
Fix: 1 quality: 1
Location: 6228.2905N, 614.0809E
Location (in degrees, works with Google Maps): 62.4715, 6.2347
Speed (knots): 0.14
Angle: 339.40
Altitude: 97.00
Satellites: 5
```

Figure 4.67: GPS Test

Echo Sounder

Testing of the echo sounder was completed as described section 3.7.1. Figure 4.68 shows the results from the test, it shows both the NMEA sentences and the parsed data. The parsed data

```
$SDDBT,4.6,f,1.4,M,0.8,F*0A
Depth below Transducer:1.40
$SDDBT,4.6,f,1.4,M,0.8,F*0A
Depth below Transducer:1.40
$YXMTW,24.3,C*17
Temperature in celcius: 24.30
```

Figure 4.68: Echo sounder Test Results

shows that the echo sounder is capturing the correct depth and temperature of the water. Furthermore, there is one NMEA sentence not displayed in the test, which is the speed. This data is only sent from the echo sounder unit when it is moving, it was tested by mowing the paddle wheel under the unit and it worked, however, it is not displayed in the figure. Lastly, the power supply showed that echo sounder drew 2 amps.

Chapter 5

DISCUSSION

5.1 Prototype

5.1.1 Design and materials

Although the design of the frame is a good solution, it still has room for improvement. The front cover reduces drag, but changes can be made to reduce the drag further. The cover could have a smaller angle which would help decrease water resistance. In addition to this, the light boxes does impact the drag negatively as they are not flush with the plates in the front. By using other production methods than what has been used in this project, more complex shapes would be possible, like a more cylindrical or a foil-like shape. This would help reduce drag even more.

In retrospect, it is also clear that the frame of the ROV should be longer. The prototype is very compact, and it can be difficult to fit all the equipment. By making it longer, the fitting of cable and boxes would be much more convenient. Additionally, it would make the ROV more stable.

The shapes of the housings containing equipment are not optimal for sub sea operations as they have a cubical shape. Boxes are not suited for withstanding pressure, and will break easily at greater depths. A better solution would be to use a cylindrical shape for the housings. A cylinder can handle pressure a lot better than a box, and is therefore well suited for underwater usage. This could be done by using acrylic cylinders or casting them. By using casting methods it is possible to achieve a custom cylinder that would fit the needs of the ROV.

The ROV is also limited by the materials used. For the side plates and front cover it would be more suitable to use a stronger material than acrylic like composite plastic or nylon. As acrylic is quite fragile and stiff, there is a chance that it will break easily. By using stronger materials, this would not be a problem. By using other production methods, more complex shapes would

be possible like a more cylindrical or a foil-like shape.

The airfoils used are cambered, which provides lift for the ROV. These could have been replaced with symmetrical ones, as the cable already provides a lift force.

All the boxes which holds the hardware are 3D printed with PLA because of the cheap and quick production. Even though it works for testing, it will not work as a permanent solution as it is exposed to a lot of pressure, and it is not waterproof. Using composite materials or metal, would solve this problem. After coating the 3D printed material, the water resistance seemed to improve.

Another encountered issue, was the material warping when tightening it with bolts. This led to water leaking through the gaskets. Some of the boxes also cracked if tightening too much. This would not have been a problem with stronger materials. A good alternative would be aluminium boxes or acrylic pipes.

5.1.2 Hardware

The hardware were a considerable part of the project and were the source of quite a few problems. This ranged from motor drivers malfunctioning to excessive heat generated by DC/DC converters. The problems were dealt with according to priority to get prototype working. Problems with having low power consumption on a DC/DC converter (<20%) and high capacitive load causing instability of the converter where first pointed out in 2008 by (Plante 2018). We acknowledge that the uncertainty of not having calculated the inductive or capacitive load of our ROV application would be an area for further improvements, as capacitors and ferritte cores have been placed without calculating whether the load is inductive or not. The reason being that 100m cable that is twisted will add inductive load and that actuators would also contribute to this. Ferritte cores were added to reduce noise of the power cables thus protecting signal wires.

One of the major problems was the occurrences of voltage spikes due to reverse voltage occurring when changing polarity of the actuators. These spikes paralysed our system as the DC/DC converter restarted, therefore actuators and connected motor drivers were attached to an own converter to hinder the SBC from losing power. This created two separated systems for the actuators and the rest of the ROV system, making the solution more redundant, by separating noise generating equipment from noise sensitive equipment. Furthermore, motor drivers were replaced to have integrated reverse power protection, and were programmed to run with lower acceleration.

The I2C bus hardware connection worked accurately, except the pressure sensor which were susceptible to noise from echo sounder. The problem were solved using an Arduino as intermediate reducing the distance and then sending data over serial communication. This had been a feared scenario as there were several devices that generated noise, such as converters, echo sounder etc. The ground for all of the signals were wired together and separated using galvanic separation, as well as twinning all SCL and SDA (I2C signals) against each other. The measures taken provided protection against noise, but further improvement that could have helped protect the pressure sensor such as re-shielded cable, relocation, I2C buffer or a band pass filter.

The communication internally in the ROV were a big part in the process of creating the prototype, and many hours were spent solving problems. The usage of I2C were simple to implement, but proved an unstable connection in conjunction with noisy equipment. The group would recommend experimenting with different protocols for data transfer.

5.1.3 Depth control system

The control system successfully managed to control the ROV's depth with a minor deviation from the setpoint. The tested gains of the controller turned out to be too low, but can be easily adjusted and optimised with a proper test setup using the hidden feature in the GUI mentioned in section 4.3.2.

For the final setup with the JRK 21v3 controllers mentioned in section 4.1.15, the depth controller could be implemented directly into the motor controller using the JRK software or directly into the java application. This would result in less electronics and would make production of the ROV cheaper, along with decreasing the chance for errors by removing the communication between the Arduino mega and the Raspberry pi. Although this would be an improvement, it is not necessary to properly utilise the ROV.

By taking into account the echo sounder from the boat, the controller can be improved further by adding safety features, to prevent collision due to sudden depth changes. A more advanced controller can also be implemented to control the actuators differently to prevent the ROV from rolling over. This would help get a more consisting video when approaching the seafloor, but is not considered necessary.

By conducting experiments to acquire the rest of the hydrodynamic model, the depth control can be simulated more accurately. This is considered a valuable foundation for further testing, making it possible to easily test changes like wing angles, faster actuators, MPC controller and numerous other adjustments.

5.1.4 Software solution

Software solution for both the ROV and the client application ended up being of considerable sizes developed in a short span of time. When developing applications of this magnitude there will always be a potential for improvement, such as limiting global variables and shortcuts taken to enable program to function. One thing that could be done differently when the client disconnected, the objects and threads were still running in background, then when the client connected again the number of threads multiplied. A solution would be to finish all the threads using the executor framework with a command from the client, then restart the start Script on SBC.

Our solution implemented fixed rate thread running, which set the update speed of each thread. This worked as intended and allowed for control of the real-time application in a better way than traditionally (free-wheeling), as when a thread slept other threads were then running, utilising more processor capacity. The fixed rate solution also enabled control of update rate of the I2C bus to be set, controlling the update rate of Gyro which were needed for utilising the complementary filter to get roll and pitch.

The GUI implemented a combination of fixed rate threads in the same way as the ROV program, and frames that listened for actions and changes in the user input. Along with implementing an observer design pattern to update the GUI only whenever a value changed, we managed to make the application run smoothly. This design pattern could further improve the performance by updating only the values that are changed, instead of the entire frame.

Extra tools like the graphic sounder and I/O controller proved to be useful. These frames could be closed and opened to keep the GUI simple while these features were not needed. Although the echo sounder frame provides the user with a graphic representation of the depth, these graphs are not synchronised in the x-axis rendering the representation less useful. This could be improved by for instance using the speed and real-time principles to scale the x-axis to real meters and compare this to the horizontal distance to the ROV. This could give the user more information regarding how close the ROV is to a possible collision at all times.

Overall the GUI was simple, responsive and worked as intended, which made it easy to use and understand. A finished product could improve this area further by allowing more customisation of the GUI to each user, and possibly decide which sensor values were necessary to log. The features to enhance the user-experience were not prioritised considering that this is the first version of the application and getting the ROV to work properly was more important to prove the usefulness of the concept.

Communication between the client and the server worked impeccable for sending and receiving commands, by using a parser developed during project. Commands were sent to the server using strings and sensor data to the client using byte-Array. However, what could be done differently were the UDP video-solution, as this required a static IP of the client to work. This could be done by creating a link between Python program and Java-application in the ROV. Then the connection from the client would be sending its' IP to the Python program, and begin video transfer to this IP. This could on the other hand lead to more errors, and therefore not implemented in this thesis.

Communication over I2C protocol worked properly after the implementation phase were over. The main problem with the I2C protocol were running a setup with multiple masters, the plan from the start was to use Arduinos as intermediates between senors and SBC, as they had integrated libraries for extracting finished data. This did not work as it crashed after some minutes. The next step were to use Teensy in place of Arduino, as it supported two I2C busses. The idea were to use Teensy as master against sensor on one bus then being a slave on main I2C line, this also failed as only approximately 50% of data from sensors reached the SBC. The decision were made to create our own application using java on the SBC, using some pre-made work and extending upon these with information found in Arduino libraries and info gained on this topic. This solution worked better than expected and were therefore extended to all sensors valid for implementation. The software for I2C were therefore a success even though the hardware part of I2C were harder to implement as discussed in section:5.1.2.

5.1.5 Test

The tank test was conducted by towing the ROV by hand. It might have been better if we could have used the tow motor to drag it through the water. This would have given a more consistency and accurate speed during the test, and would have resulted in more data about the ROVs stability.

There could have been one more tank test after the electronics were added and the compartments were filled with oil. This could have resulted in oil leaking into the tank if the ROV was not sealed good enough, which is the reason this was not done.

The dry test were conducted at the lab and involved continuously testing of the systems. This was also a part of the troubleshooting process, and each part were thoroughly tested to check if they worked as intended, both separately and together as a fully functional system. This made the dry test and troubleshooting efficient and there were no need for extra dry tests after the the ROV was completed.

The software were also thoroughly tested during this process, and modifications were made where problems were detected. Common for the dry tests were that the group members that had worked on the software section, were responsible for conducting the test. This was done so that the person who knew exactly how it should work could evaluate it, and it would be easier for him to find the fault. Additionally, the tests were conducted several times to ensure that the systems were reliable.

The data collected during the ocean test were of good quality. This data shows that the ROV is working and the controller does what it is supposed to. It displays that the ROV tries to control itself to lay in the surface, and later at 2 meters below the surface. However, some disturbances can be seen in the graph (figure: 4.64), but this is likely due to waves on the surface and the boat tugging the cable.

It has to be taken into consideration that only 10 minutes of test data was collected because the ROV lost I2C communication. More test at different speeds and depth would have given a better picture of the functionality of the ROV.

There might be several reasons for losing the I2C communication. One plausible explanation could be that it was caused by the condensation; another is that an I2C wire in one of the compartments, leading to the Raspberry Pi had come loose. However, the exact cause of the problem were not found and this being the last day of testing, it was too late to do anything about it.

The test revealed condensation, which is not ideal and need to be worked with to eliminate. There are many options that could work and some of them are: better cooling inside the camera house, since the 5V DC/DC convert generates a lot of heat. Installing some kind of a dehumidifier, but this could take up a considerable amount of space. The simplest and maybe best solution could be to use small packs with dehumidifying materials.

The boat that were used during the test was rather small and difficult to operate from due to its design. An outboard motor also made it impossible to deploy the ROV from the aft and the cable needed constant attention. A larger boat with an onboard motor and a platform at the aft would have made it easier to deploy and test the ROV. This would not influence the result directly, but would have made the testing a lot easier.

5.1.6 Possible areas for utilisation

The areas of which is applicable for a finished version of the towed ROV;

- Explore new or known areas, as well as assist in scientific research, archaeology etc.
- Discover wreckage, natural disaster, quick deployment for man recovery-search
- Charting of seafloor, however governments needs to be consulted as seabed falls under national security.
- Surveying pipe/power-lines, underwater trench examination.

5.1.7 Prototype to finished product

Equipment for different objectives

For longer periods of operation it would be profitable to have a larger vessel to operate from. It would be preferable to have power outputs for computers, which also could supply charging of batteries for the ROV. This would remove the need to dock and recharge computers when batteries are depleted.

The ROV also has a cable out from electrical box that is disconnected, this cable is for attaching external sensors.

5.1.8 Challenges with a towed-ROV

There are several challenges with building and operating a towed-ROV, as it operates under water with varying degrees of water currents, pressure and temperature. These factors lead to the ROV having multiple obstacles that needs to be handled. Other provocative features are varying degrees of wave heights, which will interfere with depth control, that in itself is a different area of study.

Upon usage, additional challenges might arrive such as collision and the cable getting stuck. The risk of these incidents happening can be greatly decreased, but can always occur. These challenges do not prevent the concept of a towed ROV to be a useful concept, but should always be accounted for.

5.1.9 Project Execution

The Gannt diagram that were developed during the pre-project report was followed as much as possible. Some of the tasks were completed ahead of schedule, and others assignments were delayed. When necessary, the work time was extended in order to meet deadlines. Additionally,

some of the tasks were rescheduled because they depended on other assignments. The Gannt diagram also shows that we were not able to complete all the tasks.

Workload was distributed between the group during the pre-project phase, so that everyone would have an equal amount of work. However, this was not always the case and when someone lacked work, they would help where it was needed. We managed to adapt to unforeseen challenges with research and guidance from our supervisors.

Continuously writing during the project helped us maintaining a consistent workflow and overview over previous and future tasks. Good documentation helped keep track of every detail, to make sure work on different subjects did not interfere. During the building phase, there wasn't much time to write and the group relied on good communication and teamwork to keep the process running forward.

Chapter 6

CONCLUSION

During the course of this project we have studied the idea of a towed ROV and built a prototype to examine the concept. The requirements specified for the prototype was to make a functional prototype which works as a multipurpose platform, that should be able to control depth automatically, either with distance from the seabed or depth of the ROV. This should be controlled from GUI using echo sounders and pressure sensors. Another feature set was ability to stream and store video of the seabed, in addition to having the option to mount external sensors as pay-load.

Our results suggest that the final prototype is functional, modular and serves its purpose. It shows that it is possible to make a towed ROV by using affordable and available materials. To make the ROV function as specified, we have looked at different factors like drag, lift and pressure. This has helped us understand the theory behind the functionality of the prototype and we have tried to use this to our advantage. The design can still be improved, but we believe that it can be used as a foundation for further work.

Testing indicates that the depth control using a single PID-controller works, by adjusting the gain transmitted using fuzzy logic. Additionally, video is streamed from ROV to the GUI where the application stores the video and attached sensor data.

The specifications is met with functions in the GUI where user defines mode to use, either depth from seafloor or depth of ROV. The user can send setpoint to the ROV which will then adjust its output from PID-controller. The depth control works but needs to be tuned for optimisation. The prototype fits the role of multipurpose platform by having the option to mount additional equipment.

We regard the project as a success, and having an interdisciplinary project with people of differ-

ent background has been exciting and educational. The prospect of controlling a project from start to finish with project management, and the opportunity to apply knowledge from 3.years of engineering school has been rewarding. We therefore conclude that towed-ROV is a valid concept that should be further developed. We believe that the concept of a towed-ROV has a role to play in further studies of the ocean.

6.1 Further Development

Based on the work done, we see that there are several aspects that can be improved:

- Trying out different materials
- Improve hydrodynamics and stability
- Simplifying assembly
- Easier solution for waterproofing
- Finish mathematical model to improve simulation
- Optimise depth control
- Improve communication in ROV
- Extend functionality of GUI

Bibliography

100 Mbps Ethernet / IEEE 802.3u including 100 Base-T (2018). electronics-notes.com. URL: https://www.electronics-notes.com/articles/connectivity/ethernet-ieee-802-3/100-base-t-802-3u.php (visited on 05/13/2018).

Adafruit 9-DOF IMU Breakout (2018). Adafruite. URL: https://cdn-learn.adafruit.com/downloads/pdf/adafruit-9-dof-imu-breakout.pdf (visited on 05/24/2018).

Adafruit Ultimate GPS (2018). Adafruit Industries. URL: https://cdn-learn.adafruit.com/downloads/pdf/adafruit-ultimate-gps.pdf (visited on 05/12/2018).

Aerotoolbox (2018). *Aerodynamic Lift, Drag and Moment Coefficients*. Aerotoolbox. URL: http://aerotoolbox.net/lift-drag-moment-coefficient/ (visited on 04/09/2018).

Airfoil-tools (2018[a]). *Power-line communications for smart grid: Progress, challenges, opportunities and status*. Airfoil tools. URL: http://airfoiltools.com/airfoil/naca4digit (visited on 01/22/2018).

— (2018[b]). *Reynolds number calculator*. Airfoil tools. URL: http://airfoiltools.com/calculator/reynoldsnumber (visited on 04/09/2018).

Airmar-DST800 (2013). Airmar Technology Corporation. URL: http://www.airmar.com/uploads/Brochures/dst800.pdf (visited on 05/12/2018).

Algorithm for computation of fundamental properties of seawater (1983). UNESCO. URL: http://ioc-unesco.org/index.php?option=com_oe&task=viewDocumentRecord&docID=3939 (visited on 05/18/2018).

Arduino.cc (2018a). *Arduino Mega*. Arduino. URL: https://store.arduino.cc/usa/arduino-mega-2560-rev3 (visited on 01/22/2018).

— (2018b). *Arduino Pro Mini*. Arduino. URL: https://store.arduino.cc/usa/arduino-pro-mini (visited on 01/22/2018).

— (2018c). *Arduino Pro Mini*. Arduino. URL: www.pjrc.com/teensy/teensy31.html (visited on 01/22/2018).

— (2018d). *Arduino Uno*. Arduino. URL: https://store.arduino.cc/arduino-uno-rev3 (visited on 01/22/2018).

ASPENCORE (2018). *The MOSFET*. URL: https://www.electronics-tutorials.ws/transistor/tran_6.html (visited on 05/21/2018).

Bluerobotics (2018). *Fathom-X Documentation*. URL: http://docs.bluerobotics.com/fathom-x/ (visited on 05/03/2018).

Bye, Robin T. (2017). *Cybernetics compendium*. NTNU. URL: https://ntnu.blackboard.com/bbcswebdav/users/robintb/Kybernetikk/Compendium/cybcompendium.pdf (visited on 02/08/2018).

Chacona, Scott and Ben Straub (2014). *Pro Git*. Apress. URL: https://github.com/progit/progit2/tree/master/book.

Company., Thomas Publishing (2018). *About Actuators*. URL: www.thomasnet.com (visited on 05/21/2018).

DNV-RP-C205 (2010). DNVGL RP. DNVGL. URL: https://rules.dnvgl.com/docs/pdf/dnv/codes/docs/2010-10/rp-c205.pdf (visited on 01/31/2018).

Dr Peter Ridley Julien Fontan, Dr Peter Corke (n.d.). *Submarine Automatic Control*. Queensland University of Technology, CSIRO Manufacturing Science, and Technology. URL: https://eprints.qut.edu.au/33854/1/33854.pdf.

Ethernet Cable Types, Performance & Pinout: Cat 5, 5e, 6, 6a, 7, 8 etc (2018). electronics-notes.com. URL: https://www.electronics-notes.com/articles/connectivity/ethernet-ieee-802-3/cables-types-pinout-cat-5-5e-6.php (visited on 05/13/2018).

Fossen, Thor I (1994). *Guidance and control of ocean vehicles*. John Wiley & Sons Inc.

Giang, Ken (2018). *PLA vs. ABS: What's the difference?* 3DHubs. URL: https://www.3dhubs.com/knowledge-base/pla-vs-abs-whats-difference (visited on 05/26/2018).

GLA750-P 12V (2018). GimsonRobotics. URL: https://gimsonrobotics.co.uk/categories/linear-actuators/products/gla750-p-12v-dc-linear-actuator-with-position-feedback (visited on 05/24/2018).

Gonzalez, Javier Fernandez (2016). *Mastering Concurrency Programming with Java 8*. 1.st ed. PACKT PUB. ISBN: 978-1-78588-612-6.

Guidance for diving supervisors IMCA D 022 (2016-08-1). "10- General diving procedures, Section 10.3 Diver's umbilicals". International Marine Contractors Association. URL: https://www.imca-int.com/publications/155/guidance-for-diving-supervisors/ (visited on 05/14/2018).

Haugen, Finn (2010). *Ziegler-Nichols' Closed-Loop Method*. TechTeach. URL: http://techteach.no/publications/articles/zn_closed_loop_method/zn_closed_loop_method.pdf (visited on 03/05/2018).

Jan Petrich, Daniel J. Stilwell (2009). *Model simplification for AUV pitch-axis control design*. The Bradley Department of Electrical, Computer Engineering, Virginia Polytechnic Institute, and State University, Blacksburg. URL: https://www.sciencedirect.com/science/article/pii/S0029801809002649.

Kölling, David J.Barnes & Michael (2017). *Objects First with Java*. 6th. Pearson. ISBN: 978-1-292-15904-1.

Konark Sharma, Lalit Mohan Saini (2018). *Power-line communications for smart grid: Progress, challenges, opportunities and status*. National Institute of Technology, Department of Electrical Engineering. URL: https://www.sciencedirect.com/science/article/pii/S1364032116305111 (visited on 05/15/2018).

Kurose, James F. and Keith W. Ross (2012). *Computer Networking: A Top-Down Approach*. Pearson. ISBN: 9780273768968.

MS5837-30BA (2015). TE Connectivity Ltd. family of companies. URL: http://www.te.com/commerce/DocumentDelivery/DDEController?Action=showdoc&DocId=Data+Sheet%7FMS5837-30BA%7FB1%7Fpdf%7FEnglish%7FENG_DS_MS5837-30BA_B1.pdf%7FCAT-BLPS0017 (visited on 05/14/2018).

Nakamura, Mealani (2018). *How an airfoil works*. MIT. URL: http://web.mit.edu/2.972/www/reports/airfoil/airfoil.html (visited on 05/21/2018).

NASA (2018). *The drag equation*. NASA. URL: https://www.grc.nasa.gov/www/k-12/airplane/drageq.html (visited on 03/09/2018).

Nedelkovski, Dejan (2018). *MEMS Accelerometer Gyroscope Magnetometer*. URL: https://howtomechatronics.com/how-it-works/electrical-engineering/mems-accelerometer-gyrocope-magnetometer-arduino/ (visited on 05/21/2018).

Negnevitsky, Michael (2011). *Artificial Intelligence A Guide To intelligent Systems*. 3rd ed. Pearson Education Litited. 471 pp. ISBN: 978-1-4082-2574-5.

Nenad Popovich Sudeep Lele, Niraj Garimella (2006). *Submarine Depth Control*. Auckland University of Technology, Faculty of Design and Creative Technology. URL: http://www.wseas.us/e-library/conferences/2006cscc/papers/532-105.pdf.

Nise, Norman S. (2011). *Control Systems Engineering*. 6th. John Wiley & Sons Inc. ISBN: 978-0-470-64612-0.

NMEA (2018). *NMEA 0183 Standard*. National Marine Electronics Association. URL: https://www.nmea.org/content/nmea_standards/nmea_0183_v_410.asp (visited on 05/11/2018).

Ormbostad, Just Erik (2014). *Montorhaandboka, NEK 400:2014*. 5th. Elforl. ISBN: 9788273456151.

Pieter, Jan (n.d.). *Reading a IMU Without Kalman: The Complementary Filter*. URL: http://www.pieter-jan.com/node/11.

Plante, Shue (2018). *Guidelines for Reliable DC/DC*. NASA. URL: https://nepp.nasa.gov/mapld_2008/presentations/t/01%20-%20Shue_Jack_mapld08_pres_1.pdf (visited on 05/24/2018).

Raymond, Eric S. (2018). *NMEA Revealed*. URL: http://www.catb.org/gpsd/NMEA.html#_sources_and_applicable_standards (visited on 05/11/2018).

ROV umbilical and tether (NA). Nexans. URL: http://www.nexans.no/eservice/pdf-localgroup/ROV_umbilical_and_tether.pdf (visited on 05/14/2018).

Scott, Jeff (2018). *NACA Airfoil series*. Aerospaceweb. URL: http://www.aerospaceweb.org/question/airfoils/q0041.shtml (visited on 05/21/2018).

Society, Marine Technology (2018). *WHAT IS AN ROV?* URL: http://rov.org/rov_overview.cfm (visited on 05/21/2018).

SOS Leak Sensor (2018). Blue Robotics. URL: http://docs.bluerobotics.com/sos/#schematic (visited on 04/12/2018).

Sound in water (n.d.). sonar, echo sounder. Simrad. URL: https://www.simrad.com/www/01/nokbg0237.nsf/AllWeb/7387353A1D263983C12574B2003BEDDF?OpenDocument (visited on 05/05/2018).

Steven W. Moore Harry Bohm, Vickie Jensen (2010). *Underwater Robotics; Science, Design & Fabrication*. Marine Advanced Technology Education Center. ISBN: 978-0-9841737-0-9.

Sujay D. Kadam, Kunal N. Tiwari (n.d.). *A simplified approach to tune PD controller for the depth control of an autonomous underwater vehicle*. Department of Instrumentation Engineering, Shri Guru Gobind Singhji Institute of Engineering and Technology. URL: http://www.communityresearch.org.nz/wp-content/uploads/formidable/PD-AUV-IPDT.pdf.

Telos (n.d.). *I2C – What's That?* URL: https://www.i2c-bus.org/.

The NMEA 0183 Information sheet (2015). Active Research Limited. URL: https://www.actisense.com/wp-content/uploads/2017/07/NMEA-0183-Information-sheet-issue-4-1-1.pdf (visited on 05/12/2018).

Wellings, Andy (2004). *Concurrent and Real-Time Programming in Java*. Wiley. ISBN: 0-470-84437-X.

Ylonen, Tatu (n.d.). *Alert on SSH Keys*. URL: https://www.ssh.com/ssh/key/.

Appendices

Appendix A

Pre-project

FEASIBILITY STUDY -REPORT

FOR BACHELOR THESIS

NTNU
Kunnskap for en bedre verden

TITLE:
Towed – ROV

CANDIDATE NUMBERS:			
Kristian Homdrom - 460012 **Marius Nonsvik - 460309** **Daniel Reite - 269658** **Morten Solli - 254801**			
DATE: **18.01-2018**	COURSE CODE: **IE303612**	COURSE: **Bachelor thesis**	DOCUMENT ACCESS: - Open
STUDY: **AUTOMATION, PRODUCT AND SYSTEM DEVELOPMENT**		PAGES/ATTACHMENT 17/	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".

The thesis towed ROV is provided by the NTNU institution and is a development project of creating a functioning prototype. The objective is to have the ROV towed by a support vessel, and regulate its distance from the seabed based on sonar and depth sensors. The vessel should be able to collect different types of data and be a multipurpose craft, with focus on easy deployment and change of purpose area.

In this pre-project report there have been made a work schedule and the workload have been distributed amongst the group members. Tools for project control and development such as Asana, Instagantt, and risk analyzes are used to get a detailed view of the task at hand. To develop the ROV there will be utilized tools for 3D-moddeling and construction, concepts about cybernetics and control system engineering for controlling the ROV, additionally develop computer program to record, transfer and store data.

One of the most challenging aspects of this project will be to make a waterproof design for the electronics, and make it stable enough to get satisfying data from the onboard sensors. It is important to stick to the scheduled work plan and procedures pointed out in this pre-project report.

This pre-project report will be the foundation on which our thesis will be built upon.

Postadresse
Høgskolen i

N-6025
Norway

Besøksadresse
Larsgårdsvegen 2Larsgårdsvegen
7694 05 00636
Ålesund
www.hials.no

Telefon

Epostadresse
postmottak@hials.no

Telefax
70 16 12 00

Bankkonto
70 16 12 00 70 16 13 00

Foretaksregisteret
NO 971 572 140

TABLE OF CONTENT

TABLE OF CONTENT **2**
1 INTRODUCTION **3**
2 CONCEPTS **3**
3 PROJECTORGANIZASION **4**
3.1 PROJECT GROUP 4
3.1.1 Assignments for the project group - organization *4*
3.1.2 Assignments for project leader *4*
3.1.3 Assignments for secretary *5*
3.1.4 Assignments for members *5*
3.2 MANAGEMENT GROUP 5
4 AGREEMENTS **5**
4.1 WORKSPACE AND RESOURCES 5
4.2 GROUP AGREEMENTS AND ATTITUDES 5
5 PROJECT DESCRIPTION **6**
5.1 THESIS PROBLEM 6
5.2 SPECIFICATION 7
5.3 METHOD 7
5.4 INFORMATION GATHERING 8
5.5 RISK ANALYZES 9
5.6 MAIN WORK ACTIVITIES 11
5.7 SCHEDULE AND TIMETABLE 12
5.7.1 Master plan *12*
5.7.2 Project control assets *13*
5.7.3 Development assets *13*
5.7.4 Evaluation Intern control *14*
5.8 DECISION MAKING PROCESS 14
6 DOCUMENTATION **15**
6.1 REPORT AND DOCUMENTATION 15
7 PLANED MEETINGS AND REPORTS **15**
7.1 MEETINGS 15
7.1.1 Meetings with control group *15*
7.1.2 Project meetings within group *16*
7.2 PERIODICAL REPORT 16
7.2.1 Progress report *16*
8 PLANNED DEVIATION MANAGEMENT **16**
9 EQUIPMENT REQUIREMENTS FOR PROJECT EXECUTION UTSTYRSBEHOV/FORUTSETNINGER FOR GJENNOMFØRING **17**
10 REFERENCES **17**
APENDIXES **17**

1 INTRODUCTION

Over 70 percent of the world is covered in water, and yet the underwater domain is less explored than the surface of the moon. This domain will most likely be the target for future exploration and resource extraction. Therefore, our main goal is to build up knowledge on how to build an underwater sea craft which can handle exploration, search and rescue and more, to meet the challenges of tomorrow.

This project will be a new way to work for all of us, having a project from the planning phase to a finished product. It is versatile, and we can use everything we have learned to this point in our studies. There are high possibilities to implement Cybernetics in the steering system, sensor handling, and vision transfer. There will also be a lot of calculations regarding hydrodynamics, buoyancy, material properties and movement, which will help in further research.
There will be a lot of challenges as the whole operation will be underwater, but we are positive about finding good solutions.

This thesis will contain an amount of research, but with great potential for future development. This might be storing of pictures on shared cloud like google maps and so on.
The ocean space is one of the least explored areas for us humans, this will change in the years to come and there will be needed competent professionals within this area.

NTNU is the provider of the thesis, and was promoted to start building up knowledge on subsea level for the campus in Aalesund. This thesis will be an experimental one with a lot of try and error, but the goal is to create a prototype for the university which can be a foundation for later studies in the field.

Over the course of this project our goal is to develop a functioning towed ROV which can be controlled automatically based on depth and distance to the seafloor, this will be done using regulation techniques. The user must be able to override the automatic control in case of an emergency or issue. The ROV is intended to be able to map the seafloor using a camera and read multiple sensor values. This information should be transferred to the user in real-time and saved. The design of the ROV must be able to handle the water resistance and pressure, as well as being easy to manufacture. Furthermore, the ROV design must be durable and versatile, making it easy to add future extensions or make changes.

2 CONCEPTS

- ROV Remote operated vehicle
- UI User interface
- Positive buoyancy The net buoyancy applied to an object is equal to the weight of the fluid displaced by the body. Positive buoyancy means that the net buoyancy is greater than the weight of the object.
- Asana Development tool for project organization, assigns tasks to members etc.
- Instagantt Accessory kit for Asana, will provide a graphical visualization for project tasks.
- LPD – Lean product development

3 PROJECTORGANIZASION

3.1 *Project group*

Studentnumbers
Kristian Homdrom- 460012 Marius Nonsvik - 460309 Daniel Rasmus Reite - 269658 Morten Solli - 254801
Email
Kristian Salvesen Homdrom- krisso@stud.ntnu.no Marius Nonsvik- mariunon@stud.ntnu.no Daniel Rasmus Reite- Danielrr@stud.ntnu.no Morten Solli- Mortels@stud.ntnu.no

3.1.1 Assignments for the project group - organization

Pre-set roles for the group before assignments.

Name	Role
Kristian Salvesen Homdrom	Project leader
Morten Lerstad Solli	Project secretary
Daniel Rasmus Reite	Project designer
Marius Nonsvik	Control systems engineer

Asana will be used to keep track of assignments. Each member will be assigned individual tasks using asana development tool, based on their qualifications. Each task will have one member take charge and at least one other member supporting.

With the use of Asana and its accessory tool Instagantt, the group will have a graphical visualization of the projects progress. Members will also receive mail from Asana when a task is nearing its deadline or is overdue.

3.1.2 Assignments for project leader

- Responsible for project progress.
- Delegate tasks when necessary.
- Contact person.
- Gets the tiebreaking vote if a tie occurs.

3.1.3 Assignments for secretary

- Responsible for sending out meeting invitations.
- Responsible for the meeting report gets written. By either delegating or writing himself.
- Follow up that group members write on the thesis continuously

3.1.4 Assignments for members

- Each member is responsible for the tasks they are assigned is completed.
- Each member must write a personal log containing thoughts about the project, what they have learned, hours they have worked, and reactions around cooperating with other group members. This is important for the self-evaluation part of the thesis.

3.2 Management group

Project advisors:

- Ottar L. Osen
- Paul Steffen Kleppe

4 AGREEMENTS

4.1 Workspace and resources

- Access to bachelor work space and laboratories for POD and Automation
- Access to tools, welding equipment, 3d printers, materials,
- Access to project advisers and lecturers in relevant subjects such as cybernetics, programing, and construction.
- No classified information
- Weekly reports

4.2 Group agreements and attitudes

The group has decided to follow normal working routines, from Monday to Friday. During which working hours on bachelor thesis from 08.15 to 16.00. The group's members may differ from this setup if there is a valid cause, but the rest of the group must be informed beforehand and the person must work in lost hours.

Industry 4.0 is a mandatory course which all members will have to attend and therefore during days this course is running, work on bachelor thesis expires.

To reach deadlines all members must be ready to work overtime if this is necessary. However, this should not be the norm, and every member should stride to make his working hours as efficient as possible to make the set deadlines.

Every member of the group should communicate with all the other members if there is an underlying problem or internal grudge that needs to be handled. It will be the responsibility of the project leader to make sure that every member is listened to and get a stake in decision making.

There is cohesiveness in the group to have at least two persons on every project task, where one has the main responsibility and the other supports. Leading to more tasks to be done in parallel and make sure that every task will have priority from one or more persons.

As an automation engineer it is important to embrace the rate of change and adapt to the rapid advancement in technology. Automation engineers should be pioneers in development and usage of new technology, as it is essential for progress to be made. If an engineer would be satisfied with the knowledge and use of current technology, and not consider new innovative ideas and advancement, productivity in the sector would be stalled. The main goal during this project is to gain useful insight in automation and development in the industry. This includes being able to apply current technology to systems, optimize equipment and processes to help further development.

As a mechanical engineer the focus will be to develop a design and construction that meets the requirements, using as simple solutions as possible. It is important to always look for things that can be improved in a product, and not settle on the first thing that works, as there always will be room for improvement. Using modern technology to achieve the most satisfying results will be necessary.

5 PROJECT DESCRIPTION

5.1 Thesis problem

The earth consists of more than 70 percent oceans, where most of the ocean is unexplored. Today sea exploration is a hot topic, and there are many new things that can be done and old methods that can be improved. ROV's are widely used for exploring and searching underwater, but they have some limitations in regards of battery life and equipment they carry.

The purpose of this bachelor-project is to build an underwater remote operated vehicle which will be towed behind a surface vessel. The underwater ROV's purpose is to be a multipurpose platform where sensors can be easily switched around depending on the task at hand. The task can be charting of the seabed, finding wreckages, and search and rescue missions.

The main objective for this thesis is to create a functional prototype and test it in real conditions. The ROV will be built from scratch, this will be done trough designing, prototype building and optimizing the build. It needs to stabilize itself behind the surface vessel at a given depth, this will be solved with use of cybernetics. To record, read, and store sensor data the ROV will need to communicate with the surface vessel, there will also be a need for controlling the steering system and camera, there will be made a Java program to help with this.

5.2 Specification

By the end of this project, the goal is for the ROV to have a waterproof design that navigates properly under water. The design must be durable enough to withstand the pressure and the forces acted upon it under water.

The depth of the ROV must be possible to adjust from the UI. Preferably the ROV will be able to maintain a specific distance from either the surface or the seafloor. The ROV will be equipped with a depth sensor, sonar to properly adjust the wings to obtain the required depth. In addition, a camera is used to capture a video of the seafloor to show the user in real time and save this video to a computer or database. The final product is considered a multi-purpose vehicle, that should have the ability to be further modified or add functionality.

5.3 Method

Asana and Instagantt will be used to keep track of when each assignment should be done, who is responsible for starting it, and which person is responsible for finishing it.

The project will start with research and information gathering to find out which design, components, and software. The idea of Lean product development will be used to development the prototype to be as efficient as possible. We will also use the knowledge gained in the research and development phase, and really on calculations.

The principle of trial and error will be applied since it is not always possible to predict how objects and equipment will behave in water and where problems will occur. Finally, the last month will be used to focused on writing the remaining of the thesis.

Lean product development is based on the lean thinking principles, which were originally designed for the lean manufacturing. Lean thinking is the process of utilizing less of everything, examples are less resources, less development time and less production time. (NPD solutions , 2016)

The fundamental cores of LPD are:

- Creation of re-usable knowledge
  - Knowledge created will be available for similar later projects.
  - Knowledge stored and easily accessed
- Set-based concurrent engineering
  - Multiple tasks ongoing at the same time
  - Decrease downtime on a project
  - More rapid development, more utilization of manpower and resources.
- Teams of responsible experts
  - Every member associated on project will have designated areas they are expert on
  - Utilize previous knowledge from the available team
- Cadence and pull
  - Autonomous teams of engineers, possible for engineers to work independently from each other
  - Plan their own work day, week and only submit report on meeting. No need for supervising, as work will be done according to plan.
- Visual management
  - Visualize project progress, datagrams, using diagrams ore charts to enable fast recognition of patterns ore trends.
- Entrepreneurial system designer
  - Have one person responsible for the design of the product.

The LPD will be a viable development tool for our project as we have limited resources and time to make our final prototype. LPD is designed to be as efficient as possible with the least resources available, thus it will be a good fit for our bachelors' thesis.

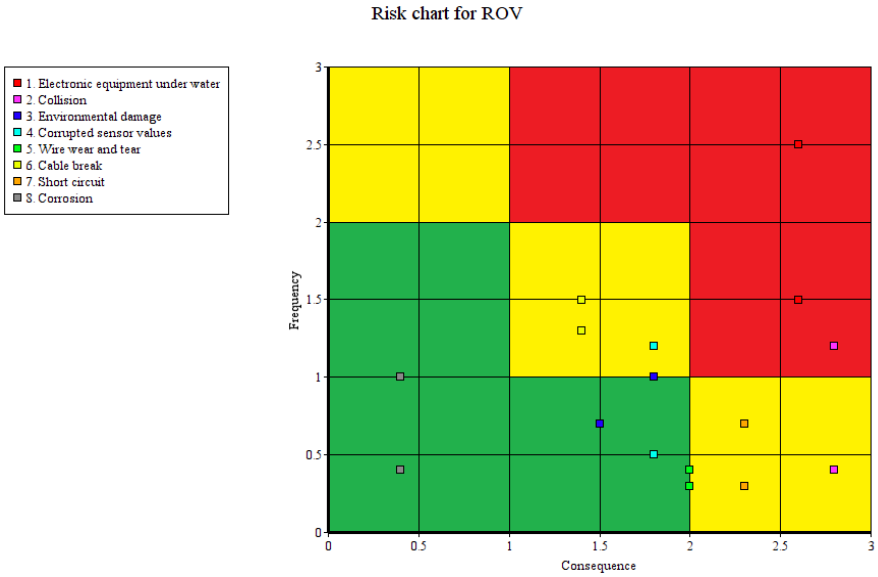
The regulation is fundamental to make the ROV autonomous, precise and agile. Deciding the appropriate regulator will be based on the theoretical basis of cybernetics, multiple simulations with various input and finally trial and error. The control system should preferably be able to react quickly, have the minimum amount of overshoot and close to zero steady state error. This will allow the ROV to avoid obstacles and stay at a steady depth to create a better overview of the seafloor.

5.4 Information gathering

Information gathered:
- Existing ROV concept developed by Sea Sciences Inc. Used for subsea data collection.
- Power point presentation of design made by advisor to promote thesis.
- Ideas proposed by advisors, and solutions that might work.
- Basic information regarding NACA airfoil profiles

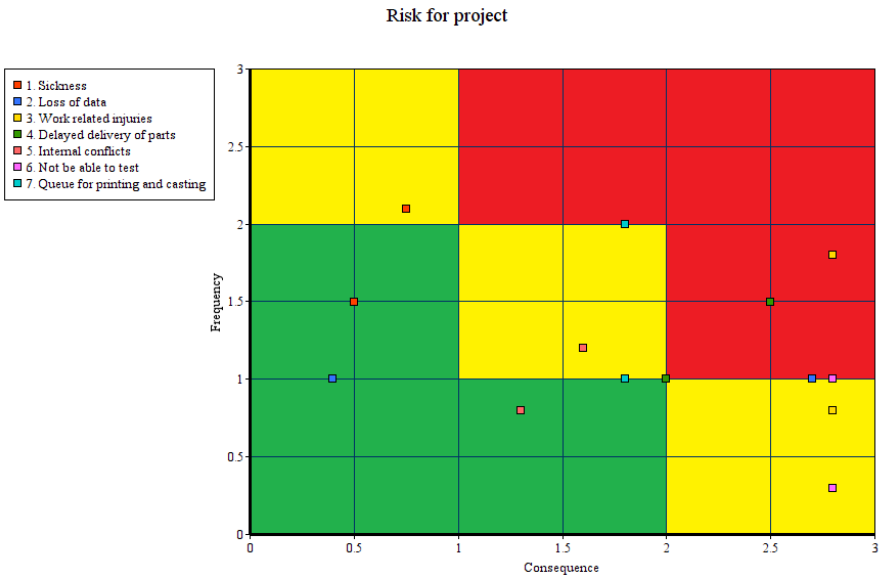
Information that will be gathered:
- Research regarding ROV concepts and technology
- Research regarding hydrodynamics
- Research regarding cybernetic to find best solution for control system
- Waterproofing
- Mechanical methods for construction
- Design ideas
- Force strain and physical behaviour
- NACA airfoil profiles
- Material properties
- Cameras and lights for underwater use
- NMEA protocol and echo sounder
- Programming source code and solutions

5.5 Risk analyzes



Figur 1: Risk analazys for the ROV

Incident	Measures to reduce risk
1. Electric equipment under water	Thoroughly seal all exposed parts of the ROV. Execute leakage tests on parts before usage.
2. Collision	Design a collision safe ROV with buoyancy that will be able to surface in due time, whenever on collision course or signal loss.
3. Environmental damage	Only use chemicals if necessary, and make sure these are eco-friendly and properly sealed.
4. Corrupted sensor values	Implement safety features in the software to make sure the values are probable. Optionally add redundant sensors.
5. Wire wear and tear	Make sure the wire is only in contact with smooth surfaces and is approved for the applied force. Optionally use other materials such as synthetic fiber rope.
6. Cable break	Apply minimum amount of force and stress possible to the cables.
7. Short circuit	Add fuses to the electrical wiring and install the cables and components in strategic places.
8. Corrosion	Design the ROV in properly suited materials.



Figur 2: Risk analazys for the project work

Incident	Measures to reduce risk
1. Sickness	Stay at home to not infect others and work from home.
2. Loss of data	Use Gdrive and sourcetree to backup all data, always have backup of all data.
3. Injuries	Use protection gear and always be two when performing risk associated work.
4. Delays	Order parts in as soon possible.
5. Conflicts	Take up disputes as soon as they arise, bring them forth during group meetings.
6. Not be able to test	Book time in advance for testing, and set a buffer time to assure all is completed before test occur
7. Queue	Plan ahead when to print and start printing early.

5.6 Main work activities

MS -Morten Lerstad Soli
KH- Kristian Salvesen Homdrom
DR- Daniel Rasmus Reite
MN- Marius Nonsvik

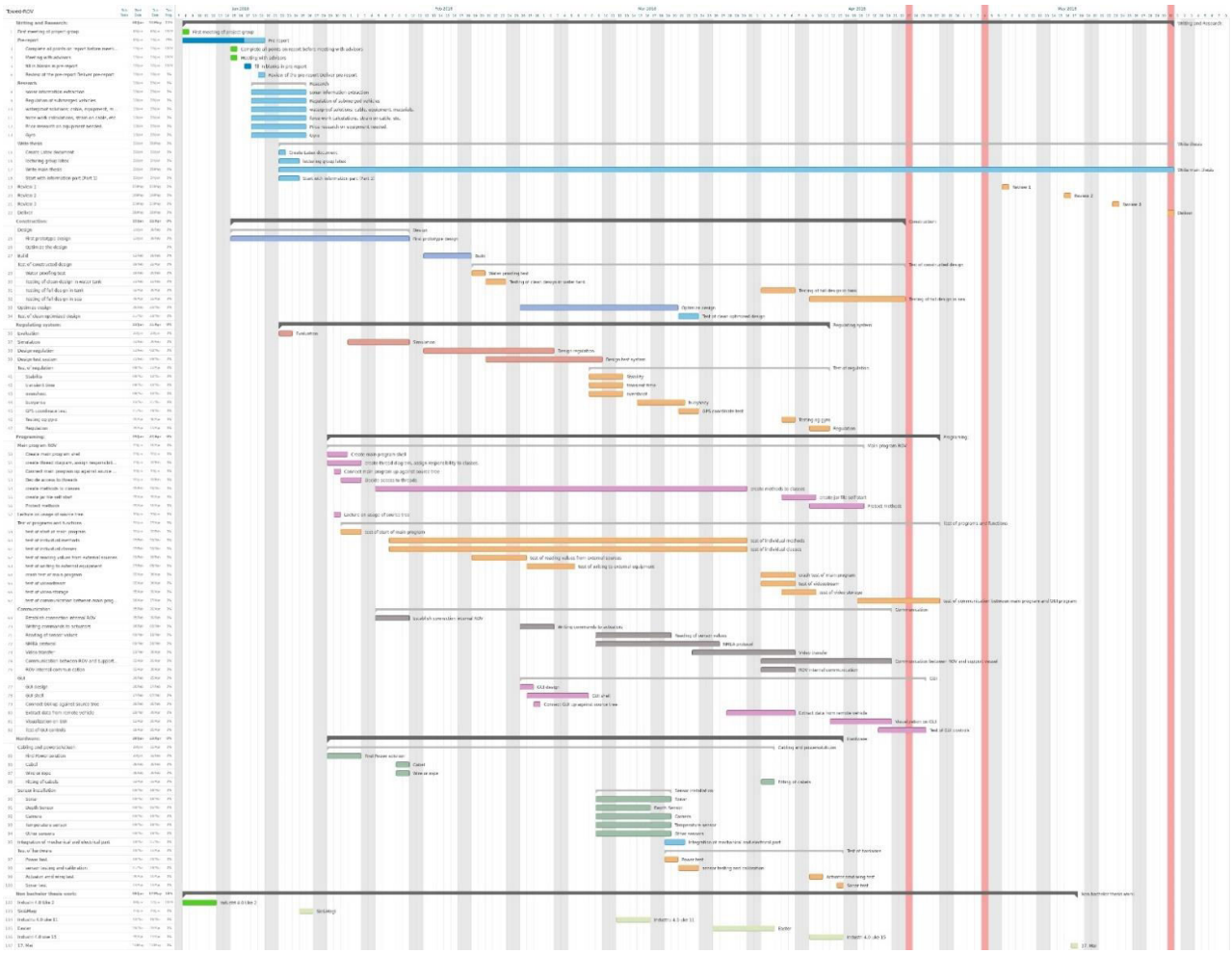
Nr	Main activity	Responsibility	Time/Scope
A1	**Writing and Research**	**MS, KH**	**5 Months**
A11	Pre-report	MS, KH	1 Week
A12	Research	KH, MN	1 Week
A13	Write thesis	MS, KH	5 Months
A14	Review 1	MN	1 Day
A14	Review 2	DR	1 Day
A14	Review 3	MS	1 Day
A15	Delivery	MS	1 Day
A2	**Construction**	**DR**	**2 Months**
A21	Design	DR	4 Weeks
A22	Build	DR	1 Week
A23	Test of constructed design	DR	3 Weeks
A24	Optimize design	DR	3 Weeks
A25	Test of optimized design	DR	3 Days
A3	**Regulating system**	**MN, MS**	**2 Months**
A31	Evaluation	MS, MN	3 Days
A32	Simulation	MN, MS	1.5 Weeks
A33	Design regulation	MN, MS	3 Weeks
A34	Design test system	MS, MN	2.5 Weeks
A35	Test of regulation	MN, MS	2 Weeks
A4	**Programing**	**KH, MN**	**2 Months**
A41	Main program ROV	KH, MS	2 Months
A42	Lecture of usage of source tree	KH	1 Day
A43	Communication	KH, MS	6.5 Weeks
A44	GUI	MN, KH	5.5 Weeks
A45	Test of programs and functions		
A5	**Hardware**	**MS, MN**	
A51	Integration of mech and el parts	MN, DR	3 Days
A52	Cabling and power solution	MS, DR	1.5 Weeks
A53	Sensor installation	MS, KH	1 Week
A54	Test of hardware	MS, MN	1.5 Weeks
A6	**Non bachelor thesis work**	**KH**	
A61	Industri 4.0 week 2	MS	4-5 Days
A62	Industri 4.0 <week 11	MN	4-5 Days
A63	Industri 4.0 week 15	DR	4-5 Days
A64	Ski & Magi	MS	2 Days
A65	Easter	KH	9 Days
A66	17. May	KH	1 Day

In section A2 Daniel is the only person who has been assigned responsibility. This is because he is the only one with the qualifications needed to accomplish these tasks. However, this does not mean he won't get help. If Daniel needs assistance, the group will help him as much as possible. This will depend on how critical their own situation is, and the situation Daniel is facing.

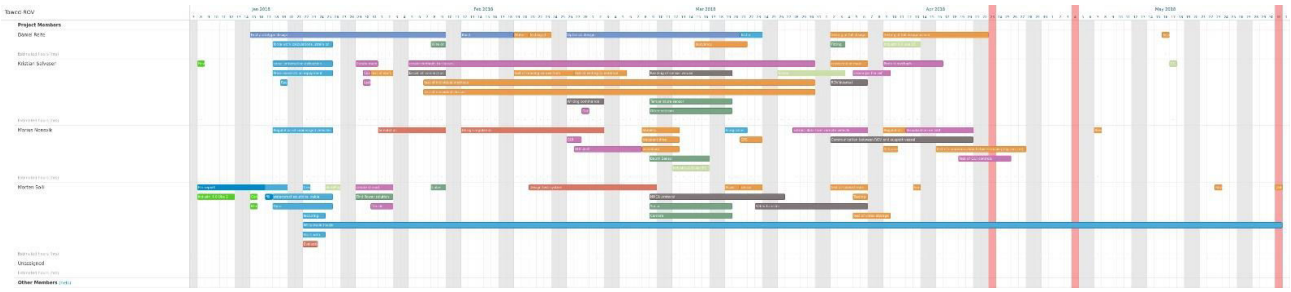
5.7 Schedule and timetable

5.7.1 Master plan

The work schedule will be closely followed due to limited time. Each member will prioritize the tasks they have been assigned and stride to finish them in time. This will prevent delays from affecting other parts of the projected.
There will also be a focus on parallel work, this means the group works in individual teams and cooperate on large asks. The important task of completing a working prototype have been assigned the most time, other parts are trivial and will be focused on when prototype is working good enough.



Figur 3: Gantt diagram of assignments. See apendix for larger picture.



Figur 4: Distribution of resbonsibility of assignments. See apendix for larger picture.

The main plan includes tasks assigned to persons. This is not necessarily being the person doing the task, but rather the one assigned for seeing it done. There will always be at least one person assisting on every task.

With the use of Asana development tool and Instagantt, we generate two timetables. One for tasks according to their due date, and one showing the responsibility of every member of the project team during the course. This enables us to reduce downtime of members and maximize efficiency.

The red lines are set deadlines we have chosen, there are three of them due to probability of delays. When reaching the second deadline all work on the prototype will cease, regardless of completion. Work on the main thesis report will then be the only priority, until the final deadline.

5.7.2 Project control assets

To support the group with scheduled work and keeping track of hours, deadlines etc. Asana will be used, this is a development kit that help developers use to track their work and include features such as:

- Gantt diagrams, graphic representation of work to be done over a period.
- Assign tasks to members
- Add priority to tasks, such as high, low etc.
- Mark tasked as completed when done.
- Easier parallel working when dependent tasks is synchronized against each other.
- Synchronized against Github, enable to mark tasks as completed when source code is uploaded.

5.7.3 Development assets

To create a product of good quality the group must 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.
- Visual paradigm, software used to generate class diagram for java applications.
- Sourcetree, used to create multiple working copies of source code and sync it to the main repository. Enables multiple persons to work on the same code.
- Github/Bitbucket, remote repository to synchronize the code used by the students.
- Google drive, to backup and share all work.
- Cura, 3D printing tool.
- PC schematics, used to create electrical and mechanical diagrams and schematics.

- Siemens NX 11.0, 3D modelling software
- AutoCad, drafting software - computer aided design
- Online chart tool – used for generating risk chart

5.7.4 Evaluation Intern control

Internal control will be handled in several ways. Every week the group will have an internal meeting where progress and challenges will be discussed, here it will also be room for change in work groups if someone is struggling. Participants will also inform all others if they will be absent in the future at these meetings.

There will also be a meeting between project team and the assigned supervisors. During which the team will report on the progress, and receive consultation on work that is to be done and if completed work could be improved.

To have an easy way to see project progress, Asana will be used to get a graphical representation of planned work. This will help break down the project into more manageable pieces which will be assigned to project members.

Criteria for marking an objective as completed will be measured in several ways depending on the respective task. This is because every aspect is different, but as we will be building an underwater vehicle it is impendent that our craft will be water proof.

Another principle we will be implementing is that solutions are good enough, this because there is limited time and a lot of work to be done. There is not enough time to fine tuning every part of the thesis. This will be the cause for our control system that controls set depth and distance to the seabed for the craft.
This does not mean it will not be a priority, but rather when we have a solution that works we need to move on to other objectives to reach the set deadlines. Fine tuning can be done if there will be time at the end of the project duration.

Every part of the project must also qualify for safety regulation, one example is that craft have a safety zone to the seabed that never will be crossed no matter. There is also a matter of environment issue that needs attention, the group will be focusing on building a clean and environmental friendly vehicle that will not contribute to pollution of the seas. This in form of no parts falling off or spillage of oil when vehicle is deployed.

5.8 Decision making process

Every decision regarding the project will be reviewed by the project group in accordance with the advisors. This can be limitations which will be set, or expectations for thesis result. The final vote for project decisions will be up to the group, but they will have to be within the main issue set in the pre-report. The final result will be an answer to the main issue presented there.
Decisions that will be handled during the project will have to be brought up during group meetings. If there are changes that members want to add to the project plan, it will have to be arranged a vote. If the vote is valid, changes will be added to the project plan. The occurrence will be an own paragraph in the weekly report, regardless of the outcome.

6 DOCUMENTATION

6.1 Report and documentation

Documentation routines followed by the group will be that every source found or used will be noted, why it was used or why it was discarded. This strongly applies to the equipment that will be used, this to document why some equipment was chosen over the other. This way hinders our errors from being repeated by others.

Every data sheet of equipment used are to be added as appendixes in the main report, allowing easy access of attributes to the equipment and its limitations.

Tests will be recorded after they have been completed, report should contain every detail on what happened. Improvements made and probably cause for failure of test. Same applies if test went well, everything to be documented.

Every source used in the project is to be referenced and be stated when and where it was found.

Storage of project materials will be done in universities workshops in assigned project areas. They will be secured under password protected doors and explicit access are needed for access to the workshop. This means there are only engineering department who will have access and not the whole campus.

Maintenance will be done with testing of all bought and provided equipment before installation. There will also be set routines for maintenance of constructed parts for each planned test run. The mechanical engineer will provide the verdict if the craft is in satisfactory shape, so it can be tested.

7 PLANED MEETINGS AND REPORTS

7.1 Meetings

Meetings are essential part of a project; however, it is important to clearly set the boundaries on what a meeting should contain. By clearly defining what we wish to achieve with every meeting, we will have a greater progress on the project.

A meeting should not be considered unnecessary by members of the project but as a way of keeping every member up to date on progress being made by other members. It is therefore essential that a meeting will follow a pre-set review. There will need to be a person who will lead the meeting and one person to write the report of every meeting.

7.1.1 Meetings with control group

After first meeting with the control group, it was agreed upon a meeting every 14th day. During these meetings advisor opinion will be sought, explanation of solutions and challenges faced. Furthermore, discussion on what is planned and what project group might do differently will be a topic.

7.1.2 Project meetings within group

There will be a weekly meeting internally in the bachelor project group. The meeting will take place every Monday 08.15, and should take a maximum of 45 minutes.

The objective of the meeting will be to update every member on last week progress, what will be focused on until next week, and update work schedule. Each member will also have the opportunity to deliver complaints or suggestions during this meeting, and inform the group if they will be absent one of the following days.

The meeting will be headed by the project leader and report will be written by the secretary. Report is to be stored on shared medium and will be reviewed every new week at the start of the meeting. Votes will also take place during this meeting if there is need, in the rare case of a voting tie, the project leader will have the deciding vote.

7.2 Periodical report

A periodical report will be written every week to keep track of the projects progress. It will be a short summary on progress made and area of focus for the next period. The report shall be used in parallel with Asana as a tool for upholding deadlines. It will also document measures taken if deadlines are to be met or have been broken.

7.2.1 Progress report

The progress report will be written every week during the course of the bachelor thesis, it should keep track on immediate work and progress made.

Planned deviation is to be documented in the periodic report, if some work must be postponed for more important matters to be finished. Measures to rectify this should also be noted and must be implemented in the next report.

8 PLANNED DEVIATION MANAGEMENT

In the event of a delayed work flow, the group will compensate with additional work hours. If the task proves to be too challenging or time consuming, the group must try to find an easier alternative that does not deteriorate the end result.

Each student will have assigned different tasks. One student is responsible for making sure these tasks are finished in time for the deadline, but will be assisted by at least one other student to complete the task.

Changes in the schedule or deadlines must be brought up and consulted at the weekly meeting. The group will then agree upon a new priority and assign the required amount of work to each student.

9 EQUIPMENT REQUIREMENTS FOR PROJECT EXECUTION UTSTYRSBEHOV/FORUTSETNINGER FOR GJENNOMFØRING

The quality of the equipment needed for project execution and creation of the prototype, will be determined by funds provided by the institution.

As we follow the Lean thinking principle, we will try to minimalize cost while still have good enough quality. There are also standards that need to be met because the ROV will be operating underwater, this means most equipment need to be IP68 classified. IP68 is waterproof for lasting dives.

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/wire	Used for towing ROV
Ethernet cable	Data transfer medium
Power supply	Powering the ROV
Mechanical machines and equipment provided by the POD lab	Constructing the ROV
3D-printer	3D printing the wings and possibly other components.
Mini pc/Odroid/raspberry	Operation processor onboard ROV, runs main program.
Waterproof actuators	Control of ROV wings
Accessories	Seals, wiring, bearings.
Waterproof floodlight	Illuminate the seafloor
Construction materials	Constructing the ROV

10 REFERENCES

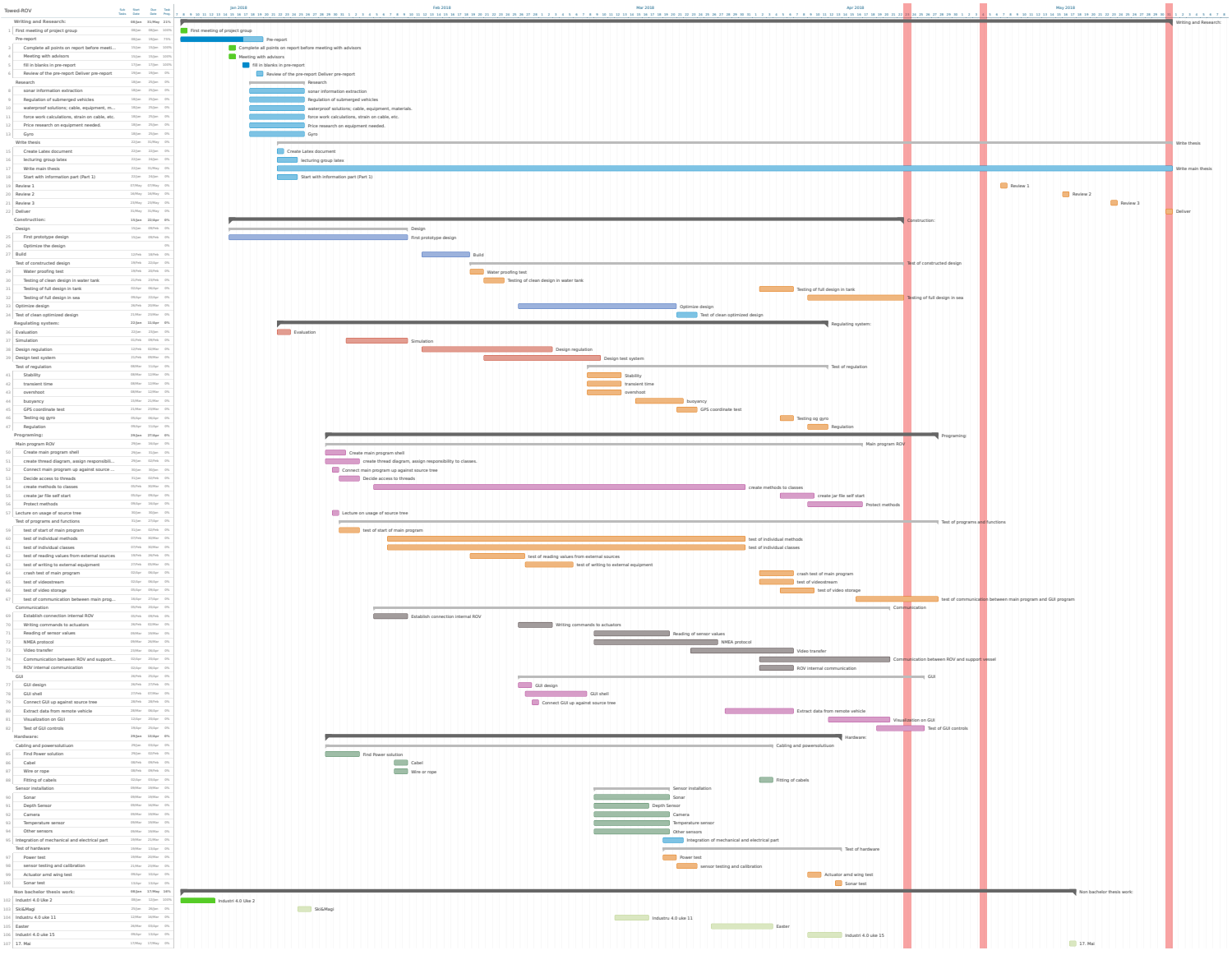
NPD solutions , 2016. *NPDsolutions.* [Internett]
Available at: http://www.npd-solutions.com/lpdpractices.html
[Funnet 18 january 2018].

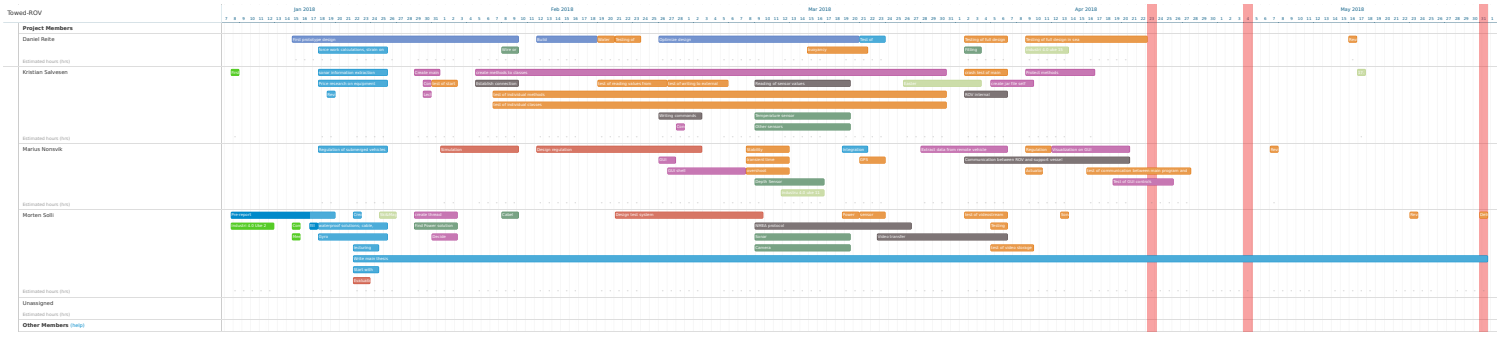
APENDIXES

Apendix A Gantt diagram

Apendix B Workload diagram

Apendix C Asana plan


Towed-ROV
Sub Tasks
Start Date
Due Date
Task Prog.
Jan 2018
Feb 2018
Mar 2018
Apr 2018
May 2018
Writing and Research:
First meeting of project group
Pre-report
Complete all points on report before meeting with advisors
Meeting with advisors
Fill in blanks in pre-report
Review of the pre-report Deliver pre-report
Research
sonar information extraction
Regulation of submerged vehicles
waterproof solutions; cable, equipment, materials.
force work calculations, strain on cable, etc.
Price research on equipment needed.
Gyro
Write thesis
Create Latex document
lecturing group latex
Write main thesis
Start with information part (Part 1)
Review 1
Review 2
Review 3
Deliver
Construction:
Design
First prototype design
Build
Test of constructed design
Water proofing test
Testing of clean design in water tank
Testing of full design in tank
Testing of full design in sea
Optimize design
Test of clean optimized design
Regulating system:
Evaluation
Simulation
Design regulation
Design test system
Test of regulation
Stability
transient time
overshoot
buoyancy
GPS coordinate test
Testing og gyro
Regulation
Programing:
Main program ROV
Create main program shell
create thread diagram, assign responsibility to classes.
Connect main program up against source tree
Decide access to threads
create methods to classes
create jar file self start
Protect methods
Lecture on usage of source tree
Test of programs and functions
test of start of main program
test of individual methods
test of individual classes
test of reading values from external sources
test of writing to external equipment
crash test of main program
test of videostream
test of video storage
test of communication between main program and GUI program
Communication
Establish connection internal ROV
Writing commands to actuators
Reading of sensor values
NMEA protocol
Video transfer
Communication between ROV and support vessel
ROV internal communication
GUI
GUI design
GUI shell
Connect GUI up against source tree
Extract data from remote vehicle
Visualization on GUI
Test of GUI controls
Hardware:
Cabling and powersolutiuon
Find Power solution
Cabel
Wire or rope
Fitting of cabels
Sensor installation
Sonar
Depth Sensor
Camera
Temperature sensor
Other sensors
Integration of mechanical and electrical part
Test of hardware
Power test
sensor testing and calibration
Actuator amd wing test
Sonar test
Non bachelor thesis work:
Industri 4.0 Uke 2
Skiå&Magi
Industriu 4.0 uke 11
Easter
Industri 4.0 uke 15
17. Mai


Towed-ROV
Jan 2018
Feb 2018
Mar 2018
Apr 2018
May 2018
Project Members
Daniel Reite
Kristian Salvesen
Marius Nonsvik
Morten Sæli
Unassigned
Other Members (help)

Towed-ROV

Printed from Asana

Writing and Research:

- [ ] **Morten Solli:** Pre-report — due January 19
  Create the Pre-report for the bachelor thesis.
  Will be the fundamental source for our bachelor thesis to be solved.
  - [x] **Morten Solli:** ~~Complete all points on report before meeting with advisors~~ — due January 15
  - [x] **Morten Solli:** ~~Meeting with advisors~~ — due January 15
  - [ ] **Morten Solli:** fill in blanks in pre-report — due January 17
  - [ ] **Kristian Salvesen:** Review of the pre-report Deliver pre-report — due January 19
- [ ] **Kristian Salvesen:** Research
  Research phase, build a general understanding on the main problems faced during the thesis. What solutions have been developed before, equipment needed, challenges expected and counter measures and prices associated with the equipment needed (budget)
  - [ ] **Kristian Salvesen:** sonar information extraction — due January 25
  - [ ] **Marius Nonsvik:** Regulation of submerged vehicles — due January 25
  - [ ] **Morten Solli:** waterproof solutions; cable, equipment, materials. — due January 25
  - [ ] **Daniel Reite:** force work calculations, strain on cable, etc. — due January 25
  - [ ] **Kristian Salvesen:** Price research on equipment needed. — due January 25
  - [ ] **Morten Solli:** Gyro — due January 25
- [ ] **Morten Solli:** Write thesis
  Make sure that everyone writes on the thesis trough out the project. Everyone should also be writing a logg for each day.
  - [ ] **Morten Solli:** Create Latex document — due January 22
  - [ ] **Morten Solli:** lecturing group latex — due January 24
  - [ ] **Morten Solli:** Write main thesis — due May 31
  - [ ] **Morten Solli:** Start with information part (Part 1) — due January 24
- [ ] **Marius Nonsvik:** Review 1 — due May 7
- [ ] **Daniel Reite:** Review 2 — due May 16
- [ ] **Morten Solli:** Review 3 — due May 23
- [ ] **Morten Solli:** Deliver — due May 31

Construction:

- [ ] **Daniel Reite:** Design

- [ ] **Daniel Reite:** First prototype design — due February 9
- [ ] **Daniel Reite:** Optimize the design

- [ ] **Daniel Reite:** Build — due February 18
  daniel?
- [ ] **Daniel Reite:** Optimize design — due March 20

- [ ] **Daniel Reite:** Test of constructed design
  - [ ] **Daniel Reite:** Water proofing test — due February 20
  - [ ] **Daniel Reite:** Testing of clean design in water tank — due February 23
  - [ ] **Daniel Reite:** Testing of full design in tank — due April 6
  - [ ] **Daniel Reite:** Testing of full design in sea — due April 22
- [ ] **Daniel Reite:** Test of clean optimized design — due March 23

Regulating system:
Making a regulation system for controlling of depth and stability with use of fins.

- [ ] **Morten Solli:** Evaluation — due January 23
- [ ] **Marius Nonsvik:** Simulation — due February 9
- [ ] **Marius Nonsvik:** Design regulation — due March 2
- [ ] **Morten Solli:** Design test system — due March 9
- [ ] **Marius Nonsvik:** Test of regulation
  - [ ] **Marius Nonsvik:** Stability — due March 12
  - [ ] **Marius Nonsvik:** transient time — due March 12
  - [ ] **Marius Nonsvik:** overshoot — due March 12
  - [ ] **Marius Nonsvik:** GPS coordinate test — due March 23
  - [ ] **Morten Solli:** Testing og gyro — due April 6
  - [ ] **Daniel Reite:** buoyancy — due March 21
  - [ ] **Marius Nonsvik:** Regulation — due April 11

Programing:

- [ ] **Kristian Salvesen:** Main program ROV
  - [ ] **Morten Solli:** Decide access to threads — due February 2

- [ ] **Kristian Salvesen:** Protect methods — due April 16
- [ ] **Kristian Salvesen:** create jar file /self start — due April 9
- [ ] **Kristian Salvesen:** create methods to classes — due March 30
- [ ] **Kristian Salvesen:** Connect main program up against source tree — due January 30
- [ ] **Kristian Salvesen:** Create main program shell — due January 31
- [ ] **Morten Solli:** create thread diagram, assign responsibility to classes. — due February 2

- [ ] **Kristian Salvesen:** Lecture on usage of source tree — due January 30

- [ ] **Kristian Salvesen:** Communication
  - [ ] **Marius Nonsvik:** Communication between ROV and support vessel — due April 20
  - [ ] **Kristian Salvesen:** Establish connection internal ROV — due February 9
    Transfer of all data between remote vehicle and support vessel, in between sensors and computer on remote vehicle, storage of data.
  - [ ] **Morten Solli:** Video transfer — due April 6
  - [ ] **Kristian Salvesen:** Reading of sensor values — due March 19
  - [ ] **Kristian Salvesen:** Writing commands to actuators — due March 2
  - [ ] **Morten Solli:** NMEA protocol — due March 26
  - [ ] **Kristian Salvesen:** ROV internal communication — due April 6

- [ ] **Marius Nonsvik:** GUI
  - [ ] **Marius Nonsvik:** GUI design — due February 27
  - [ ] **Marius Nonsvik:** GUI shell — due March 7
  - [ ] **Kristian Salvesen:** Connect GUI up against source tree — due February 28
  - [ ] **Marius Nonsvik:** Extract data from remote vehicle — due April 6
  - [ ] **Marius Nonsvik:** Visualization on GUI — due April 20
  - [ ] **Marius Nonsvik:** Test of GUI controls — due April 25

- [ ] **Kristian Salvesen:** Test of programs and functions
  - [ ] **Kristian Salvesen:** crash test of main program — due April 6
  - [ ] **Kristian Salvesen:** test of individual methods — due March 30

- [ ] **Morten Solli:** test of video storage — due April 9
- [ ] **Morten Solli:** test of videostream — due April 6
- [ ] **Kristian Salvesen:** test of start of main program — due February 2
- [ ] **Kristian Salvesen:** test of individual classes — due March 30
- [ ] **Kristian Salvesen:** test of reading values from external sources — due February 26
- [ ] **Kristian Salvesen:** test of writing to external equipment — due March 5
- [ ] **Marius Nonsvik:** test of communication between main program and GUI program — due April 27

Hardware:

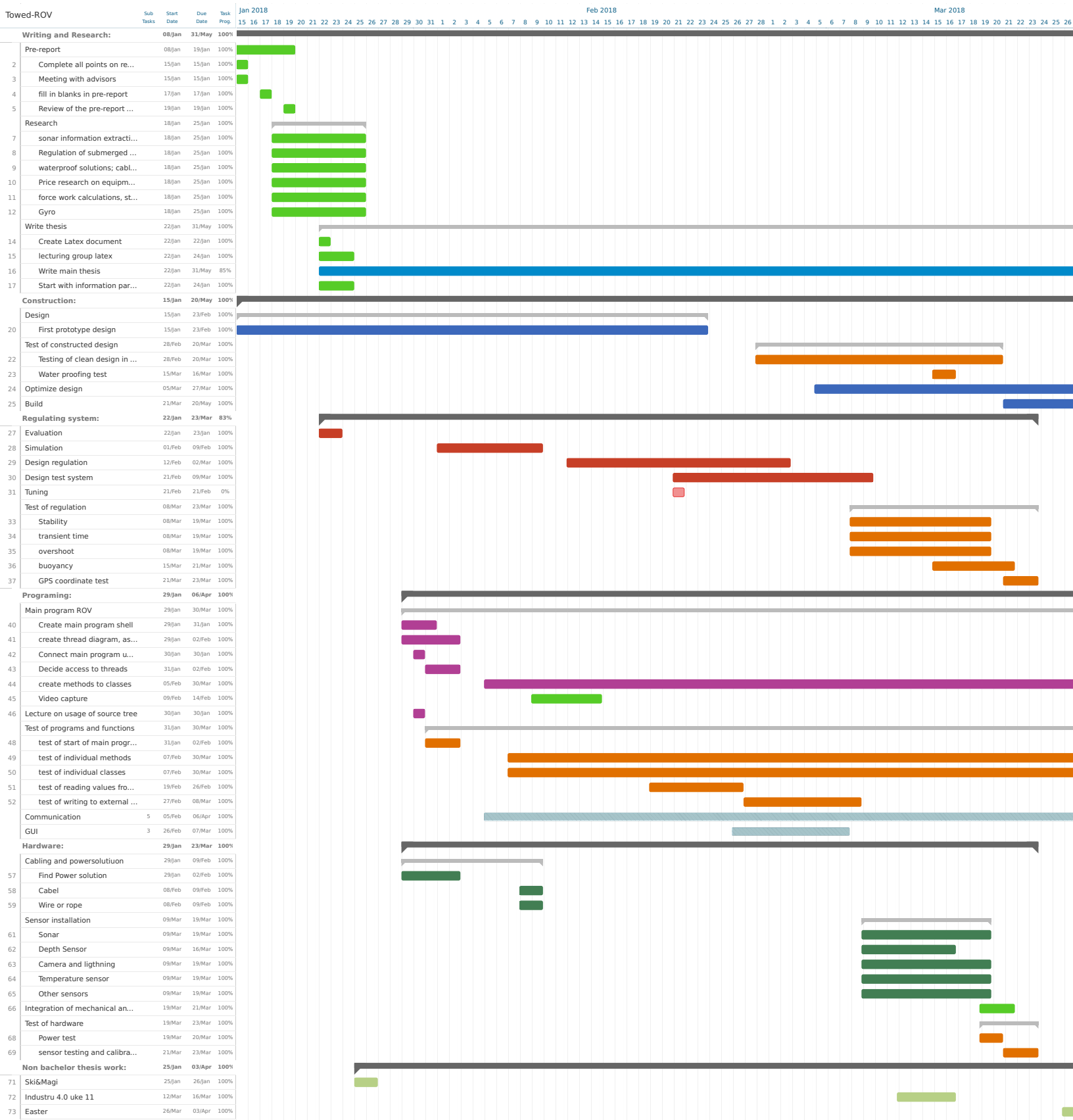
- [ ] **Marius Nonsvik:** Integration of mechanical and electrical part — due March 21
  Integrate the mechanical part of the design with the electrical, actuators with the wings etc.
- [ ] **Morten Solli:** Cabling and powersolutiuon
  - [ ] **Morten Solli:** Find Power solution — due February 2
    which voltage to use and where to get power from.
  - [ ] **Morten Solli:** Cabel — due February 9
    Dimension, Insulation,, Waterproof, Strength.
  - [ ] **Daniel Reite:** Wire or rope — due February 9
  - [ ] **Daniel Reite:** Fitting of cabels — due April 3
    Fitting of cabels on ROV
- [ ] **Morten Solli:** Sensor installation
  mounting of sensors on remote vehicle and support vehicle, connection between sensors and extraction of data to java program
  - [ ] **Morten Solli:** Sonar — due March 19
  - [ ] **Marius Nonsvik:** Depth Sensor — due March 16
  - [ ] **Morten Solli:** Camera — due March 19
  - [ ] **Kristian Salvesen:** Temperature sensor — due March 19
  - [ ] **Kristian Salvesen:** Other sensors — due March 19
- [ ] **Morten Solli:** Test of hardware
  - [ ] **Morten Solli:** sensor testing and calibration — due March 23
  - [ ] **Morten Solli:** Power test — due March 20
    Voltage drop, enough power, etc.
  - [ ] **Marius Nonsvik:** Actuator amd wing test — due April 10
  - [ ] **Morten Solli:** Sonar test — due April 13

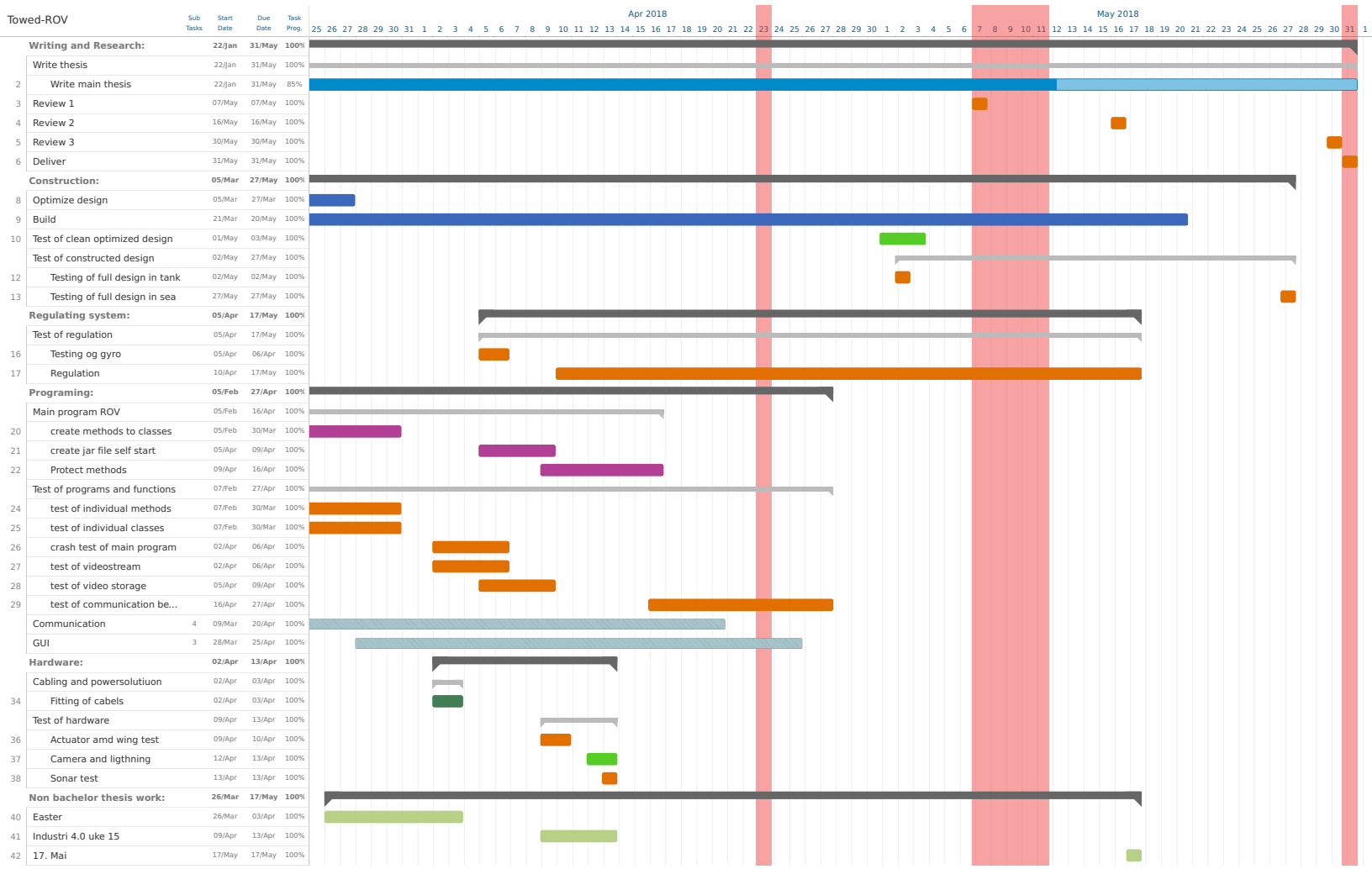
Non bachelor thesis work:

- [ ] **Morten Solli:** Ski&Magi — due January 26
- [ ] **Marius Nonsvik:** Industru 4.0 uke 11 — due March 16
- [ ] **Daniel Reite:** Industri 4.0 uke 15 — due April 13
- [ ] **Kristian Salvesen:** Easter — due April 3
  Work if neccessary but try to take some time off. It is important to relax a little bit.
- [ ] **Kristian Salvesen:** 17. Mai — due May 17

Appendix B

Gannt diagram


Towed-ROV
Sub Tasks
Start Date
Due Date
Task Prog.
Jan 2018
Feb 2018
Mar 2018
15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26
Writing and Research: 08/Jan 31/May 100%
Pre-report 08/Jan 19/Jan 100%
2 Complete all points on re... 15/Jan 15/Jan 100%
3 Meeting with advisors 15/Jan 15/Jan 100%
4 fill in blanks in pre-report 17/Jan 17/Jan 100%
5 Review of the pre-report ... 19/Jan 19/Jan 100%
Research 18/Jan 25/Jan 100%
7 sonar information extracti... 18/Jan 25/Jan 100%
8 Regulation of submerged ... 18/Jan 25/Jan 100%
9 waterproof solutions; cabl... 18/Jan 25/Jan 100%
10 Price research on equipm... 18/Jan 25/Jan 100%
11 force work calculations, st... 18/Jan 25/Jan 100%
12 Gyro 18/Jan 25/Jan 100%
Write thesis 22/Jan 31/May 100%
14 Create Latex document 22/Jan 22/Jan 100%
15 lecturing group latex 22/Jan 24/Jan 100%
16 Write main thesis 22/Jan 31/May 85%
17 Start with information par... 22/Jan 24/Jan 100%
Construction: 15/Jan 20/May 100%
Design 15/Jan 23/Feb 100%
20 First prototype design 15/Jan 23/Feb 100%
Test of constructed design 28/Feb 20/Mar 100%
22 Testing of clean design in ... 28/Feb 20/Mar 100%
23 Water proofing test 15/Mar 16/Mar 100%
24 Optimize design 05/Mar 27/Mar 100%
25 Build 21/Mar 20/May 100%
Regulating system: 22/Jan 23/Mar 83%
27 Evaluation 22/Jan 23/Jan 100%
28 Simulation 01/Feb 09/Feb 100%
29 Design regulation 12/Feb 02/Mar 100%
30 Design test system 21/Feb 09/Mar 100%
31 Tuning 21/Feb 21/Feb 0%
Test of regulation 08/Mar 23/Mar 100%
33 Stability 08/Mar 19/Mar 100%
34 transient time 08/Mar 19/Mar 100%
35 overshoot 08/Mar 19/Mar 100%
36 buoyancy 15/Mar 21/Mar 100%
37 GPS coordinate test 21/Mar 23/Mar 100%
Programing: 29/Jan 06/Apr 100%
Main program ROV 29/Jan 30/Mar 100%
40 Create main program shell 29/Jan 31/Jan 100%
41 create thread diagram, as... 29/Jan 02/Feb 100%
42 Connect main program u... 30/Jan 30/Jan 100%
43 Decide access to threads 31/Jan 02/Feb 100%
44 create methods to classes 05/Feb 30/Mar 100%
45 Video capture 09/Feb 14/Feb 100%
46 Lecture on usage of source tree 30/Jan 30/Jan 100%
Test of programs and functions 31/Jan 30/Mar 100%
48 test of start of main progr... 31/Jan 02/Feb 100%
49 test of individual methods 07/Feb 30/Mar 100%
50 test of individual classes 07/Feb 30/Mar 100%
51 test of reading values fro... 19/Feb 26/Feb 100%
52 test of writing to external ... 27/Feb 08/Mar 100%
Communication 5 05/Feb 06/Apr 100%
GUI 3 26/Feb 07/Mar 100%
Hardware: 29/Jan 23/Mar 100%
Cabling and powersolutiuon 29/Jan 09/Feb 100%
57 Find Power solution 29/Jan 02/Feb 100%
58 Cabel 08/Feb 09/Feb 100%
59 Wire or rope 08/Feb 09/Feb 100%
Sensor installation 09/Mar 19/Mar 100%
61 Sonar 09/Mar 19/Mar 100%
62 Depth Sensor 09/Mar 16/Mar 100%
63 Camera and ligthning 09/Mar 19/Mar 100%
64 Temperature sensor 09/Mar 19/Mar 100%
65 Other sensors 09/Mar 19/Mar 100%
66 Integration of mechanical an... 19/Mar 21/Mar 100%
Test of hardware 19/Mar 23/Mar 100%
68 Power test 19/Mar 20/Mar 100%
69 sensor testing and calibra... 21/Mar 23/Mar 100%
Non bachelor thesis work: 25/Jan 03/Apr 100%
71 Ski&Magi 25/Jan 26/Jan 100%
72 Industru 4.0 uke 11 12/Mar 16/Mar 100%
73 Easter 26/Mar 03/Apr 100%


Towed-ROV
Sub Tasks
Start Date
Due Date
Task Prog.
Apr 2018
May 2018
25 26 27 28 29 30 31 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 1
Writing and Research: 22/Jan 31/May 100%
Write thesis 22/Jan 31/May 100%
2 Write main thesis 22/Jan 31/May 85%
3 Review 1 07/May 07/May 100%
4 Review 2 16/May 16/May 100%
5 Review 3 30/May 30/May 100%
6 Deliver 31/May 31/May 100%
Construction: 05/Mar 27/May 100%
8 Optimize design 05/Mar 27/Mar 100%
9 Build 21/Mar 20/May 100%
10 Test of clean optimized design 01/May 03/May 100%
Test of constructed design 02/May 27/May 100%
12 Testing of full design in tank 02/May 02/May 100%
13 Testing of full design in sea 27/May 27/May 100%
Regulating system: 05/Apr 17/May 100%
Test of regulation 05/Apr 17/May 100%
16 Testing og gyro 05/Apr 06/Apr 100%
17 Regulation 10/Apr 17/May 100%
Programing: 05/Feb 27/Apr 100%
Main program ROV 05/Feb 16/Apr 100%
20 create methods to classes 05/Feb 30/Mar 100%
21 create jar file self start 05/Apr 09/Apr 100%
22 Protect methods 09/Apr 16/Apr 100%
Test of programs and functions 07/Feb 27/Apr 100%
24 test of individual methods 07/Feb 30/Mar 100%
25 test of individual classes 07/Feb 30/Mar 100%
26 crash test of main program 02/Apr 06/Apr 100%
27 test of videostream 02/Apr 06/Apr 100%
28 test of video storage 05/Apr 09/Apr 100%
29 test of communication be... 16/Apr 27/Apr 100%
Communication 4 09/Mar 20/Apr 100%
GUI 3 28/Mar 25/Apr 100%
Hardware: 02/Apr 13/Apr 100%
Cabling and powersolutiuon 02/Apr 03/Apr 100%
34 Fitting of cabels 02/Apr 03/Apr 100%
Test of hardware 09/Apr 13/Apr 100%
36 Actuator amd wing test 09/Apr 10/Apr 100%
37 Camera and ligthning 12/Apr 13/Apr 100%
38 Sonar test 13/Apr 13/Apr 100%
Non bachelor thesis work: 26/Mar 17/May 100%
40 Easter 26/Mar 03/Apr 100%
41 Industri 4.0 uke 15 09/Apr 13/Apr 100%
42 17. Mai 17/May 17/May 100%

Appendix C

Towed-ROV User Manual

Towed-ROV User Manual

This is a user manual and guide for how to use, operate and maintenance the ROV.

First there will be some general information about the ROV, followed by what to do before use, then how to use it, and lastly how do maintenance and install of extra sensors.

Before use:

The following points should be completed before operating the ROV.

- Perform maintenance.
- Attach 36V supply to power lines (+ orange wire) (- yellow wire).
- Connect Fanthom x3 to 12V supply. White infinite dot wire to positive tether interface, remaining white wire to negative tether interface.
- Connect client computer to Fanthom x3.
- Change IP of client computer to local IP: 192.168.0.20.

Operating:

To operate the ROV, follow the instructions bellow.

1. Fasten the cable to the surface vessel.
2. Start client PC.
3. Set IP-address of client PC to: 192.168.0.20
4. Connect USB for the onboard microcontroller.
5. Start client application when ready.
6. Deploy ROV to the water surface.
7. Switch on ROV power supply (36VDC).
8. Wait 2 minutes for the SBC to boot and sensors to calibrate.
9. Video will start when the ROV has booted.
10. Use the “file” option in the GUI-application and click “connect”
11. Enter 192.168.0.10 as the IP-address and click “Ok”.
12. Values will be updating, video is showing, and commands can be sent.
13. Return to step 5 if client fails to connect.
14. Release the ROV into the water and set wanted depth.

Maintenance:

Follow these points to preform maintenance on the ROV.

- Inspection of hardware and connections.
- Check for damage or leaks on the ROV.
- Fill containers with mineral oil if needed.
- Apply lubricant before sealing the lids.
- Apply antirust after use.

Add sensors:

It is easy to attach additional equipment to the ROV. There is a yellow cable from the electronics box which is vulcanized on the end. This cable is connected to the extra I/O Arduino inside. There are 4 cable pairs for extra inputs, 4 pairs for extra outputs and a pair for 5VDC.

To add additional equipment, choose one of the cable pairs listed below and connect the equipment to it.

- White two dots: Channel 1, analog input.
- White three dots: Channel 2, analog input.
- White four dots: Channel 3, analog input.
- White infinite dots: Channel 4, analog input.
- Pink two dots: Channel 5, digital output.
- Pink three dots: Channel 6, digital output.
- Pink four dots: Channel 7, digital output.
- Pink infinite dots: Channel 8, digital output.
- Orange wire + 5V.
- Yellow wire GND.

The channels can be labeled in the option frame of the GUI to match the attached equipment.

The channels can be controlled from the I/O controller frame.

Appendix D

Meetings with Supervisors

Report meeting with advisors

Date: Monday 29.01.18
Place: A433, Main building, NTNU Ålesund

Present: Ottar, Paul Steffen, Kristian, Marius, Daniel, Morten.
Time: 11:00 – 12:00

Issues

- Budget, the group will get what they need within a reasonable amount.
- Design, try to make a design with as little welding as possible. The group should look at solutions with plexiglass, metal pipe spacers, with pretensions thread rods, this will be faster to make and easier to modify. Can also easy make mounting holes. Laser cut parts.
- Tailfin and trim, maybe make two static tailfins with an adjustable spoiler in between for horizontal stability and vertical pitch. A bolt for regulating the angle of the spoiler is sufficient. Think fish design.
- Design and regulation, look at "Mechatronics Concept Designer" for simulating parts with physics, can be connected to MATLAB for simulating of the control system.
- Cable, use higher voltage for less voltage drop and a smaller cross-section of the conductors. It is sufficient with 4 conductors for ethernet 100Mbps. Look at using the cable as rope, here we must take into consideration how much strain the cable can handle and bend radius.
- Actuators, do not need to be linear. It might be easier (and cheaper) to go for rotating actuators.
- Echo sounder, 2 echo sounders will be used. One on the ROV itself and the other on the surface vessel. The echo sounders will be used for regulation and will be compared to each other, the one on the surface vessel will show what's in front of the ROV, the one mounted on the ROV will tell us depth below the ROV. Making a 2d representation of bottom data will be considered.
- Control solution, find out how fast the ROV can rise, this can be done through testing in water. Consider making different regulations for different speeds, one regulation for 2 knops, one for 4 and one for 6 maybe. Consider combining the three different regulations with fuzzy-logic to make it work through all speeds from 2 to 6 knops, this makes the ROV more versatile.

New Business

- Find cable, preferably which can be used as rope too.
- Finish parts lists and order.
- Continuing with scheduled work.

Adjournment
Meeting was adjourned at 11:55 by Kristian. The next general meeting will be at 09:00 - 10:00 on February 12, 2018, in NTNU Aalesund.

Approved by: Morten L. Solli

Report meeting with supervisors

Date: Monday 26.02.18
Place: F423a Main building, NTNU Ålesund

Present: Ottar, Paul Steffen, Marius, Daniel, Morten.
Absent: Kristian.
Time: 09:00 – 10:00

Issues

- What to do, focus on making something that works with regulation and experiment with different solutions to the ROV, preferably things that can argument why we have chosen what we have.
- Modelling; Advisors thinks this is too much work compared to what we gain from it. It might be good to have a simple model, but don't use too much time on it. It will be a very little part of the overall thesis and we should focus on working on several things to fill up the paper with. But since we have started with it a simulation could be made with the use of Ansys Discovery.
- Experiments; Try with different ideas which can be written about in the thesis. Write about everything that has failed and has pictures of everything to include in the report. Experiment with turbulence regarding the wing, look at solutions from nature.
- Clip on wings with T-shaft may have to be fatter to distribute force, or threaded rod.
- Sugeno controller; Start with this already know, such that it is ready, and we can focus on the control system later.
- Use PD regulation with three different controllers for different speeds.
- Actuator Duty cycle; This should not be a problem. Recommends monitoring the duty cycle and have a warning for if it is close to maximum. But it shouldn't be a problem regarding that we are not at maximum load, under water and not constantly actuate.
- Thesis; Focus on getting everything into the thesis. Use the time wisely and focus on working on the tasks which will generate much quality content. Focus both to get a good length on the thesis and quality content
- Ottar thinks we are on track and that we have done very much good work, the only thing he doesn't want us to focus on is the modelling because this is too much work regarding what it gives back.

New Business

- Simulation and visualization.

- Look at NMEA and echo sounder.
- Build.
- Experiments and tests.
- Sugeno controller.
- Continuing with scheduled work.

Adjournment
Meeting was adjourned at 10:00 by Ottar. The next general meeting will be at 09:00 - 10:00 on March 12, 2018, in NTNU Aalesund.

Approved by: Morten L. Solli

Report meeting with supervisors

Date: Monday 13.02.18
Place: B311, Main building, NTNU Ålesund

Present: Ottar, Paul Steffen, Kristian, Marius, Daniel, Morten.
Time: 09:00 – 09:31

Issues

- Design
    - Start with using plexiglass for the frame, then later move on to stronger material if necessary.
    - Hydrodynamic profiles for wings.
- Actuator
    - Linear actuators are okay to use. The main reason the supervisors didn't recommend it was that they thought the price was higher than rotating.
    - Worm gear. The idea of making a worm gear to rotate the wings, sturdy but slow solution. Most then use rotating actuator/stepper motor, but they do not need to be very strong.
    - Rotating actuator with planetary gear. If we use rotating motors with low torque planetary gear can be used to increase the power of these.
    - Oil filled chamber. Making an oil filled chamber to house the actuators and transferring system.

- Control system
    - Fuzzy logic. Use different actuators combined with a fuzzy controller which have inputs for speed and depth, it will then calculate how much of each controller it will use.
    - Approximate ROV by finding approximated graphs and coefficients from water tank test.
- Thesis
    - Theoretical basis should be very general about things used in the thesis. Do not explain much about Java and similar things, just that they are used.
    - Method and Material shall be about how we have applied the theory described in the chapter above and why we have chosen the solution we did. Ex; Theory: What is a NACA-profile, M&M: We used the NACA profile like this, why we did it and our calculations.

New Business

- Finishing orders.
- Start to build a prototype and test sealings.

Adjournment

Meeting was adjourned at 09:31 by Ottar. The next general meeting will be at 09:00 - 10:00 on February 26, 2018, in NTNU Aalesund.

Approved by: Morten L. Solli

Report meeting with supervisors

Date: Monday 12.03.18
Place: A433 Main building, NTNU Ålesund

Present: Ottar, Paul Steffen, Marius, Kristian, Morten.
Absent: Daniel.
Time: 09:00 – 10:00

Issues

- 3D Printing; Print with 20% infill, 100% wastes time and material, especially in the prototyping phase. This makes it possible to print longer profiles. Can result in the air within the print which will create pressure.
  Increase resolution of exported file in MX to get smoother surfaces.
- Wings; Make another set of wing profile to compare which is best. Look at the NACA rother profiles which have neutral buoyancy.
- Front cover; Paul Steffen recommends starting out with plexiglass here, however, 3D-printed covers can be an alternative, but it needs to be printed in modules.
- Electronics boxes and actuator sealing; Can buy boxes from "Clas Ohlson" with gaskets and levers for electronic boxes, this will possibly be sufficient with the use of cable fittings (needs to be tested). An alternative to this would be to 3D-print the boxes with grooves for sealings. Check what is possible for actuators.
- Foil; Foil can be used to smoothen surfaces such as wings and ruder.
- Waterproofing; boxes can be filled with either wax or epoxy if they don't have to be re-opened.
- Check out; Efoil community and electronic surfboard forums for ideas.
- PID and Sugeno; Do not have 3 different PID controllers, this will make the system unstable. Use 1 PID controller where the 3 different gains(Kp, Ti, Td) are changed based on the sugeno's output. 3 different Sugeno formulas are required for this, one for each gain.
  Doing it this way will give a more stable controller because switching PID's will result in large errors where the controller will try to stabilize.
- Log and display sensor data; Make channels for sensor logging, where a sensor is automatically assigned a channel when connected. Should consider a way to change the name of the channel after a sensor is connected. 8 channels should be sufficient.
- Create a separate logging for sensor data (Notepad, Excel, etc.).

- Actuators; Maybe no need to use PWM, use a position controller with an integrator and use a set voltage to the actuators, no PWM needed.
-

New Business

- Continue building.
- Experiments and tests.
- Sugeno controller.
- Try other profiles.
- Sensor data and logging.
- Actuators.
- Continuing with scheduled work.

Adjournment

The meeting was adjourned at 10:00 by Ottar. The next general meeting will be at 09:00 10:00 on March 26, 2018, in NTNU Aalesund.

Approved by: Morten L. Solli

Report meeting with supervisors

Date: Monday 04.04.18
Place: F432a Main building, NTNU Ålesund

Present: Ottar, Paul Steffen, Marius, Kristian, Morten, Daniel.
Absent:
Time: 09:00 – 10:00

Issues

- Problem with computers on FabLab, after the IT-department upgraded the computers the Monday before Easter none of the software has been reinstalled. This makes it impossible for us to mill circuit boards and cut out pieces.
- Screw sleeves for the camera house keep falling out. An alternative here is to use UV- based glue or screws through the bracket with nuts on the other side.
- Waterproof boxes for actuators with stuffing box, alternatively use PG-nipples. The box can be 3D-printed.
- Camera stream is done through python because no Java library supports the I2C camera for capturing video. There will be a UDP stream for video streaming and TCP connection for sending commands.
- Echo sounder and GPS to be on the same Arduino, use UART soft serial inputs or Arduino/Teensy with more UARTS.
- I2C sensor → Arduino I2C master/slave → Raspberry I2C Master is a bad solution and causes problems in about 50% of the cases, connects instead the I2C sensors directly to the RBPi.
- I/O. Have about 8 flexible analogue inputs (can also handle digital inputs), where analogue sensors can be connected directly and start to display values at once and the possibility to set names in the GUI. Priority two is to also have digital outputs, in the example to take water tests with opening and closing a chamber.
  Use an Arduino for the I/O's, if anything gets destroyed it will be the Arduino and not the RBPi. Send sensor values all the time, but only look for the interesting ones in the GUI. Control outputs from GUI can implement extra functionality and easy GUI to use.
- Cable. Swend higher voltage through the cable. Test how long it takes for the cable to gain high temperatures under different loads. Because of water, there is some cooling, the whole cable won't be in the water.
- Interferences could be a problem if the power consumption rapidly changes. To avoid this, we should install a battery, large capacitor and a small capacitor in the ROV to avoid changing power draw from the surface vessel.

- Simulate the regulator in Simulink with step input and ramp input, use a first order LP-filter as a load.
- Test sensors and log data.
- Utilize sketches wherever it can be used. One should be a sketch showing the hole system, then have two different zoomed in on the boat and the ROV, showing the different parts on each side. This should come very early in the thesis.
- Introduction. Look from above then go into details. “Men have always wanted to make maps, much have been done by satellites drones and UAV/ROVs on land. But there is very little done under water, we build a towed ROV to achieve to do this on the ocean floor.”
  Why is this a good idea? Everyone should understand the summary, but not everyone should be able to understand the thesis.
- No Javadoc in the thesis, this should come in an appendix. In the thesis write more general what the code is doing and how it is done, don’t mention any understandable class names. Use blocks and sketches to show how parts of the code are tied together and where data is sent.
- Use www.draw.io to make sketches and so on.

New Business

- Experiments and tests.
- Try other profiles.
- Inputs, outputs; digital(second priority).
- Actuators.
- Continuing with scheduled work.
- Make sketches.
- Simulate in Simulink ramp/step input.

Adjournment

The meeting was adjourned at 10:30 by Kristian. The next general meeting will be at 09:00 -10:00 on April 16, 2018, in NTNU Aalesund.

Approved by: Morten L. Solli

Report meeting with supervisors

Date: Monday 24.04.18
Place: A433 Main building, NTNU Ålesund

Present: Paul Steffen, Marius, Kristian, Daniel, Morten.
Absent: Ottar.
Time: 09:00 – 10:00

Issues

- Small leakage after tank test, this might not be crucial.
- Software mainly finished, will perform dry test.
- Explain test with figures.
- Take loots of pictures, video to presentation. Show tests and simulations. Do colors and rendering in 3D. A good presentation is key.

New Business

- Experiments and tests.
- Make videos and good figures.
- Continuing with scheduled work.

Adjournment

The meeting was adjourned at 10:00 by Ottar. The next general meeting will be at 09:00 10:00 on April 24, 2018, in NTNU Aalesund.

Approved by: Morten L. Solli

Report meeting with supervisors

Date: Monday 15.05.18
Place: L160 Lab building, NTNU Ålesund

Present: Ottar, Marius, Kristian, Daniel, Morten.
Absent: Paul Steffen.
Time: 09:00 – 10:00

Issues

- Problem with voltage spikes and ethernet.
- New motor driver and dc/dc convert ordered.
- DC powerline adapter for ethernet already ordered and waiting for.

New Business

- Finish prototype and test in the ocean.
- Finish thesis.

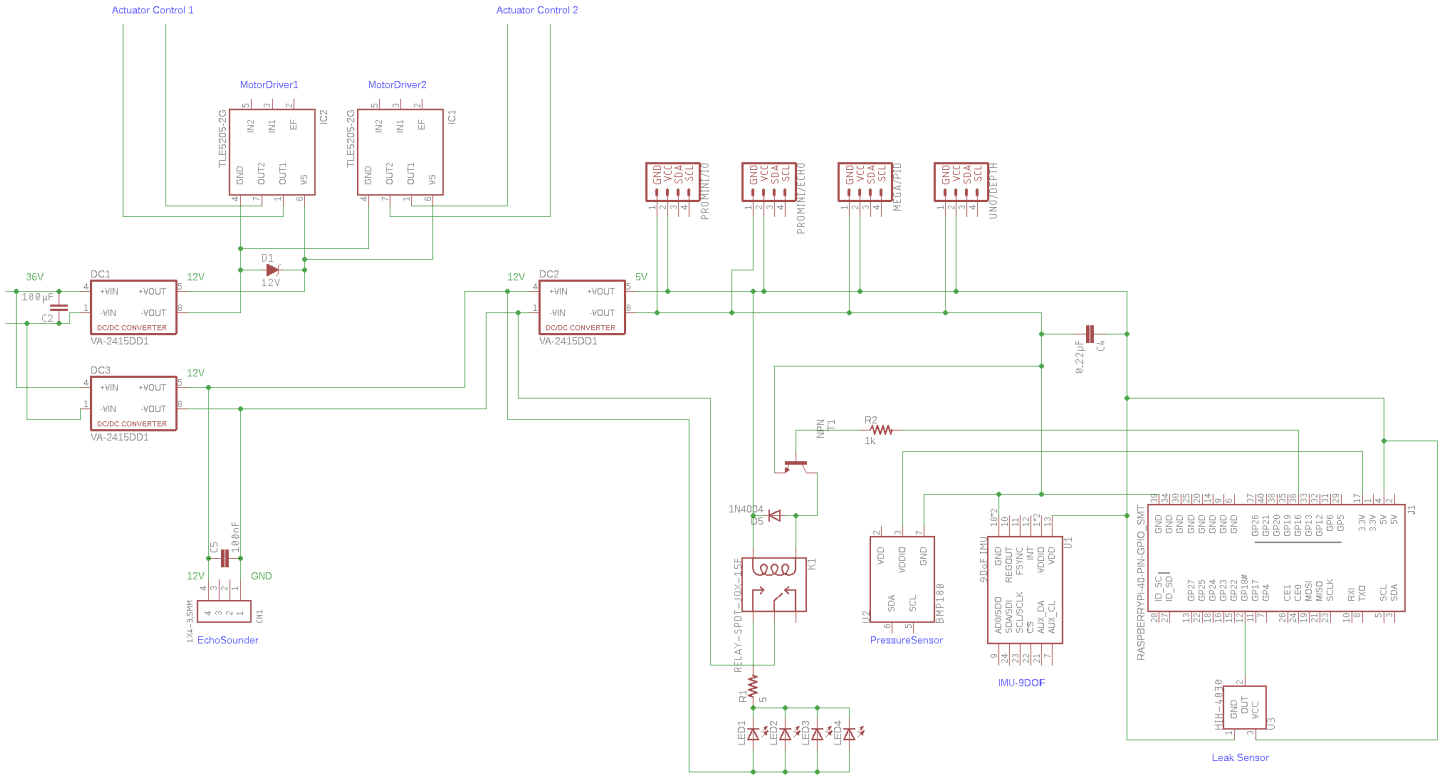
Adjournment

The meeting was adjourned at 10:00 by Ottar. The next general meeting will be at 09:00 10:00 on May 15, 2018, in NTNU Aalesund.

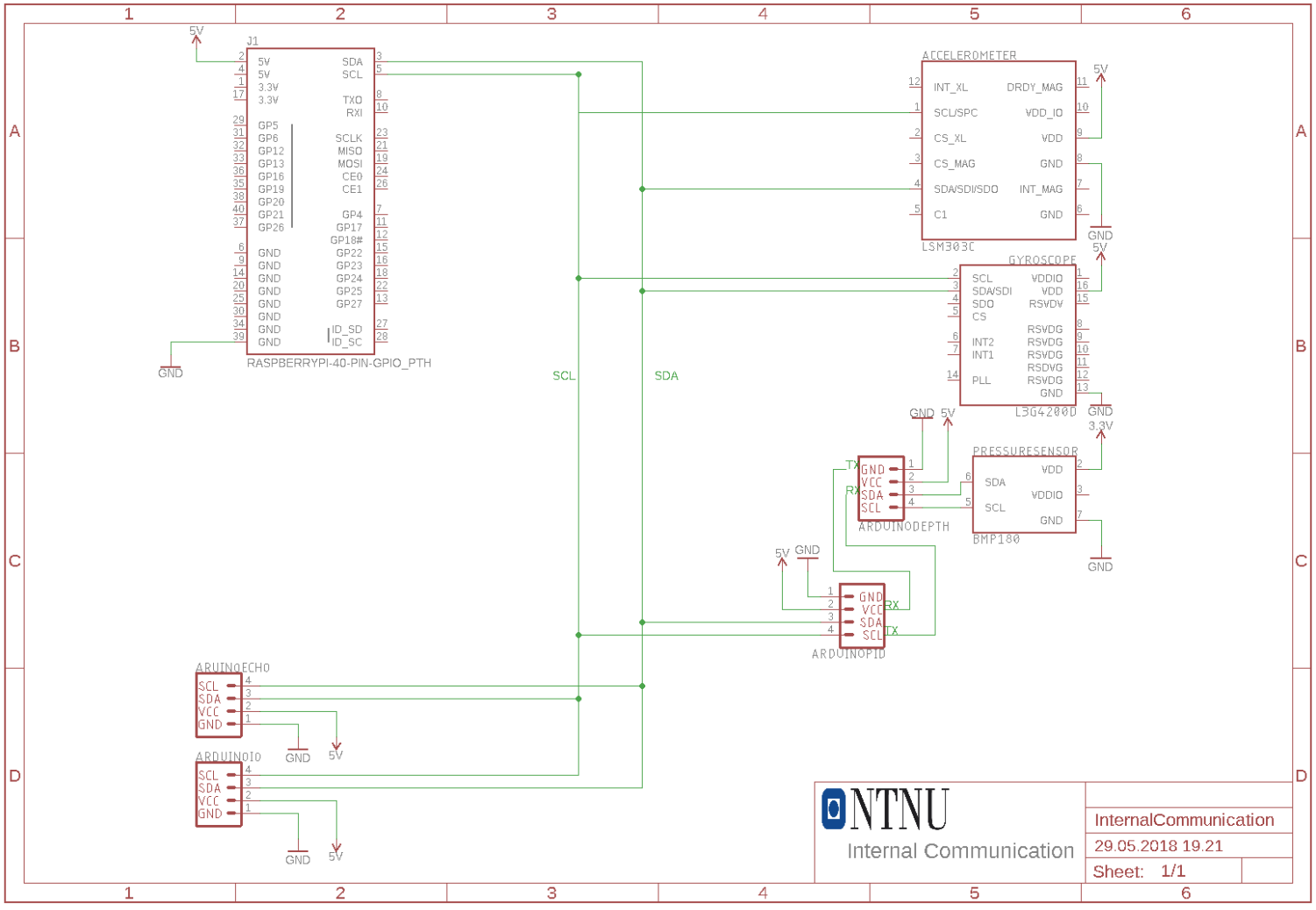
Approved by: Morten L. Solli

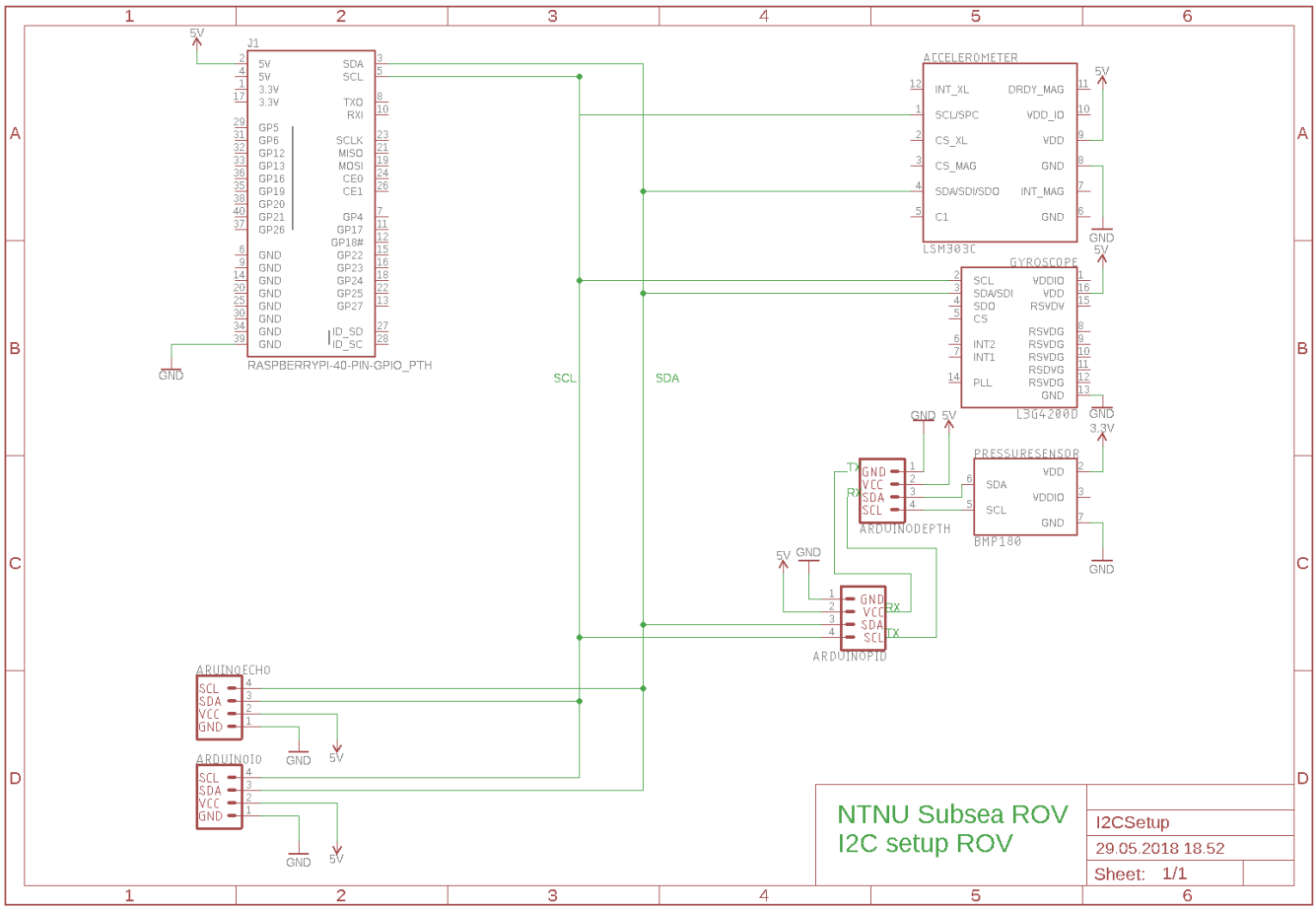
Appendix E

Wiring Diagrams



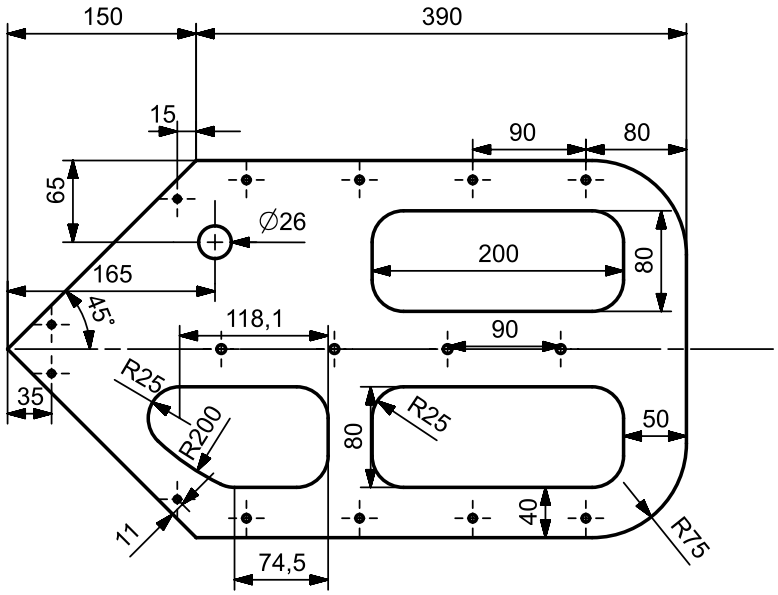
NTNU	
Electrical supply diagram ROV	ElDrawing
	29.05.2018 19.17
	Sheet: 1/1

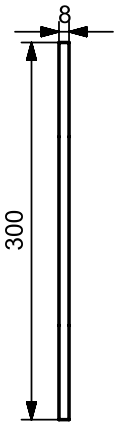

1
2
3
4
5
6
A
B
C
D
5V
J1
5V
5V
3.3V
3.3V
SDA
SCL
TXD
RXI
GP5
GP6
GP12
GP13
GP16
GP19
GP20
GP21
GP26
SCLK
MISO
MOSI
CE0
CE1
GP4
GP17
GP18#
GP22
GP23
GP24
GP25
GP27
GND
GND
GND
GND
GND
GND
GND
ID_SD
ID_SC
RASPBERRYPI-40-PIN-GPIO_PTH
SCL
SDA
ACCELEROMETER
INT_XL
SCL/SPC
CS_XL
CS_MAG
SDA/SDI/SDO
C1
DRDY_MAG
VDD_IO
VDD
GND
INT_MAG
GND
LSM303C
GYROSCOPE
SCL
SDA/SDI
SDO
CS
INT2
INT1
PLL
VDDIO
VDD
RSVD
RSVD
RSVD
RSVD
RSVD
GND
L3G4200D
PRESSURESENSOR
SDA
SCL
VDD
VDDIO
GND
BMP180
GND
VCC
SDA
SCL
TX
RX
ARDUINODEPTH
ARDUINOPID
ARDUINOECHO
ARDUINOIO
NTNU
Internal Communication
InternalCommunication
29.05.2018 19.21
Sheet: 1/1


J1
RASPBERRYPI-40-PIN-GPIO_PTH
ACCELEROMETER
LSM303C
GYROSCOPE
L3G4200D
PRESSURESENSOR
BMP180
ARDUINODEPTH
ARDUINOPID
ARDUINOECHO
ARDUINOIO
SCL
SDA
NTNU Subsea ROV
I2C setup ROV
I2CSetup
29.05.2018 18.52
Sheet: 1/1

Appendix F

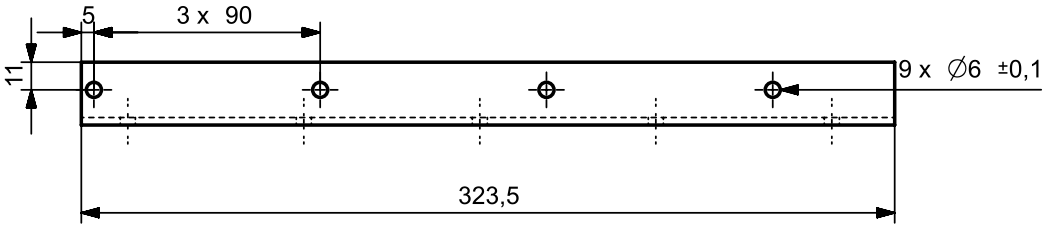
Mechanical Drawings

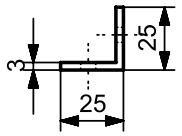


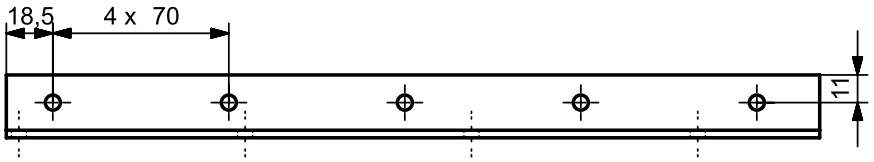


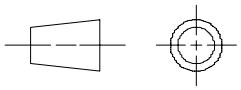
NTNU NTNU Aalesund Larsgardsvegen 2 6009 Aalesund Norway	This drawing is the property of NTNU Aalesund. The content can not be made public, copied or otherwise used without our written consent.	Drawn By: Daniel R. Reite	Date: 29.05.2018
	Part Name / Part Descritption: Sideplate	Sheet: A3 - 1 of 1	Scale: 1:4
		Part No / ID No: 1	Revision: A

A3



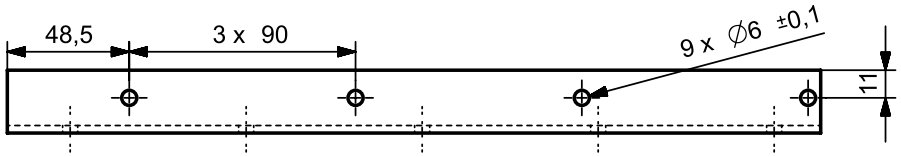


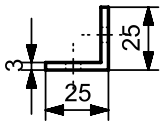


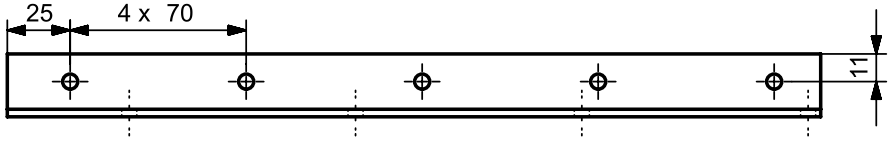


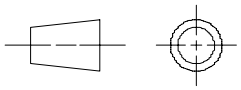
NTNU NTNU Aalesund Larsgardsvegen 2 6009 Aalesund Norway	This drawing is the property of NTNU Aalesund. The content can not be made public, copied or otherwise used without our written consent.	Drawn By: Daniel R. Reite	Date: 29.05.2018
	Part Name / Part Descritption: Lower left angle	Sheet: A3 - 1 of 1	Scale: 1:2
		Part No / ID No: 2	Revision: A

A3

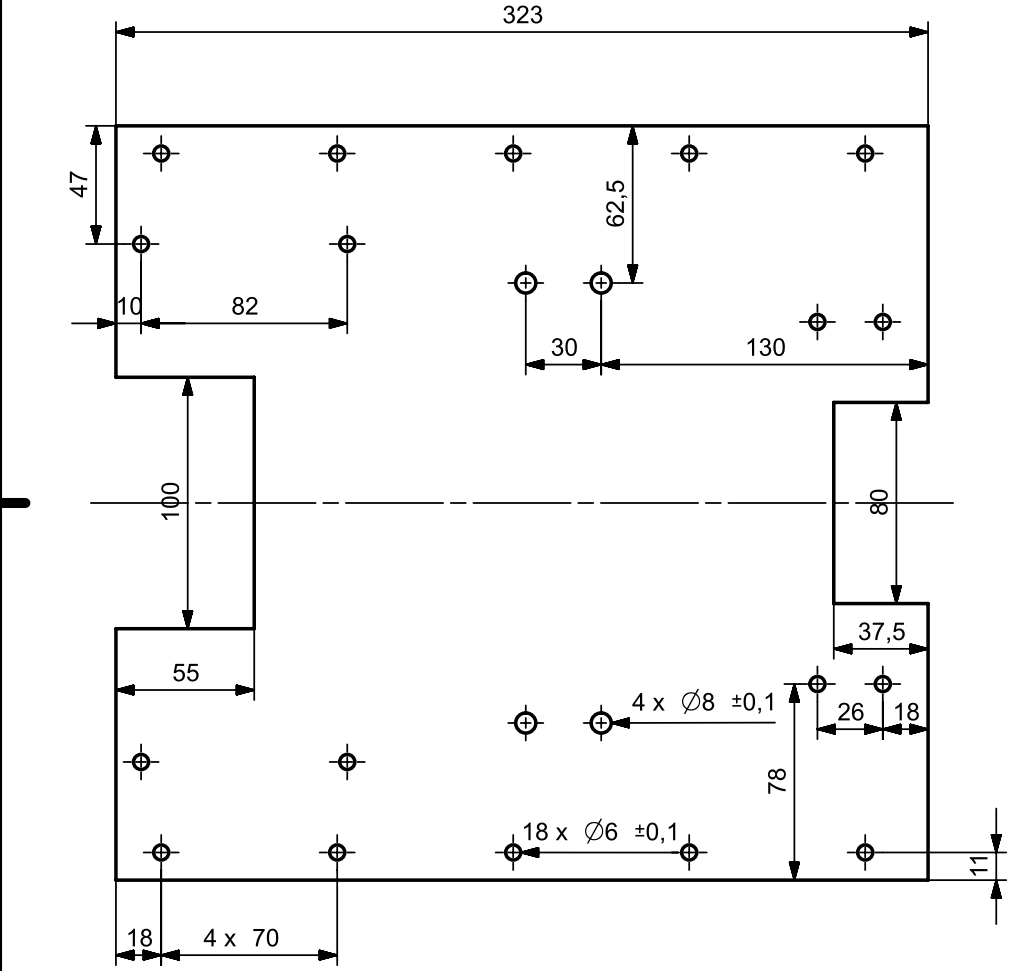


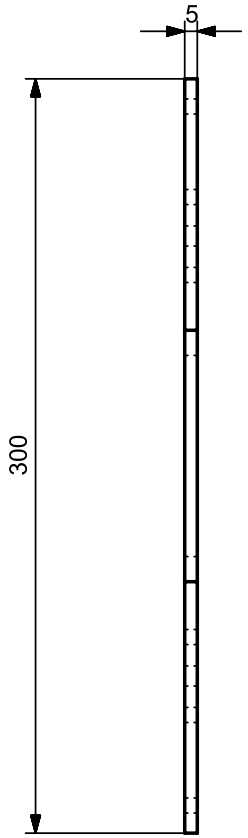


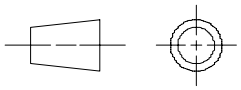




NTNU NTNU Aalesund Larsgardsvegen 2 6009 Aalesund Norway	This drawing is the property of NTNU Aalesund. The content can not be made public, copied or otherwise used without our written consent.	Drawn By: Daniel R. Reite	Date: 29.05.2018
	Part Name / Part Descritption: Lower right angle	Sheet: A3 - 1 of 1	Scale: 1:2
		Part No / ID No: 3	Revision: A





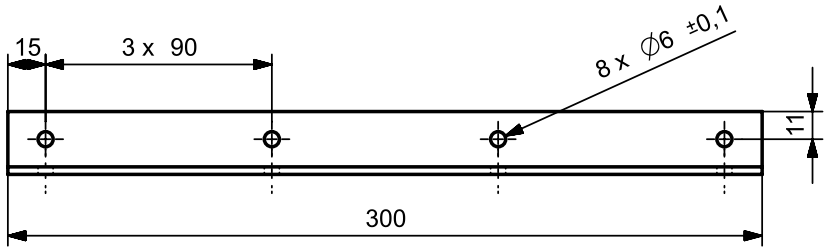


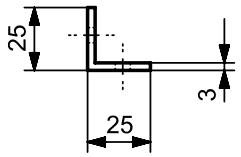
NTNU
NTNU Aalesund
Larsgardsvegen 2
6009 Aalesund
Norway

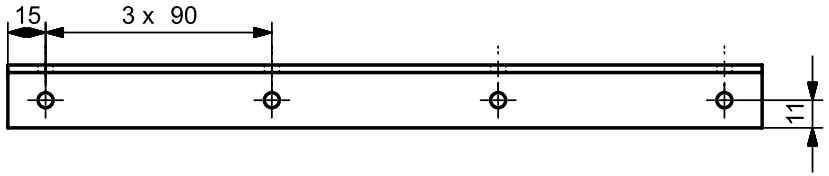
This drawing is the property of NTNU Aalesund. The content can not be made public, copied or otherwise used without our written consent.

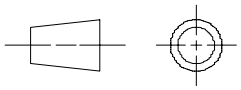
Drawn By:	Daniel R. Reite
Date:	29.05.2018
Part Name / Part Descritption	Lower plate
Sheet:	A3 - 1 of 1
Scale:	1:2
Part No / ID No:	4
Revision:	A

A3

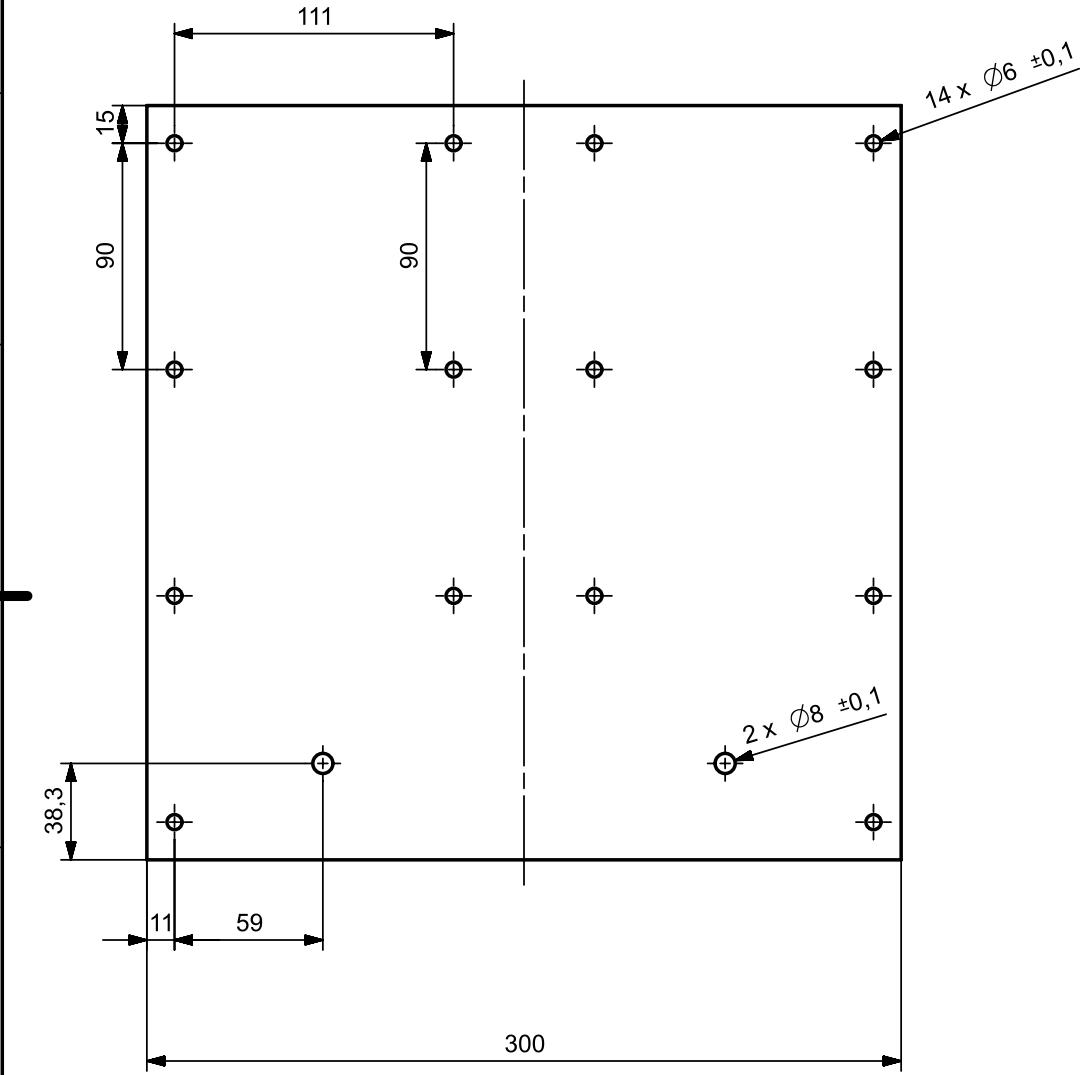


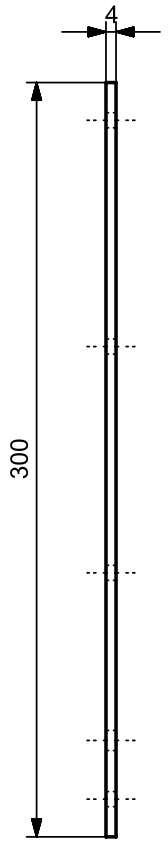






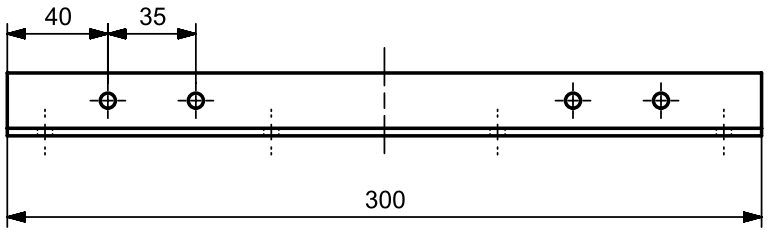
NTNU NTNU Aalesund Larsgardsvegen 2 6009 Aalesund Norway	This drawing is the property of NTNU Aalesund. The content can not be made public, copied or otherwise used without our written consent.	Drawn By: Daniel R. Reite	Date: 29.05.2018
	Part Name / Part Descritption: Middle angle	Sheet: A3 - 1 of 1	Scale: 1:2
		Part No / ID No: 5	Revision: A

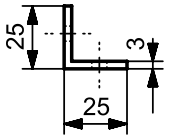


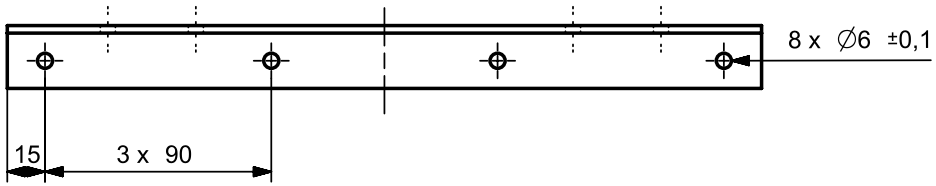


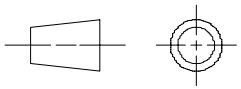
NTNU NTNU Aalesund Larsgardsvegen 2 6009 Aalesund Norway	This drawing is the property of NTNU Aalesund. The content can not be made public, copied or otherwise used without our written consent.	Drawn By: Daniel R. Reite	Date: 29.05.2018
	Part Name / Part Descritption Middle plate	Sheet: A3 - 1 of 1	Scale: 1:2
		Part No / ID No: 6	Revision: A

A3



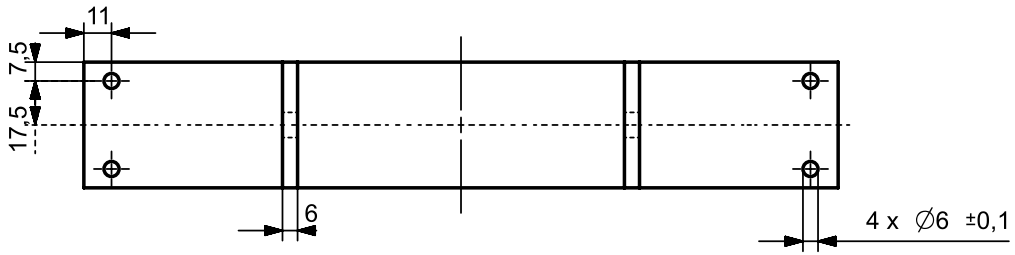


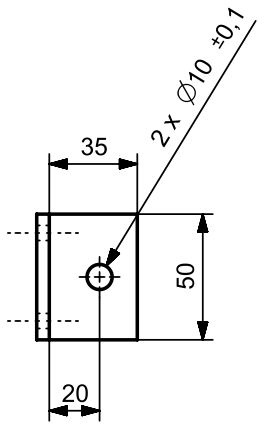


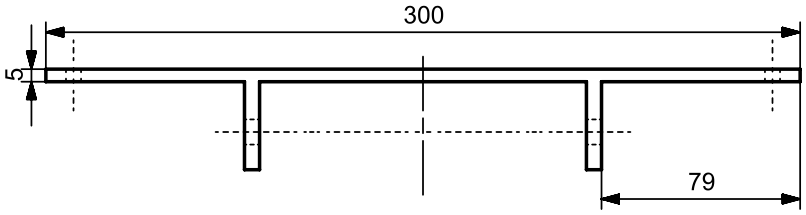


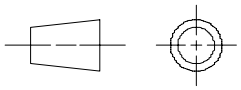
NTNU NTNU Aalesund Larsgardsvegen 2 6009 Aalesund Norway	This drawing is the property of NTNU Aalesund. The content can not be made public, copied or otherwise used without our written consent.	Drawn By: Daniel R. Reite	Date: 29.05.2018
	Part Name / Part Descritption Upper angle	Sheet: A3 - 1 of 1	Scale: 1:2
		Part No / ID No: 7	Revision: A

A3



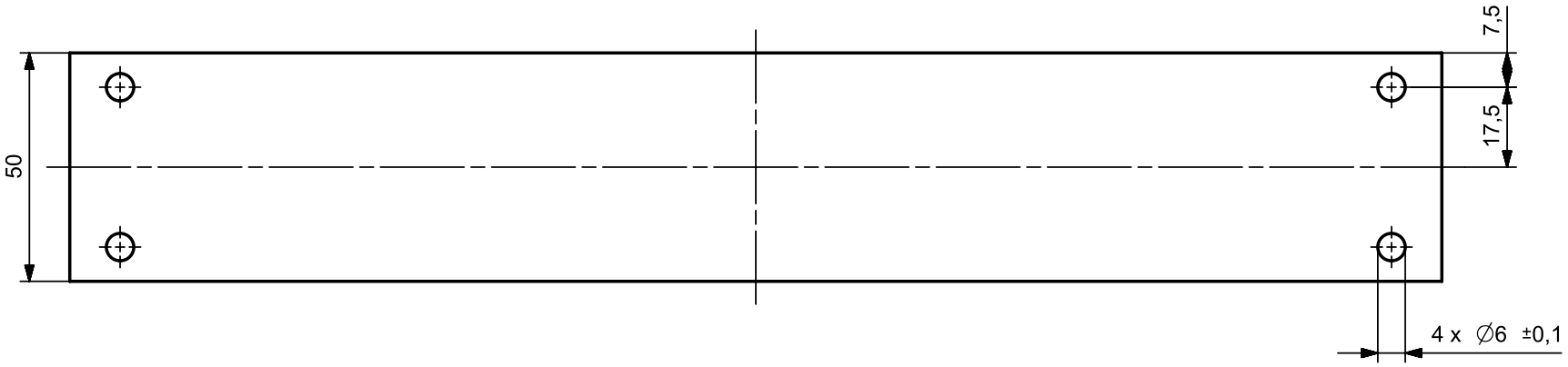


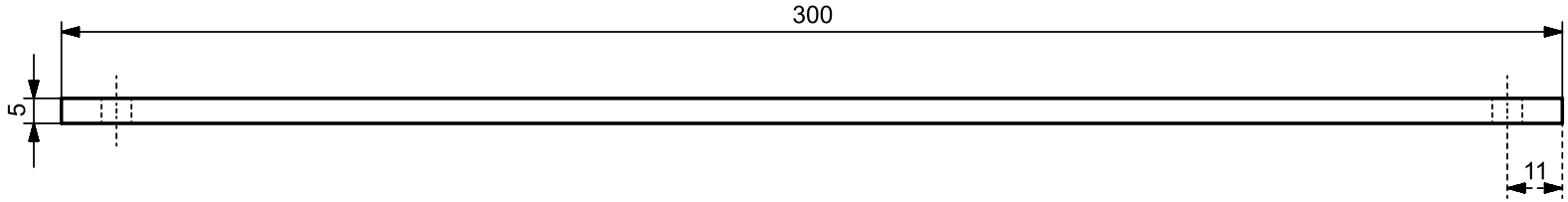


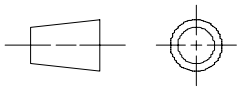


NTNU NTNU Aalesund Larsgardsvegen 2 6009 Aalesund Norway	This drawing is the property of NTNU Aalesund. The content can not be made public, copied or otherwise used without our written consent.	Drawn By: Daniel R. Reite	Date: 29.05.2018
	Part Name / Part Descritption: Upper front plate	Sheet: A3 - 1 of 1	Scale: 1:2
		Part No / ID No: 8	Revision: A

A3

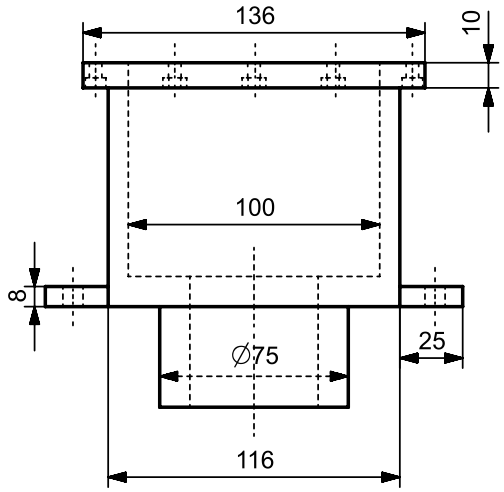


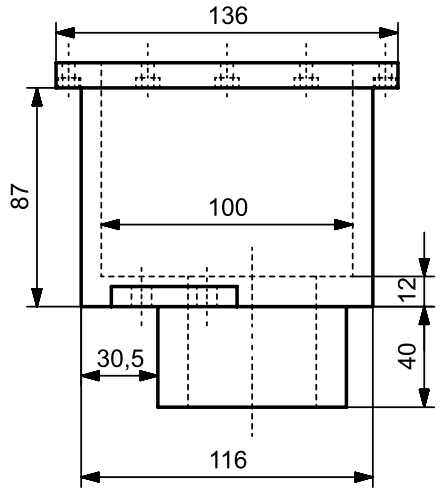


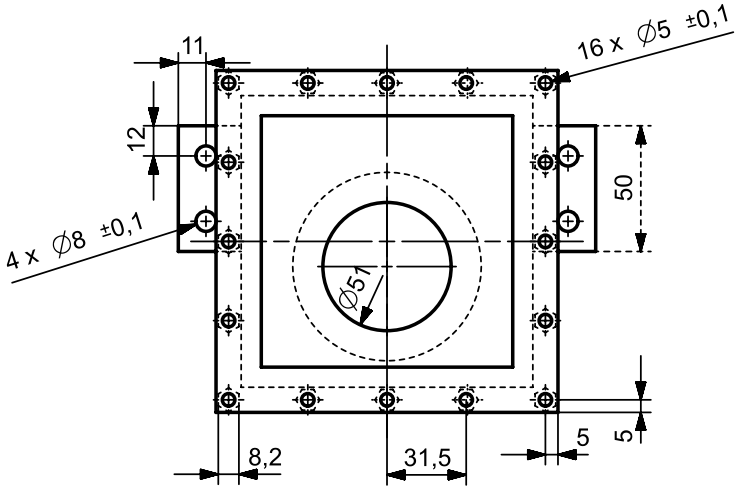


NTNU NTNU Aalesund Larsgardsvegen 2 6009 Aalesund Norway	This drawing is the property of NTNU Aalesund. The content can not be made public, copied or otherwise used without our written consent.	Drawn By: Daniel R. Reite	Date: 29.05.2018
	Part Name / Part Descritption Upper back plate	Sheet: A3 - 1 of 1	Scale: 1:1
		Part No / ID No: 9	Revision: A

A3

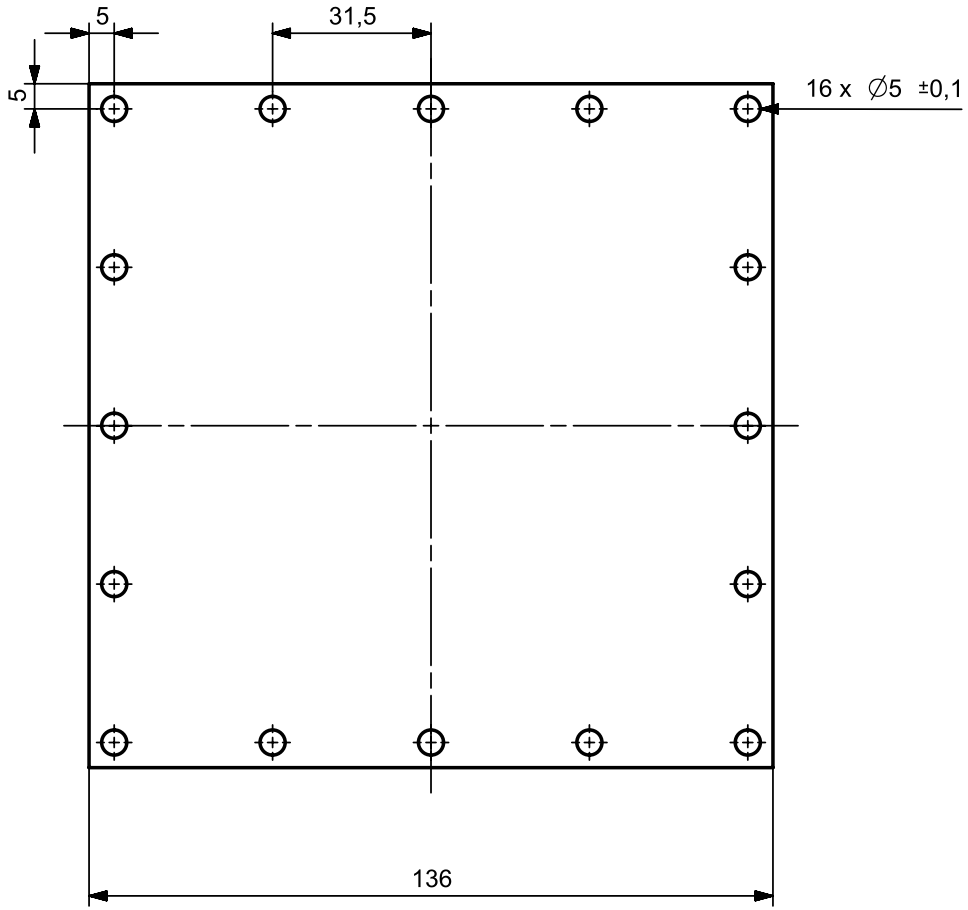


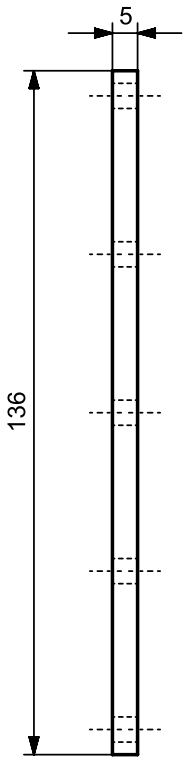


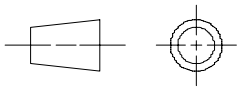


NTNU NTNU Aalesund Larsgardsvegen 2 6009 Aalesund Norway	This drawing is the property of NTNU Aalesund. The content can not be made public, copied or otherwise used without our written consent.	Drawn By: Daniel R. Reite	Date: 29.05.2018
	Part Name / Part Descritption: Echo sounder box	Sheet: A3 - 1 of 1	Scale: 1:2
		Part No / ID No: 10	Revision: A

A3

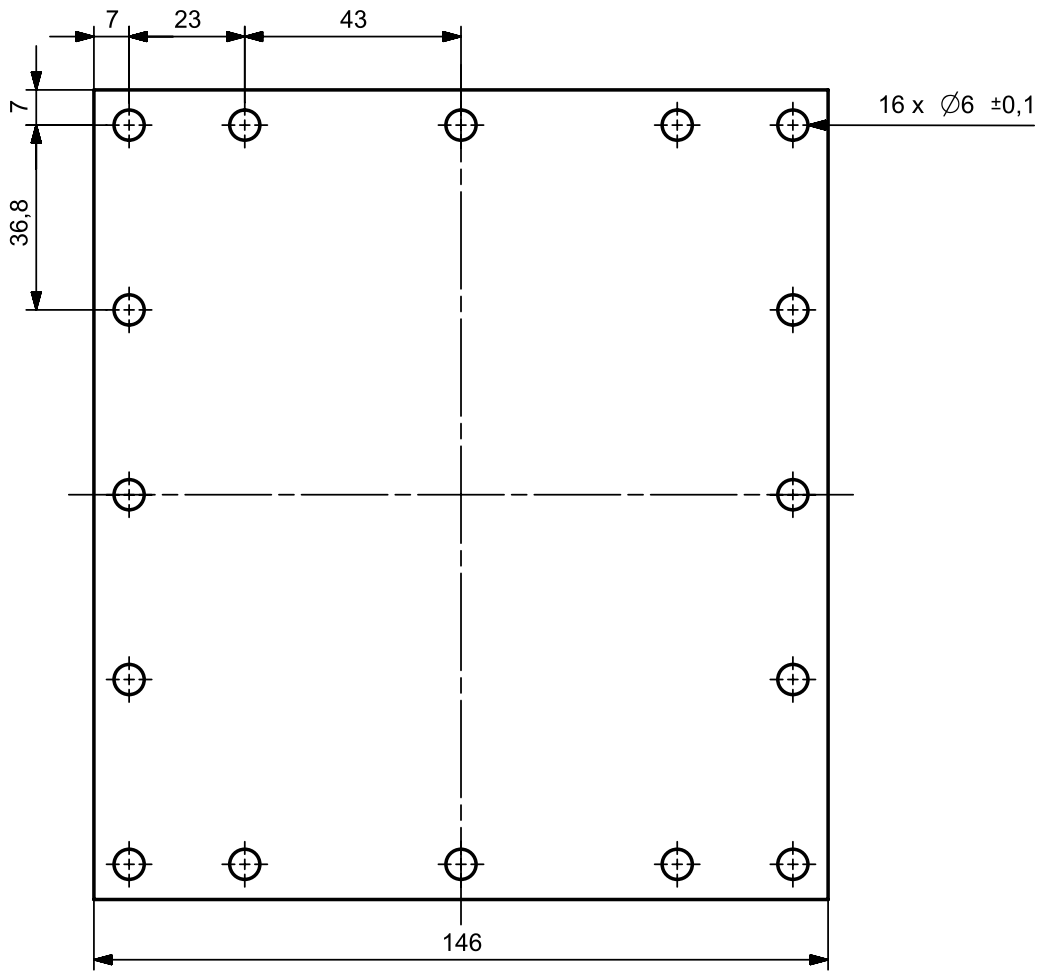


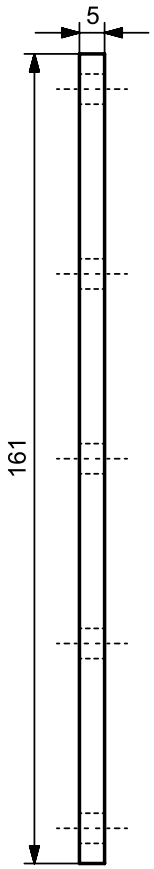


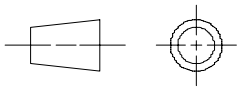


NTNU NTNU Aalesund Larsgardsvegen 2 6009 Aalesund Norway	This drawing is the property of NTNU Aalesund. The content can not be made public, copied or otherwise used without our written consent.	Drawn By: Daniel R. Reite	Date: 29.05.2018
	Part Name / Part Descritption: Lid for echo sounder box	Sheet: A3 - 1 of 1	Scale: 1:1
		Part No / ID No: 11	Revision: A

A3

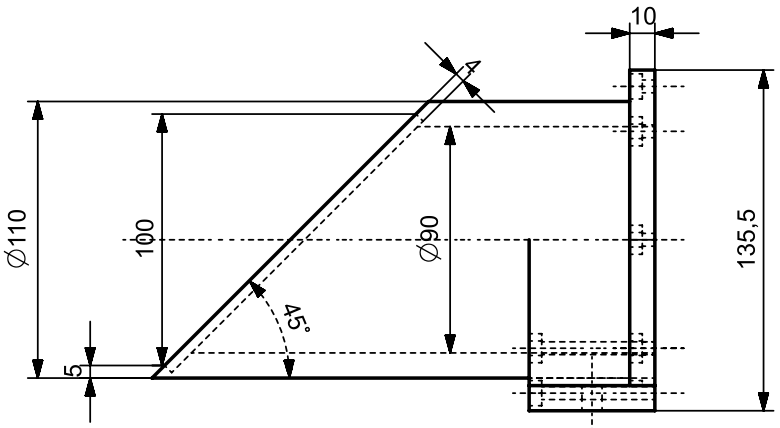


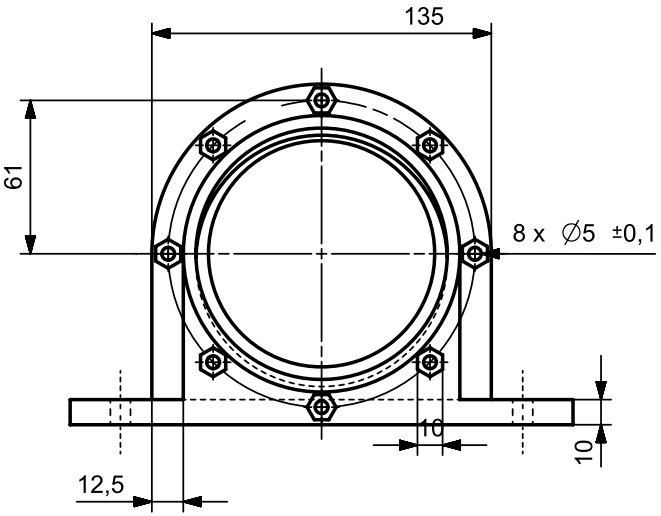


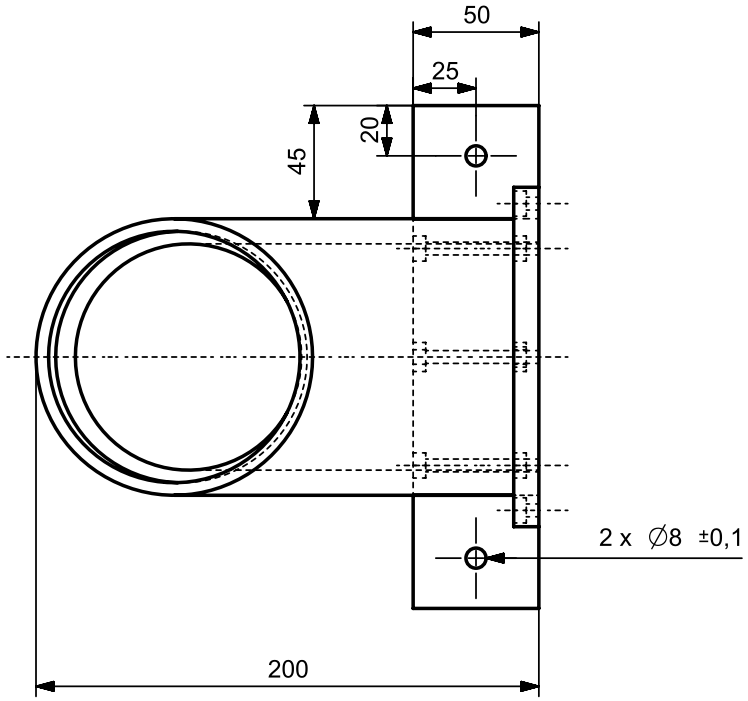


NTNU NTNU Aalesund Larsgardsvegen 2 6009 Aalesund Norway	This drawing is the property of NTNU Aalesund. The content can not be made public, copied or otherwise used without our written consent.	Drawn By: Daniel R. Reite	Date: 29.05.2018
	Part Name / Part Descritption Lid for electronics box	Sheet: A3 - 1 of 1	Scale: 1:1
		Part No / ID No: 12	Revision: A

A3

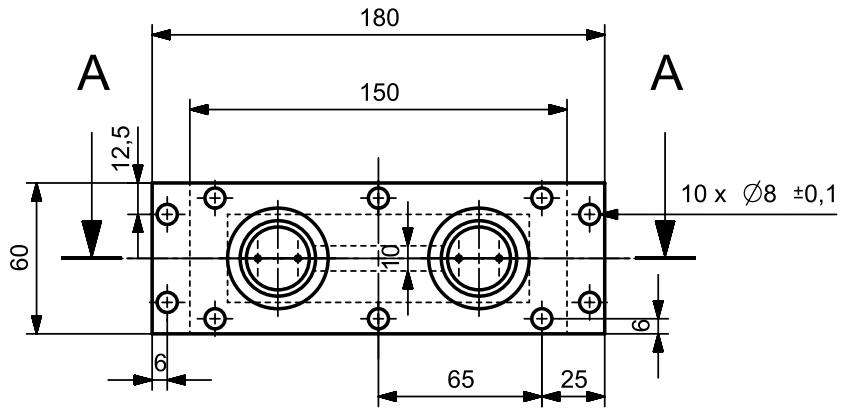


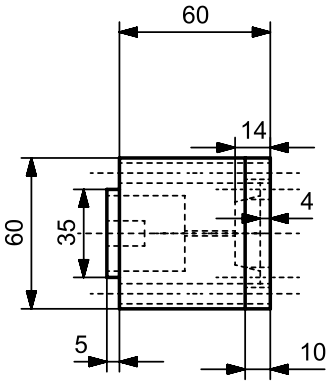


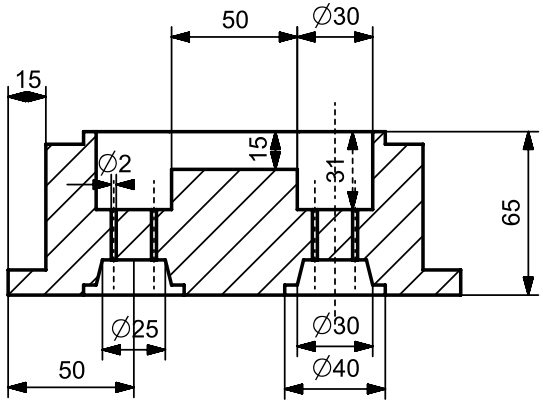


NTNU NTNU Aalesund Larsgardsvegen 2 6009 Aalesund Norway	This drawing is the property of NTNU Aalesund. The content can not be made public, copied or otherwise used without our written consent.	Drawn By: Daniel R. Reite	Date: 29.05.2018
	Part Name / Part Descritption: Camera housing	Sheet: A3 - 1 of 1	Scale: 1:2
		Part No / ID No: 13	Revision: A

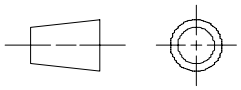
A3





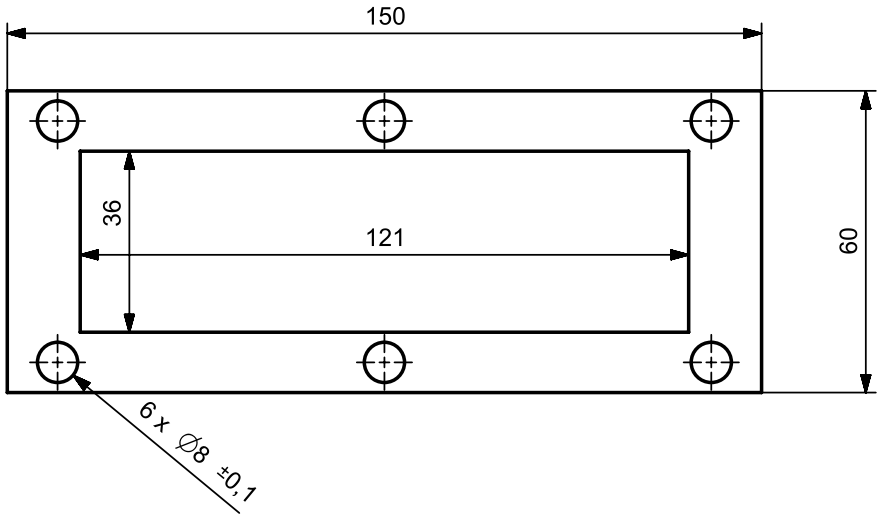


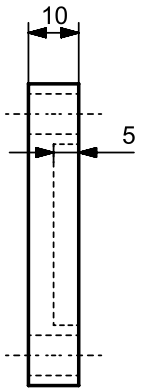
SECTION A-A



NTNU NTNU Aalesund Larsgardsvegen 2 6009 Aalesund Norway	This drawing is the property of NTNU Aalesund. The content can not be made public, copied or otherwise used without our written consent.	Drawn By: Marius Nonsvik	Date: 29.05.2018
	Part Name / Part Descritption: Lighting box	Sheet: A3 - 1 of 1	Scale: 1:2
		Part No / ID No: 14	Revision: A

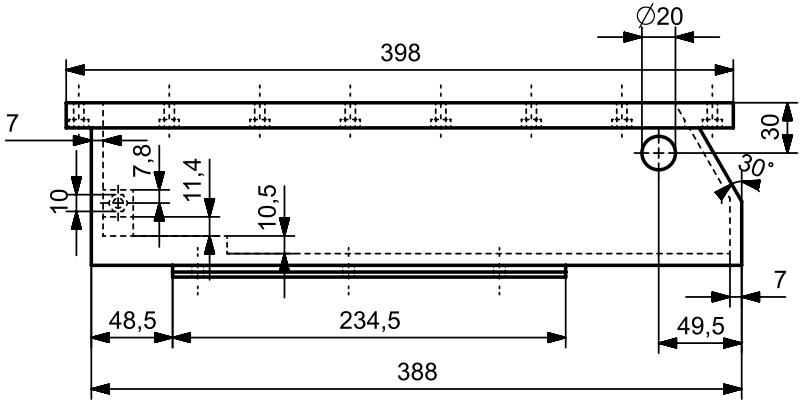
A3

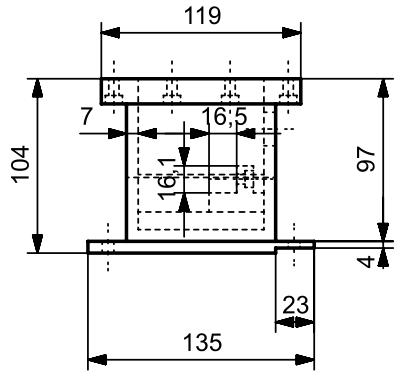


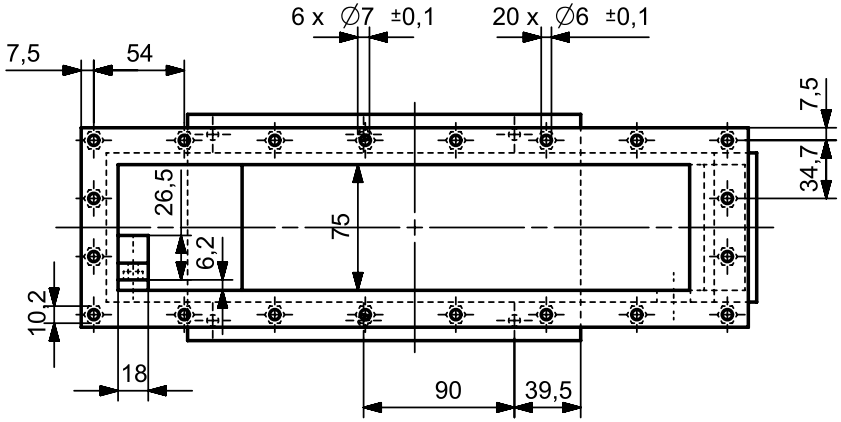


NTNU NTNU Aalesund Larsgardsvegen 2 6009 Aalesund Norway	This drawing is the property of NTNU Aalesund. The content can not be made public, copied or otherwise used without our written consent.	Drawn By: Daniel R. Reite	Date: 29.05.2018
	Part Name / Part Descritption Lid for lighting box	Sheet: A3 - 1 of 1	Scale: 1:1
		Part No / ID No: 15	Revision: A

A3

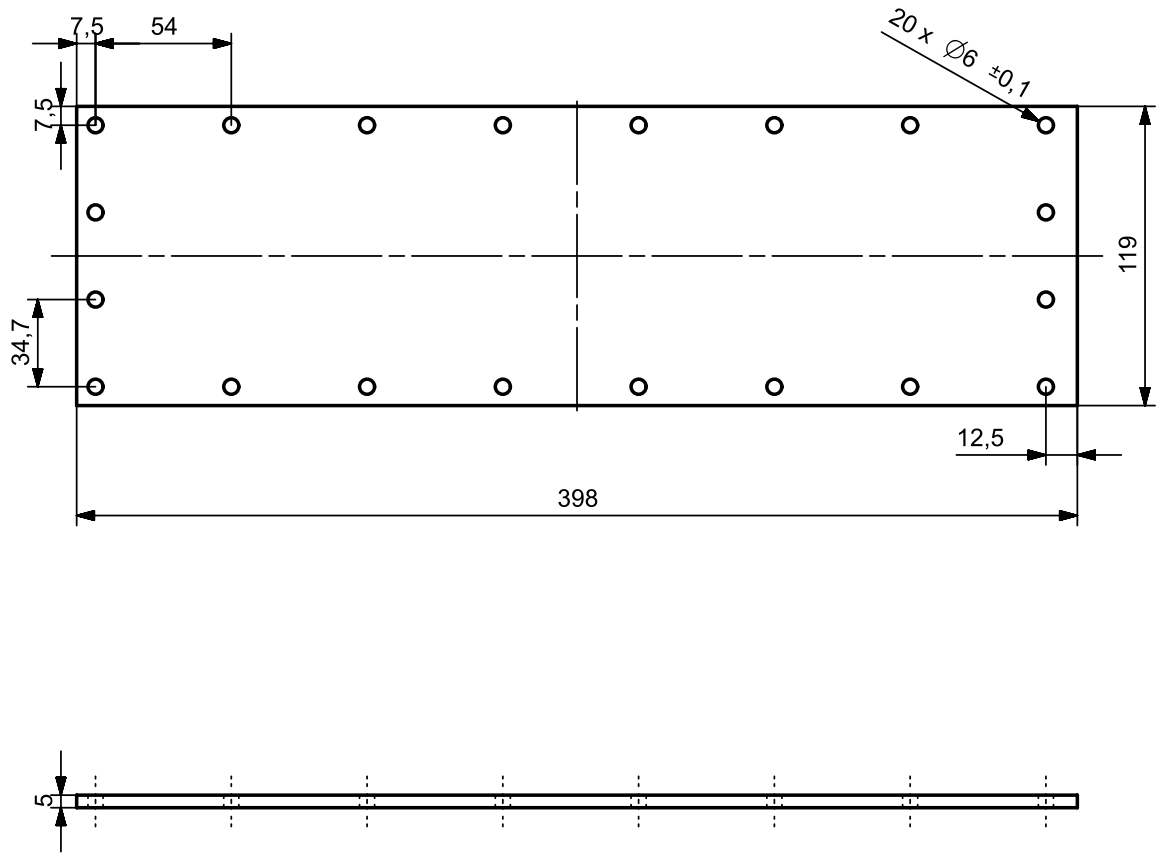






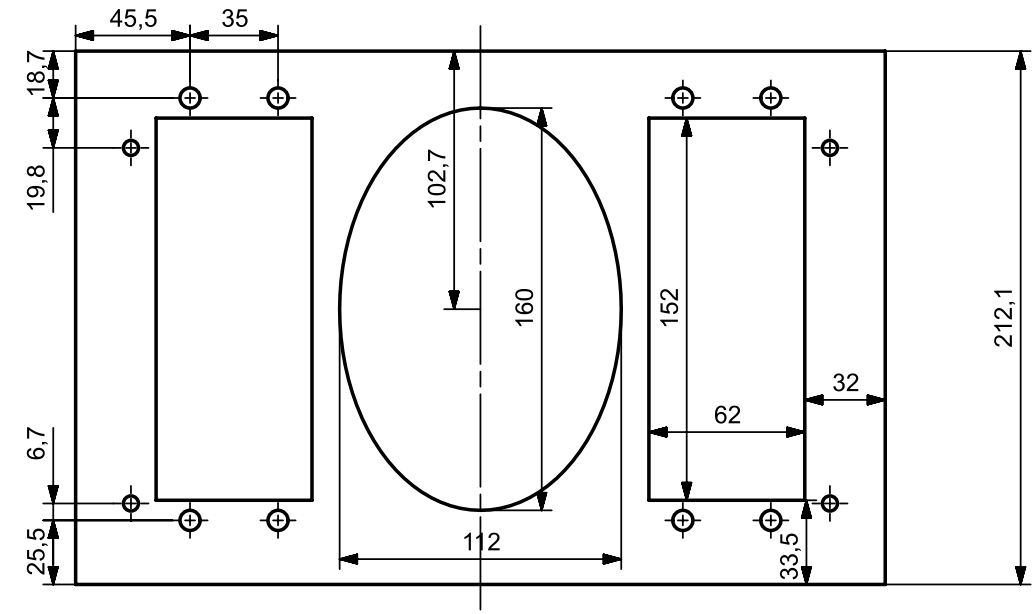
NTNU NTNU Aalesund Larsgardsvegen 2 6009 Aalesund Norway	This drawing is the property of NTNU Aalesund. The content can not be made public, copied or otherwise used without our written consent.	Drawn By: Daniel R. Reite	Date: 29.05.2018
	Part Name / Part Descritption: Actuator box	Sheet: A3 - 1 of 1	Scale: 1:3
		Part No / ID No: 16	Revision: A

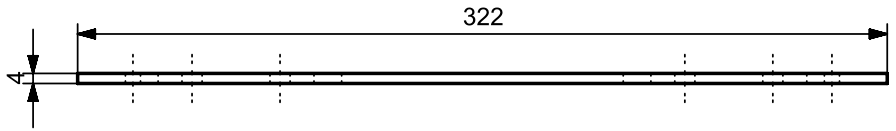
A3



NTNU NTNU Aalesund Larsgardsvegen 2 6009 Aalesund Norway	This drawing is the property of NTNU Aalesund. The content can not be made public, copied or otherwise used without our written consent.	Drawn By: Daniel R. Reite	Date: 29.05.2018
	Part Name / Part Descritption: Lid for actuator box	Sheet: A3 - 1 of 1	Scale: 1:2
		Part No / ID No: 17	Revision: A

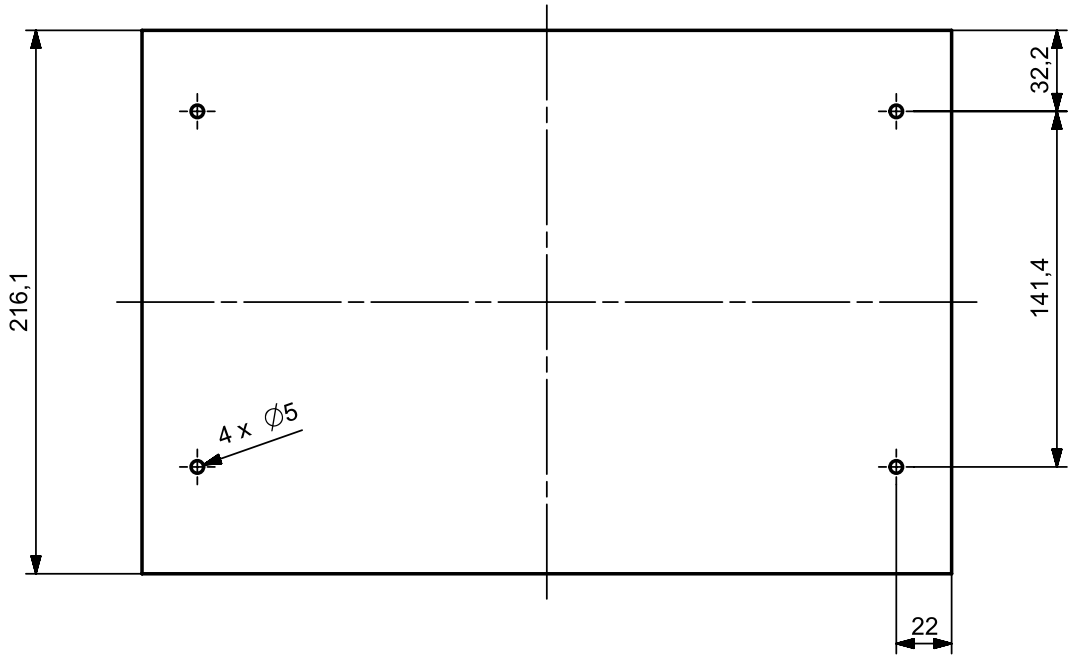
A3

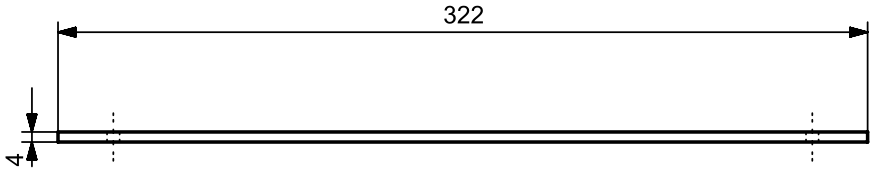


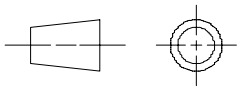


NTNU NTNU Aalesund Larsgardsvegen 2 6009 Aalesund Norway	This drawing is the property of NTNU Aalesund. The content can not be made public, copied or otherwise used without our written consent.	Drawn By: Daniel R. Reite	Date: 29.05.2018
	Part Name / Part Descritption: Lower cover	Sheet: A3 - 1 of 1	Scale: 1:2
		Part No / ID No: 18	Revision: A

A3

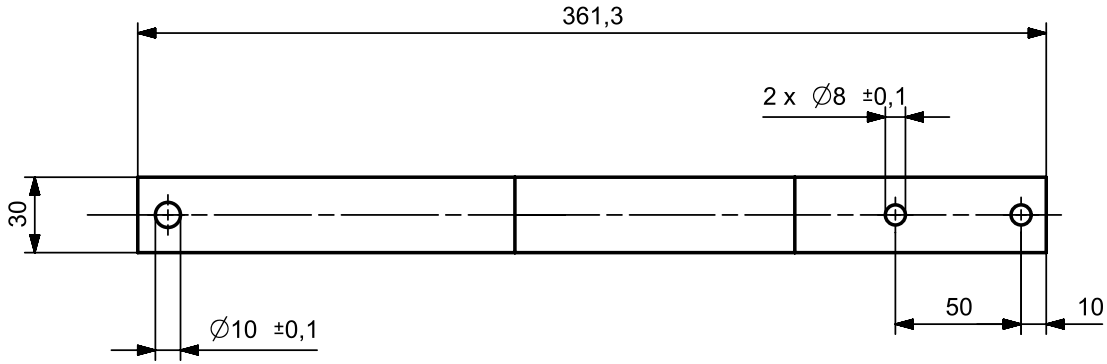


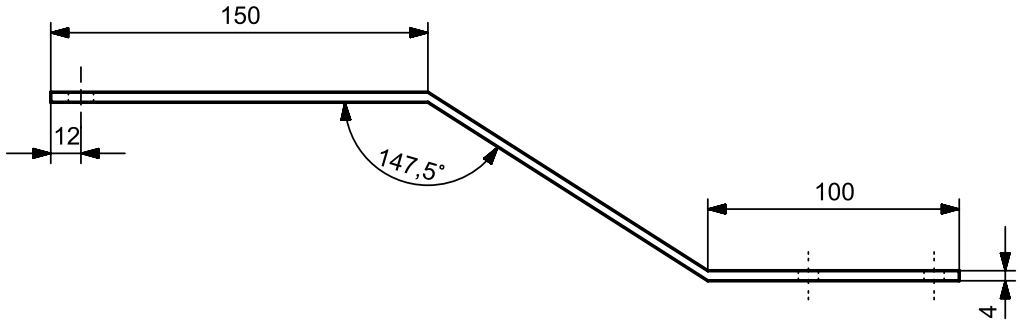


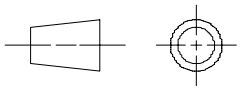


NTNU Aalesund Larsgardsvegen 2 6009 Aalesund Norway	This drawing is the property of NTNU Aalesund. The content can not be made public, copied or otherwise used without our written consent.	Drawn By: Daniel R. Reite	Date: 29.05.2018
	Part Name / Part Descritption: Upper cover	Sheet: A3 - 1 of 1	Scale: 1:2
		Part No / ID No: 19	Revision: A

A3

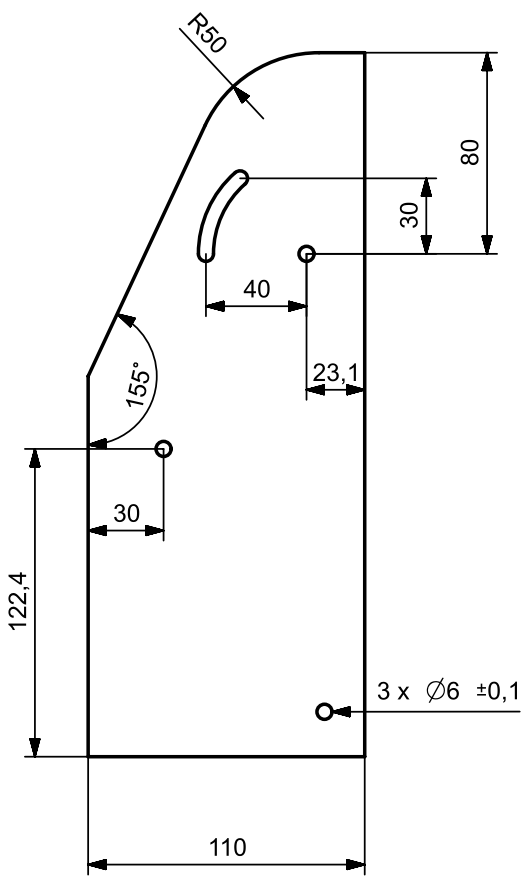


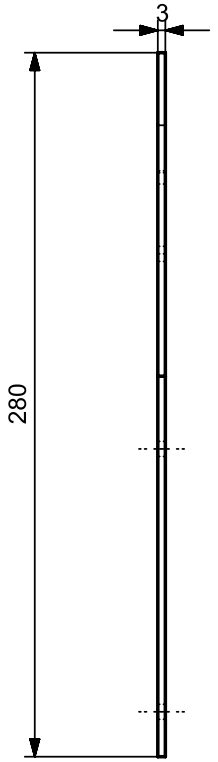




NTNU NTNU Aalesund Larsgardsvegen 2 6009 Aalesund Norway	This drawing is the property of NTNU Aalesund. The content can not be made public, copied or otherwise used without our written consent.	Drawn By: Daniel R. Reite	Date: 29.05.2018
	Part Name / Part Descritption: Cable joint	Sheet: A3 - 1 of 1	Scale: 1:2
		Part No / ID No: 20	Revision: A

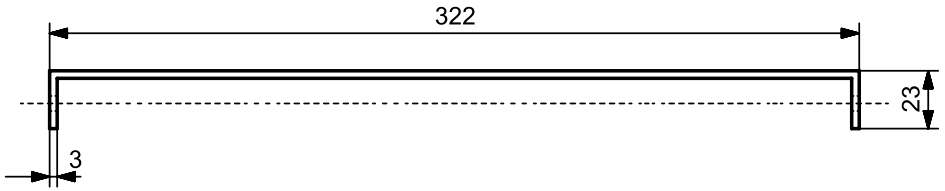
A3

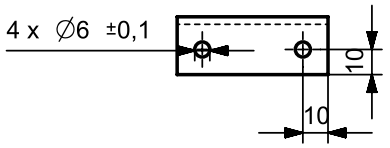


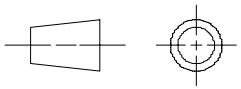


NTNU NTNU Aalesund Larsgardsvegen 2 6009 Aalesund Norway	This drawing is the property of NTNU Aalesund. The content can not be made public, copied or otherwise used without our written consent.	Drawn By: Daniel R. Reite	Date: 29.05.2018
	Part Name / Part Descritption: Spoiler plate	Sheet: A3 - 1 of 1	Scale: 1:2
		Part No / ID No: 21	Revision: A

A3

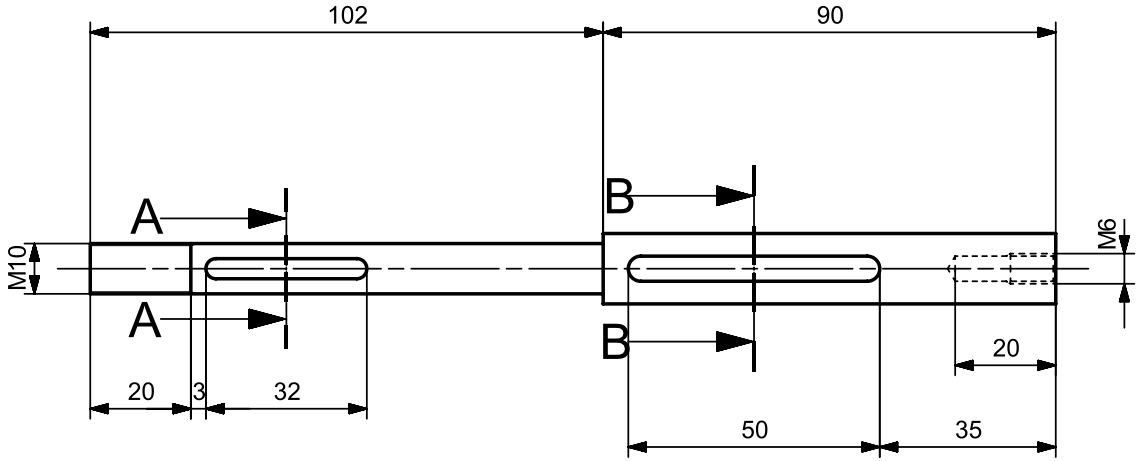


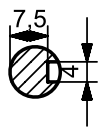




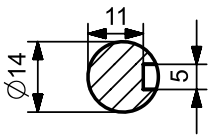
NTNU NTNU Aalesund Larsgardsvegen 2 6009 Aalesund Norway	This drawing is the property of NTNU Aalesund. The content can not be made public, copied or otherwise used without our written consent.	Drawn By: Daniel R. Reite	Date: 29.05.2018
	Part Name / Part Descritption: Spoiler wing	Sheet: A3 - 1 of 1	Scale: 1:2
		Part No / ID No: 22	Revision: A

A3





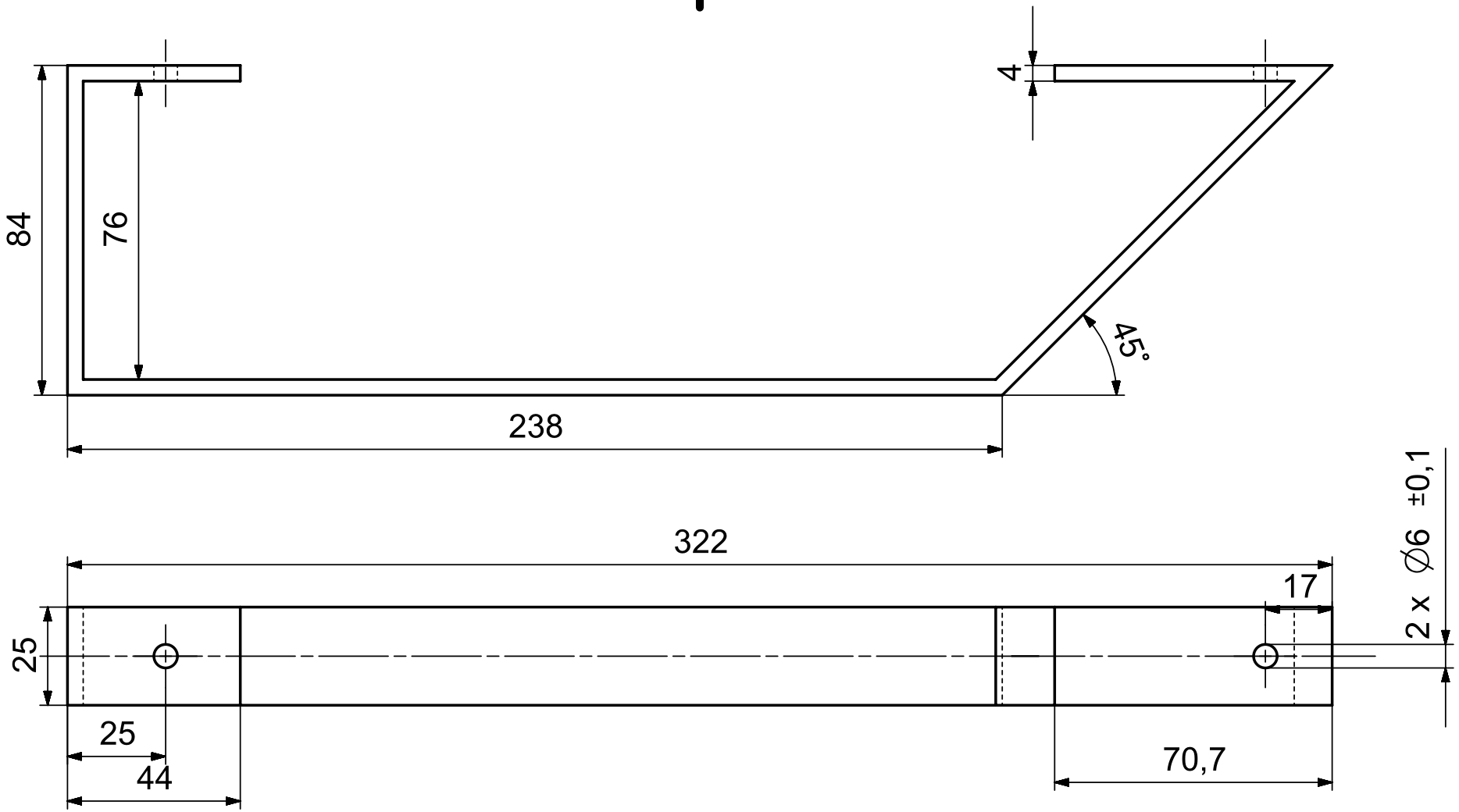
SECTION A-A



SECTION B-B

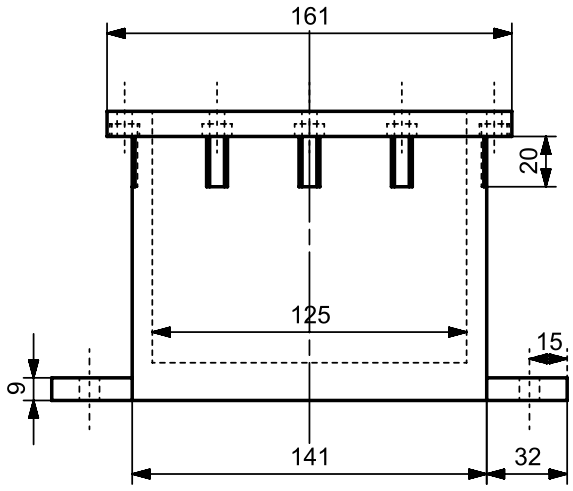
NTNU NTNU Aalesund Larsgardsvegen 2 6009 Aalesund Norway	This drawing is the property of NTNU Aalesund. The content can not be made public, copied or otherwise used without our written consent.	Drawn By: Daniel R. Reite	Date: 29.05.2018
	Part Name / Part Descritption: Foil shaft	Sheet: A3 - 1 of 1	Scale: 1:2
		Part No / ID No: 23	Revision: A

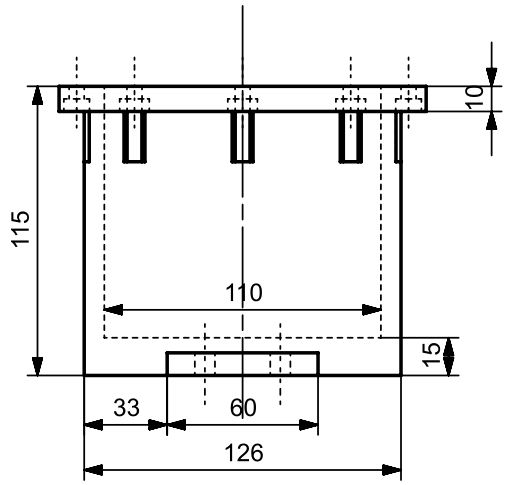
A3

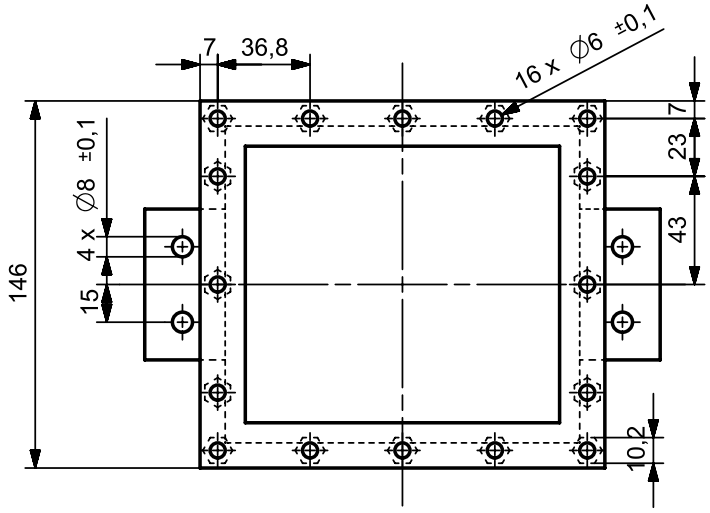


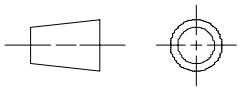
NTNU NTNU Aalesund Larsgardsvegen 2 6009 Aalesund Norway	This drawing is the property of NTNU Aalesund. The content can not be made public, copied or otherwise used without our written consent.	Drawn By: Daniel R. Reite	Date: 29.05.2018
	Part Name / Part Descritption: Feet for ROV	Sheet: A3 - 1 of 1	Scale: 1:2
		Part No / ID No: 24	Revision: A

A3



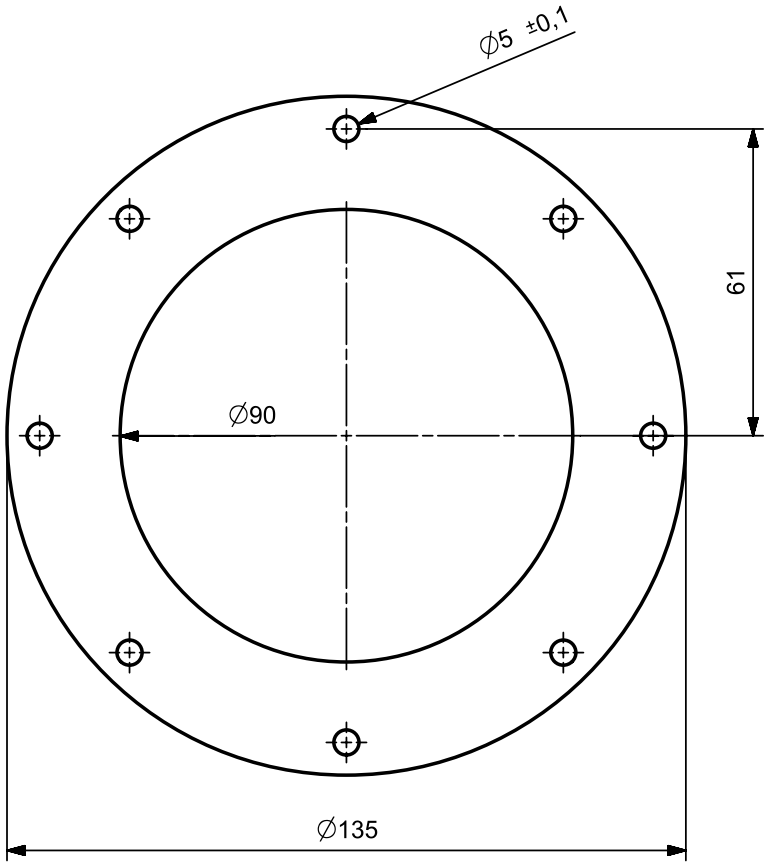


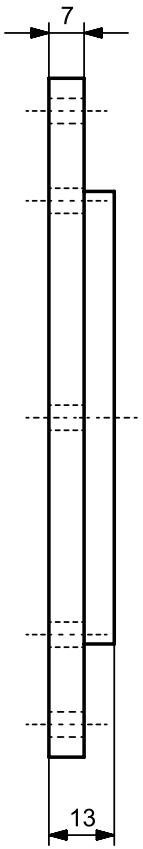


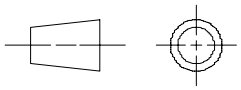


NTNU NTNU Aalesund Larsgardsvegen 2 6009 Aalesund Norway	This drawing is the property of NTNU Aalesund. The content can not be made public, copied or otherwise used without our written consent.	Drawn By: Daniel R. Reite	Date: 29.05.2018
	Part Name / Part Descritption Electronics box	Sheet: A3 - 1 of 1	Scale: 1:2
		Part No / ID No: 25	Revision: A

A3

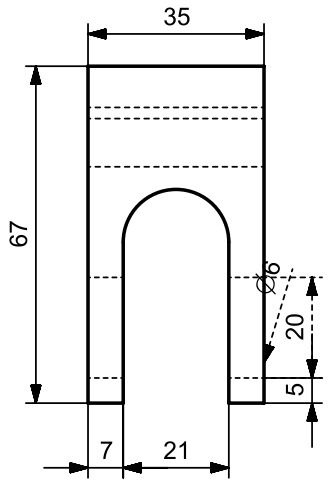


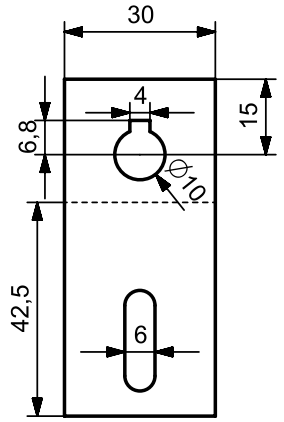


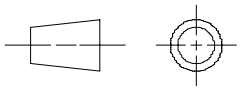


NTNU NTNU Aalesund Larsgardsvegen 2 6009 Aalesund Norway	This drawing is the property of NTNU Aalesund. The content can not be made public, copied or otherwise used without our written consent.	Drawn By: Daniel R. Reite	Date: 29.05.2018
	Part Name / Part Descritption Lid for camera housing	Sheet: A3 - 1 of 1	Scale: 1:1
		Part No / ID No: 26	Revision: A

A3







NTNU NTNU Aalesund Larsgardsvegen 2 6009 Aalesund Norway	This drawing is the property of NTNU Aalesund. The content can not be made public, copied or otherwise used without our written consent.	Drawn By: Daniel R. Reite	Date: 29.05.2018
	Part Name / Part Descritption Joint for wing shaft	Sheet: A3 - 1 of 1	Scale: 1:1
		Part No / ID No: 27	Revision: A

A3

Appendix G

Airfoil Calculations

Total weight	20	kg
Volume (approx. from NX)	17375941	mm^3
	0,01737594	m^3
Buoyancy	174,719431	N
Submerged weight	196,2	
Required lift force for zero buoyancy	21,4805693	N
Require force per wing	10,7	N

Desired upwards force of ROV	30	N
Required force per wing for lift	25,7402846	N

Lift coefficient C_L	Angle
0,07	-4

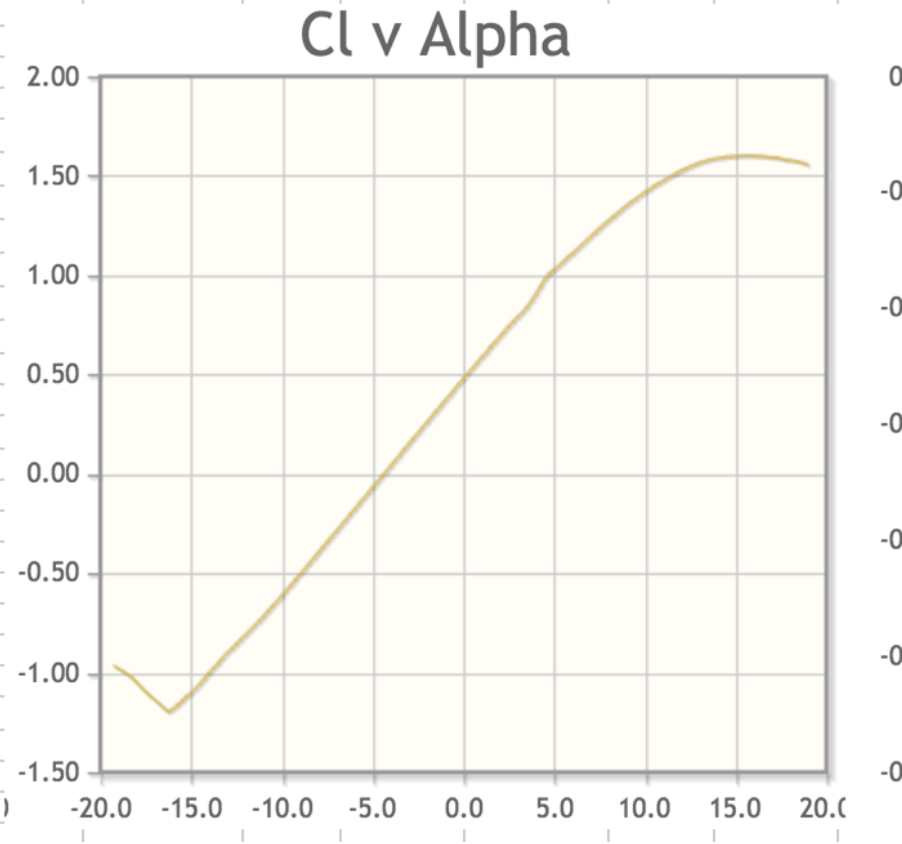
Moment coefficient C_m	-0,12	
Required moment	-6,9	Nm

Distance from actuator to shaft	33	mm
	0,033	m
Required force from actuators	-208,8	N

Surface of one wing	78399	mm^2
	0,078	m^2
Total surface	156798	mm^2

Chord length	150	mm
	0,15	m

Mass density water	
1025	Kg/m^3

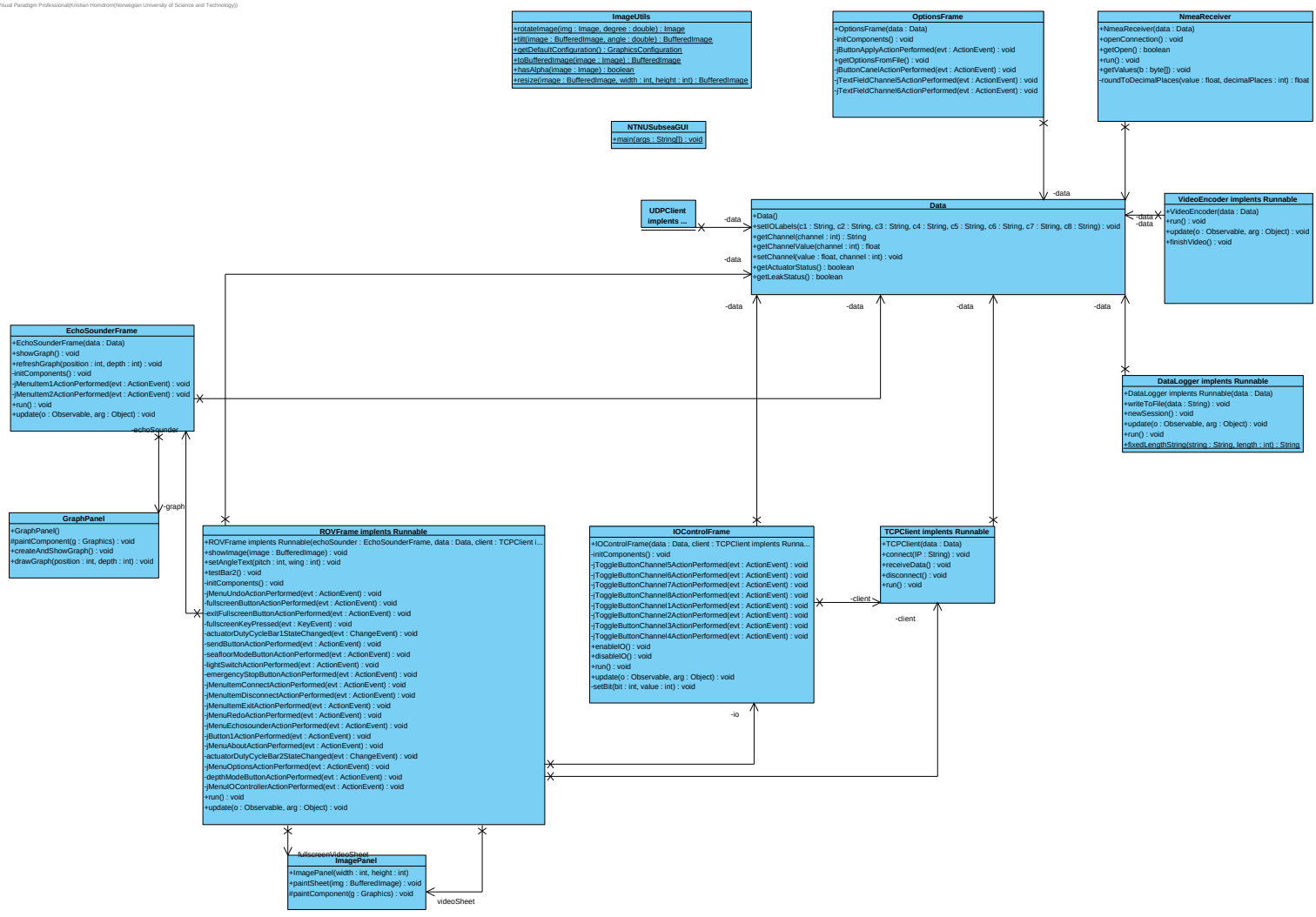


Appendix H

Class Diagram

H.1 Server Application Class Diagram

Visual Paradigm Professional(Kristian Homdrom(Norwegian University of Science and Technology))



H.2 Client Application Class Diagram

Visual Paradigm Professional(Kristian Homdrom(Norwegian University of Science and Technology))

Appendix I

Code

I.1 Arduino Code

```
////////////////////////////////////////////////////////////////////
// I/O ARDUINO FOR CONNECTING EXTRA SENSORS AND EQUPMENT

/*
 * This arduino handles pay-load. Extra equipment and sensors can be connected
 * and controller from the GUI.
 */
#include "Wire.h"
#define arduino 0x07
const int CHANNEL1_PIN = A0;
const int CHANNEL2_PIN = A1;
const int CHANNEL3_PIN = A2;
const int CHANNEL4_PIN = A3;
const int CHANNEL5_PIN = 10;
const int CHANNEL6_PIN = 11;
const int CHANNEL7_PIN = 12;
const int CHANNEL8_PIN = 13;
boolean channel1On = false;
boolean channel2On = false;
boolean channel3On = false;
boolean channel4On = false;
const int SDA_PIN = 2;
const int SCL_PIN = 3;
byte dataOut[16];
union dataOutUnion1 {
  byte b1[4];
  float fval1;
} channel1;
union dataOutUnion2 {
  byte b2[4];
  float fval2;
} channel2;
union dataOutUnion3 {
  byte b3[4];
  float fval3;
} channel3;
union dataOutUnion4 {
  byte b4[4];
  float fval4;
} channel4;

void setup() {
 Wire.begin(arduino);
 Serial.begin(9600);
 Wire.onReceive(receiveData);
 Wire.onRequest(sendData);
```

```
}

/*
 * Reads the analog inputs that are turned on
 */
void loop() {
  if(channel1On){
  channel1.fval1 = analogRead(CHANNEL1_PIN);
   }
   else{
     channel1.fval1 = 0;
   }
  if(channel2On){
  channel2.fval2 = analogRead(CHANNEL2_PIN);
   }
   else{
     channel2.fval2 = 0;
   }
  if(channel3On){
  channel3.fval3 = analogRead(CHANNEL3_PIN);
   }
   else{
     channel3.fval3 = 0;
   }
  if(channel4On){
  channel4.fval4 = analogRead(CHANNEL4_PIN);
   }
   else{
     channel4.fval4 = 0;
   }
}

/*
 * Receives a bit from the I2C, reads the 8 bits and turns on the
 * inputs and outputs that have the values 1.
 */
void receiveData() {
  while (Wire.available()) {
    byte digitalDataIn = Wire.read();
   digitalWrite(CHANNEL5_PIN, bitRead(digitalDataIn, 0));
   digitalWrite(CHANNEL6_PIN, bitRead(digitalDataIn, 1));
   digitalWrite(CHANNEL7_PIN, bitRead(digitalDataIn, 2));
   digitalWrite(CHANNEL8_PIN, bitRead(digitalDataIn, 3));
    channel1On = bitRead(digitalDataIn, 4);
    channel2On = bitRead(digitalDataIn, 5);
```

```
    channel3On = bitRead(digitalDataIn, 6);
    channel4On = bitRead(digitalDataIn, 7);
    Serial.println(digitalDataIn);
  }
}

/*
 * Sends the sensor values read from the inputs
 */
void sendData() {
  int fBytes = 4;
  for (int i = 0; i < fBytes; i++) {
    dataOut[i] = channel1.b1[i];
    dataOut[i + fBytes] = channel2.b2[i];
    dataOut[i + (fBytes * 2)] = channel3.b3[i];
    dataOut[i + (fBytes * 3)] = channel4.b4[i];
  }
 Serial.println(channel1.fval1);
 Serial.println(channel2.fval2);
  Wire.write(dataOut, 16);
}
```

```
////////////////////////////////////////////////////////////////////////////////
// TOWED ROV CONTROL SYSTEM ARDUINO

/*
 * This arduino controls the actuators using a PID regulator.
 * Receives a setpoint, depth, and gains. Calculates an output
 * based on these values.
 */
#include "Wire.h"
#include <PID_v1.h>
#define arduino 0x09
typedef signed char int8_t;
double setpoint, input, posOutput , output, actPosition1, actPosition2;
 float depthFromPressure =0;
double Kp = 0.8;
double Ki = 0.2;
double Kd = 0.1;
byte mode = 0;
byte currentlyRunning = 0;
PID myPID(&input, &output, &setpoint, Kp, Ki, Kd, P_ON_E, DIRECT);
const int RETRACT_PIN1 = 4;
const int EXTEND_PIN1 = 5;
const int RETRACT_PIN2 = 6;
const int EXTEND_PIN2 = 7;
const int POSITION_PIN_1 = A0;
const int POSITION_PIN_2 = A2;
volatile byte actRunning1 = false;
volatile byte actRunning2 = false;
float rovDepth, oceanDepth, userSetpoint;
char data [17];
volatile long timeReset1 = 0;
volatile long timeReset2 = 0;
volatile long timeResetPress = 0;

void setup() {
 Wire.begin(arduino);
 Wire.setClock(400000);
 Serial.begin(115200);
 Serial1.begin(4800);
 Serial2.begin(9600);
 Serial3.begin(9600);
 Serial.println("Initializing I2C devices...");
  input = 0;
  setpoint = 0;
 myPID.SetMode(AUTOMATIC);
 myPID.SetOutputLimits(-512, 511);
```

```
 Wire.onReceive(receiveData);
 Wire.onRequest(sendData);
 pinMode(EXTEND_PIN1, OUTPUT);
 pinMode(EXTEND_PIN2, OUTPUT);
 pinMode(RETRACT_PIN1, OUTPUT);
 pinMode(RETRACT_PIN2, OUTPUT);
}

/*
 * Reads the position of the actuators potensiometers, runs the PID, retrieves depth
and
 * moves the actuators to the desired position.
 */
void loop() {

 actPosition1 = (analogRead(POSITION_PIN_1));
 actPosition2 = analogRead(POSITION_PIN_2);
  runPid();
  // Timer to make the H-bridge alternate directions smoothly
  if (timerHasPassed(100, 1)) {
    runActuators();
    resetTimer();
    Serial.print("Setpoint: ");
    Serial.println(setpoint);
    Serial.print("Depth: ");
    Serial.println(input);
    Serial.print("Output: ");
    Serial.println(output);
  }
 if(timerHasPassedPress(100)){
receivePressure();
  }
}

/*
 *Receive depth over serial communication link
 */
void receivePressure() {
  typedef union {
    float f;
    byte b[4];
  } fDepth;
   fDepth packedFloat;
 // send data only when you receive data:
 byte s=Serial1.available();
```

```
       // 4 bytes available read.
       if(s==4){
      Serial1.readBytes(packedFloat.b,4);
      //Check if value is not real
      if((packedFloat.b!=0) )
      {
      depthFromPressure= packedFloat.f;

      }
        // flush serial link for data
        Serial1.flush();
          }
          else{
        Serial1.flush();
          }

          //reset timer for new session
          resetTimerPress();}

/**
 * Receive data event over I2C and sets the values received
 */
void receiveData(int n) {
  int i = 0;
   while (Wire.available() > 0) {
    data[i] = Wire.read();
    i++;
  }
  setValues();

}

/*
 * Runs the actuators to the calculated position
 */
void runActuators() {
  if ((posOutput) > actPosition2 + 50) {
    Serial.println("Inn 2");
    moveActuator1(0);
    currentlyRunning = 1;
  }
  // Output mindre enn posisjon - dødsone
  else if ((posOutput) < actPosition2 - 50) {
    Serial.println("Ut 2");
    moveActuator1(4095);
    currentlyRunning = 1;
```

```
  }
  else {
    Serial.println("Idle 2");
    moveActuator1(2048);
     currentlyRunning = 0;
  }
  // Output større enn posisjon + dødsone
  if ((posOutput) > actPosition1 + 50) {
    Serial.println("Inn");
    moveActuator2(0);
     currentlyRunning = 1;
  }
  // Output mindre enn posisjon - dødsone
  else if ((posOutput) < actPosition1 - 50) {
    moveActuator2(4095);
     currentlyRunning = 1;
  }
  else {
    Serial.println("Idle");
    moveActuator2(2048);
     currentlyRunning = 0;
  }
}

/*
 * Sends the depth sensor value and actuator status to the server
 */
void sendData() {
  typedef union {
     float f;
     byte b[4];
  } fDepth2;
   fDepth2 packedFloat;

 packedFloat.f= depthFromPressure;
 Serial.println("dybde");
 Serial.println(depthFromPressure);
   byte data[5];
   for(int i = 0 ; i<4; i++)
   {
      data[i] = packedFloat.b[i];

   }
  data[4]=currentlyRunning;
 Wire.write(data,5);
}
```

```
/*
 * Parses and sets the values received over I2C
 */
void setValues() {
  int fBytes = 5;
  String rovDepthString;
  String oceanDepthString;
  String setpointString;
  String kpString;
  String kdString;
  String kiString;
  String modeString;
  for (int i = 0; i < fBytes; i++) {
    if (i < 2) {
      kpString += data[i];
      kdString += data[i + 2];
      kiString += data[i + 4];
    }
//    rovDepthString += data[i + 6];
    oceanDepthString += data[i + 6];
    setpointString += data[i + 11];
  }
  modeString = data[16];
  mode = modeString.toFloat();
  Kp = kpString.toFloat() / 1.0;
  Kd = kdString.toFloat() / 10.0;
  Ki = kiString.toFloat() / 10.0;
  Serial.println(Kp);
  Serial.println(Kd);
  Serial.println(Ki);
 // rovDepth = rovDepthString.toFloat() / 50.0;
  oceanDepth = oceanDepthString.toFloat() / 50.0;
  input = depthFromPressure * 20.0;
  if (mode == 0) {
    setpoint = (setpointString.toFloat() / 50.0);
    Serial.println(setpoint);
  }
  else if (mode == 1) {
    setpoint = (oceanDepth - setpointString.toFloat() / 50.0);
    Serial.println(setpoint);
  }
}

void runPid() {
  myPID.SetTunings(Kp, Ki, Kd);
```

```
  myPID.Compute();
  posOutput = output + 512;
}

float bytesToFloat(byte packet[], int i) {
  float out;
  memcpy(&out, &packet[i], sizeof(float));
  return out;
}

boolean timerHasPassed (long limit, int timerIndex) {
  long timer = 0;
  if (timerIndex == 1) {
    timer = timeReset1;
  }
  if (timerIndex == 2) {
    timer = timeReset2;
  }
  boolean result = false;
  if ((millis() - timer) > limit) {
    result = true;
    return result;
  }
  return result;
}

// Cheks if the timer have passed

boolean timerHasPassedPress (long limit) {
  boolean result = false;
  if ((millis() - timeResetPress) > limit) {
    result = true;
    return result;
  }
  return result;
}

//Reset the timer so it can be used again
void resetTimerPress() {
  timeResetPress = millis();
  // Serial.println("Reset");
}

//Reset the timer so it can be used again
void resetTimer() {
  timeReset1 = millis();
```

```
  // Serial.println("Reset");
}

//Reset the timer so it can be used again
void resetTimer2() {
  timeReset2 = millis();
}

//sets the new target for the JRK21V3 controller, this uses pololu high resulution
protocal
void moveActuator1(int x) {
 Serial.println("move1");
  word target = x;  //only pass this ints, i tried doing math in this and the
remainder error screwed something up
  Serial2.write(0xAA); //tells the controller we're starting to send it commands
  Serial2.write(0xB);   //This is the pololu device # you're connected too that is
found in the config utility(converted to hex). I'm using #11 in this example
  Serial2.write(0x40 + (target & 0x1F)); //first half of the target, see the pololu
jrk manual for more specific
  Serial2.write((target >> 5) & 0x7F);   //second half of the target, " " "
}

//sets the new target for the JRK21V3 controller, this uses pololu high resulution
protocal
void moveActuator2(int x) {
 Serial.println("move2");
  word target = x;  //only pass this ints, i tried doing math in this and the
remainder error screwed something up
  Serial3.write(0xAA); //tells the controller we're starting to send it commands
  Serial3.write(0xB);   //This is the pololu device # you're connected too that is
found in the config utility(converted to hex). I'm using #11 in this example
  Serial3.write(0x40 + (target & 0x1F)); //first half of the target, see the pololu
jrk manual for more specific
  Serial3.write((target >> 5) & 0x7F);   //second half of the target, " " "
}
```

```
////////////////////////////////////////////////////
////////////////////
// DEPTH SENSOR ARDUINO

#include <Wire.h>
#include "MS5837.h"

MS5837 sensor;
uint32_t timer = millis();
long timeReset1;
//-------------------------------------------------
------------------
/**
   Setup, initalises Arduino
*/
void setup() {
  Serial.begin(4800);
  Wire.begin();
  // Initialize pressure sensor
  // Returns true if initialization was successful
  while (!sensor.init()) {
    Serial.println("Init failed!");
    Serial.println("Are SDA/SCL connected
correctly?");
    Serial.println("Blue Robotics Bar30:
White=SDA, Green=SCL")  ;
    Serial.println("\n\n\n");
    delay(5000);
  }
  sensor.setModel(MS5837::MS5837_30BA);
```

```
  //kg/m^3(997freshwater,1029 for seawater)
  sensor.setFluidDensity(1029);
}
//-----------------------------------------------
-----------------
/**
   Send data with coninous interval
*/
void loop() {
  // Sending data over serial link
  if (timerHasPassed(500)) {
    sendData();
  }
}
//-----------------------------------------------
-----------------
/**
   Send data over serial link
*/
void sendData() {
  // Update pressure and temperature readings
  sensor.read();
  float depth = sensor.depth();
  // Union for same reference in memory
  typedef union {
    float f;
    byte b[4];
  } fDepth;
  //Union variable
  fDepth packedFloat;
```

```
  packedFloat.f = depth;
  byte data[4];
  // Set byte array of union equal to data array
  for (int i = 0 ; i < 4; i++)
  {
    data[i] = packedFloat.b[i];
  }

  // Check if within  valid range
  if ((depth > 0.0) && (depth < 60.0)) {
    //Send 4 bytes of data
    Serial.write(data, 4);
  }
  //reset timer for new session
  resetTimer();
}
//-------------------------------------------------
-----------------
// Cheks if the timer have passed
boolean timerHasPassed (long limit) {
  boolean result = false;
  if ((millis() - timeReset1) > limit) {
    result = true;
    return result;
  }
  return result;
}
//-------------------------------------------------
-----------------
```

```
//Reset the timer so it can be used again
void resetTimer() {
  timeReset1 = millis();
}
```

```
////////////////////////////////////////////////////////////////////////////////
// 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
#define arduinoAddress 0x04   // I2C Address


// Connect Vin (5V) and Gnd
// GPS TX --> Digital 11
// GPS RX --> Digital 10
SoftwareSerial nmeaSerial(5,6); // 5 RX pin, 6 TX pin, if
                                // is unreadable try switching.
NMEA nmeaDecoder(ALL);

  // Set true if debugging
  boolean debug = true;

  // Variables for NMEA data
  float depthBelowTd = 0;
  float depth = 0;
  float offsetFromTd = 0;
  float speedKnots = 0;
  float speedKm = 0;
  float temp;

  // 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;

 void setup() {

  // Serial setup
  Serial.begin(4800);
```

```
 nmeaSerial.begin(4800);
 Serial.println("Starting");

//-------------------------------------------------------------------------------
  // 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);

 }

 // Main loop, collecting data, parsing and sending to RBPi
 void loop() {
  //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.
     sendNMEAData();

  }
 }
}

//-------------------------------------------------------------------------------
// 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);
        }
      }

    //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);
        }
```

```
      }

   //if t0 is Water speed and heading
  if(strcmp(t0,"VWVHW")==0){
        speedKnots = atof(t5);   //Extracts field #5 which is speed in knots.
        speedKm = atof(t7);      //Extracts field #7 which is speed in km/h.

        if(debug){
         Serial.print("Speed in Knots: ");
         Serial.println(speedKnots);
         Serial.print("Speed in Km/h: ");
         Serial.println(speedKm);
         }
       }

   //if t0 is Mean Temperature of Water
  if(strcmp(t0,"YXMTW")==0){
       temp = atof(t1);        //Extracts field #1 which is mean temperature.

       if(debug){
         Serial.print("Temperature in celcius: ");
         Serial.println(temp);
        }
       }
}

//--------------------------------------------------------------------------------
// Sends data to the GUI over serial line.

void sendSeraialNMEAData() {

  //Create byte array of sensor values
 byte data[6] = {(depthBelowTd),(depth),(offsetFromTd),
                  (speedKnots),(speedKm),(temp)};

  //Write array "data" with 6 byte.
 Serial.write(data,sizeof(data));

  if(debug){
    Serial.println("Sending information");
  }
}



//--------------------------------------------------------------------------------
// Sends data to RBPi via Wire.Write();
```

```
void sendNMEAData() {

  //Create byte array of sensor values
  byte data[6] = {(depthBelowTd),(depth),(offsetFromTd),
                  (speedKnots),(speedKm),(temp)};

  //Write array "data" with 6 byte.
 Wire.write(data,sizeof(data));
}

//-------------------------------------------------------------------------------
//Recives data from computer

void receiveData(int b) {

  //recieves connection from Pi
  Serial.println("Event Recived");

}
```

```
////////////////////////////////////////////////////////////////////////////////
///
// NMEA PARSER FOR SURFACE VESSEL

// NMEA ES and GPS Parser for Adafruit GPS Breakout board with
// external antenna and NMEA 0183 echo sounder
// Connect Vin (5V) and Gnd
// GPS TX --> Digital 16
// GPS RX --> Digital 17
// ES  TX --> Digital 18
// ES  RX --> Digital 19
// MorSol94

#include <Adafruit_GPS.h>     //GPS Library
#include <nmea.h>             //NEMA Library

//Extra hardware serial ports for GPS and ES
#define ESSerial Serial1
#define GPSSerial Serial2

// Connect to the GPS on the hardware port
Adafruit_GPS GPS(&GPSSerial);

NMEA nmeaDecoder(ALL);

// Set GPSECHO to 'false' to turn off echoing the GPS data to the Serial console
// Set to 'true' if you want to debug and listen to the raw GPS sentences
#define GPSECHO false  //if true, NMEA sentences from GPS are printed in text

boolean debug = false;

uint32_t timer = millis();

// Variables for GPS
float latitude = 0.00;
float longitude = 0.00;
float angle = 0.00;

// Variables for ES data
float depthBelowTd = 0.00;
float depth = 0.00;
float offsetFromTd = 0.00;
float speedKnots = 0.00;
float speedKm = 0.00;
float temp = 0.00;
```

```
// Timer variavle
long timeReset1;

// 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;

void setup()
{
  // Starting serial port at baud rate 115200 to not drop cahrs.
 Serial.begin(115200);

  // Starting echo sounder serial port, default baud rate for ES is 4800.
 ESSerial.begin(4800);

  // Starting GPS serial port, default baud rate for GPS is 9600.
 GPS.begin(9600);

  // uncomment this line to turn on RMC (recommended minimum) and GGA (fix data)
including altitude
 GPS.sendCommand(PMTK_SET_NMEA_OUTPUT_RMCGGA);

  // Set the update rate, using the recomandet rate of 1 Hz
 GPS.sendCommand(PMTK_SET_NMEA_UPDATE_1HZ); // 1 Hz update rate

  // Request updates on antenna status, comment out to keep quiet
 GPS.sendCommand(PGCMD_ANTENNA);

  delay(1000);
}

// Main loop, collecting, parsing and sending data.
void loop()
{
  // read data from the GPS.
  char c = GPS.read();

  // if you want to debug.
```

```
  if (GPSECHO)
    if (c) Serial.print(c);  // Printing GPS NMEA sentence

  // if a sentence is received it will be checked and handled.
  if (GPS.newNMEAreceived()) {

    if (!GPS.parse(GPS.lastNMEA())) // this also sets the newNMEAreceived() flag to
false
      return; // IF sentence fail to parse, return and wait for a new.
  }

  // Reset timer when it wraps around.
  if (timer > millis()) timer = millis();

  // approximately every second or so, print out the current stats
  if (millis() - timer > 1000) {
    timer = millis(); // reset the timer
    if (GPS.fix) {

      // Setting variables to correct value
      latitude = (GPS.latitudeDegrees);
      longitude = (GPS.longitudeDegrees);
      angle = (GPS.angle);
    }
  }

  // Updating and handeling NMEA sentences from echo sounder if avilable.
  // in debug mode the sentences will be printed out.
  if (Serial1.available()) {    // Checks for incoming NMEA sentences.
    if (nmeaDecoder.decode(Serial1.read())) {   //checks for valid NMEA sentence
from ES

      // Decoding nmea sentence
      decodeNmeaSentence();

      // Parse nmea sentence
      parseNmeaSentence();
    }
  }

  // Sending data to the GUI.
 if(timerHasPassed(300)){
 sendSeraialNMEAData();
  }
}
```

```
//--------------------------------------------------------------------------------
// NMEA decoder, Splitts the NMEA sentence where it is seperated by commas.
// Check http://www.catb.org/gpsd/NMEA.html for build up of NMEA sentences.
// t0 is always sentence identifyer, 5 letters telling wich devie is talking,
// the rest is data in the NMEA sentence, for each comma add a number.

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
in meters.

    if (debug) {
      Serial.print("Depth below Transducer (m): ");
      Serial.println(depthBelowTd);
    }
  }

  //if t0 is Depth of Water
  if (strcmp(t0, "SDDPT") == 0) {
    depth = atof(t1);           //Extracts field #1 which is depth of water in
meters.
    offsetFromTd = atof(t3);    //Extracts field #3 which is transducer offset.

    if (debug) {
      Serial.print("Depth (m): ");
      Serial.println(depth);
      Serial.print("Transducer offset: ");
```

```
    Serial.println(offsetFromTd);
   }
  }

  //if t0 is Water speed and heading
  if (strcmp(t0, "VWVHW") == 0) {
   speedKnots = atof(t5);   //Extracts field #5 which is speed in knots.
   speedKm = atof(t7);      //Extracts field #7 which is speed in km/h.

   if (debug) {
    Serial.print("Speed in Knots: ");
    Serial.println(speedKnots);
    Serial.print("Speed in Km/h: ");
    Serial.println(speedKm);
   }
  }

  //if t0 is Mean Temperature of Water
  if (strcmp(t0, "YXMTW") == 0) {
   temp = atof(t1);       //Extracts field #1 which is mean temperature.

   if (debug) {
    Serial.print("Temperature in celcius: ");
    Serial.println(temp);
   }
  }
}

//--------------------------------------------------------------------------------
// Sends data to the GUI over serial line.

void sendSeraialNMEAData() {
  typedef union {
   float f;
   byte bytes[4];
  } fLat,fLong,fAngle,fDepth,fSpeed;
  byte data[24];
  //byte data[20];
  fLat packedFloat;
  packedFloat.f = latitude;
  for (int i = 0; i<4; i++){
   data[i] = 0;                      // Satrt bytes
   data[i+4] = packedFloat.bytes[i];   // Bytes containing latitude coordinates
  }
   fLong packedFloat2;
   packedFloat2.f = longitude;
```

```
    for (int i = 0; i<4; i++){
    data[i+8] = packedFloat2.bytes[i];   // Bytes containing longitude coordinates
  }
    fAngle packedFloat3;
    packedFloat3.f = angle;
    for (int i = 0; i<4; i++){
    data[i+12] = packedFloat3.bytes[i];  // Bytes containing heading
  }
    fDepth packedFloat4;
    packedFloat4.f = (depthBelowTd);
    for (int i = 0; i<4; i++){
    data[i+16] = packedFloat4.bytes[i];  // Bytes containing ES depth below
transducer
  }
    fSpeed packedFloat5;
    packedFloat5.f = (speedKnots);
    for (int i = 0; i<4; i++){
      data[i+20] = packedFloat5.bytes[i];  // Bytes containing ES speed in knots
    }
    //   for (int i = 0; i<24; i++){
    //    Serial.println(data[i]);
    //   }
    char test = '<';
 //Serial.write(test);
 Serial.write(data,24);
 resetTimer();

  if (debug) {
    Serial.println("Sending information");
  }
}

//-------------------------------------------------------------------------
// Cheks if the timer have passed

boolean timerHasPassed (long limit) {
  boolean result = false;
  if ((millis() - timeReset1) > limit) {
    result = true;
    return result;
  }
  return result;
}

//-------------------------------------------------------------------------
//Reset the timer so it can be used again
```

```
void resetTimer() {
  timeReset1 = millis();
  // Serial.println("Reset");
}
```

I.2 Python Code

```
# Videostream server from the Towed-ROV in Python 2.7.
# This code captures and sends video stream via a UDP socket.
# author MorSol.

# import the necessary packages.
from picamera.array import PiRGBArray
from picamera import PiCamera
import socket
import time
import cv2

# initialize the camera and grab a reference to the raw camera capture.
camera = PiCamera()
camera.resolution = (1280, 720)
camera.framerate = 90
camera.vflip = True
UDP_IP = "192.168.0.20"
UDP_PORT = 5002
sock = socket.socket(socket.AF_INET,   # Internet socket.
            socket.SOCK_DGRAM)  #UDP.
rawCapture = PiRGBArray(camera, size=(1280, 720))



# allow the camera to warmup.
time.sleep(0.1)




# capture frames from the camera
for frame in camera.capture_continuous(rawCapture,
format="bgr", use_video_port=True):
# 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", img, 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)

# if the `q` key was pressed, break from the loop.
if input == ord("q"):
break
```

I.3 Java Server Application Code

```
/*
 * 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
 * sesnors. There will also be a video stream from the ROV.
 *
 * The system consists of a Raspberry Pi on the ROV that is connected to
several
 * Arduino microcontrollers that are connected to sensors and and programed
with
 * the control system.
 * The external computer which is on the surface vessel is connected to a GPS,
 * echo sounder and the RBPi. It will present and save data in addition to
 * handle user commands.
 */


import ExternalCommunication.StorageBox;
import java.util.Observable;
import java.util.Observer;

/**
 * This class has responsiblity of converting read values from the Pressure
 * sensor obtained over the i2c bus to depth, pressure(mBar) and
temper(Celsius)
 *
 * @author Kristian Homdrom
 */
public class PressureValues implements Observer {

    private volatile int C1, C2, C3, C4, C5, C6;
    private static volatile double depth;
    private static volatile double mBar;
    private static volatile double tempC;
    private final I2CBusComm i2c;
    private StorageBox sb;

    /**
     * Constructor for Pressure Values class
     *
     * @param i2c i2c link to the I2CBusComm class
```

```
     * @param sb StorageBox linkt to the Storagebox class
     */
    public PressureValues(I2CBusComm i2c, StorageBox sb) {
        this.i2c = i2c;
        this.sb = sb;
    }

    /**
     * Set calibration stat from sensor to be used in temp and pressure
     * measuring
     *
     * @param stats paramters used for calibration, obtained from pressure
     * sensor
     */
    public void setCalibrationPressureStat(int[] stats) {
        C1 = stats[0];
        C2 = stats[1];
        C3 = stats[2];
        C4 = stats[3];
        C5 = stats[4];
        C6 = stats[5];
    }

    /**
     * send the depth in meters
     *
     */
    public void sendDepth() {
        sb.setPressureDepth((float) depth);
    }

    /**
     * send the pressure in millibar of the surroundings of the ROV
     *
     */
    public void sendPressure() {
        sb.setPressure((float) mBar);
    }

    /**
     * send the temperature of the water outside ROV
```

```
     *
     */
    public void sendTempC() {
        sb.setTemperature((float) tempC);
    }

    /**
     * convert array into raw readings of the depth sensor
     *
     * @param data byte array of readings from depth sensor
     */
    public void pressureRawData(byte[] data) {
        long D1 = ((data[0] & 0xFF) * 65536 + (data[1] & 0xFF) * 256 +
(data[2] & 0xFF));
        long D2 = ((data[3] & 0xFF) * (long) 65536 + (data[4] & 0xFF) * (long)
256 + (data[5] & 0xFF));
        // Convert the raw data to pressure, depth, tempC
        conversion(D1, D2);
    }

    /**
     * Convert valuies from pressure sensor to readable values
     *
     * @param D1 Raw value 1
     * @param D2 Raw value 2
     */
    public void conversion(long D1, long D2) {

        long dT = D2 - C5 * 256;
        long TEMP = 2000 + dT * C6 / (long) 8388608;
        long OFF = C2 * 65536L + (C4 * dT) / 128L;
        long SENS = C1 * 32768L + (C3 * dT) / 256L;
        long T2 = 0;
        long OFF2 = 0;
        long SENS2 = 0;

        if (TEMP >= 2000) {
            T2 = 2 * (dT * dT) / 137438953472L;
            OFF2 = ((TEMP - 2000) * (TEMP - 2000)) / 16;
            SENS2 = 0;
        } else {
```

```
            if (TEMP < 2000) {
                T2 = 3 * (dT * dT) / 8589934592L;
                OFF2 = 3 * ((TEMP - 2000) * (TEMP - 2000)) / 2;
                SENS2 = 5 * ((TEMP - 2000) * (TEMP - 2000)) / 8;
                if (TEMP < -1500) {
                    OFF2 = OFF2 + 7 * ((TEMP + 1500) * (TEMP + 1500));
                    SENS2 = SENS2 + 4 * ((TEMP + 1500) * (TEMP + 1500));
                }
            }
        }

        TEMP = TEMP - T2;
        OFF = OFF - OFF2;
        SENS = SENS - SENS2;
        // Generate milliBar from readings
        mBar = ((((D1 * SENS) / 2097152l - OFF)) / (long) 8192) / (long) 10.0;

        // Convert milliBar to deciBar
        double deciBar = mBar / 100.0;
        // Apply gravity, this could vary, but for our prototype this is not a
part worth having 100% correct
        double gravity = 9.780318;
        //Correect readings from Unesco formula
        depth = (((((-1.82 * Math.pow(10, -15) * deciBar + 2.279 *
Math.pow(10, -10)) * deciBar - 2.2512 * Math.pow(10, -5)) * deciBar + 9.72659)
* deciBar) / gravity);
        tempC = TEMP / 100.0;

    }

    @Override
    public void update(Observable o, Object o1) {
        sendDepth();
        sendPressure();
        sendTempC();
        if ((C1 == 0) || (C4 == 0)) {
            setCalibrationPressureStat(i2c.getPressureCalibrationData());
        }
        pressureRawData(i2c.getPressVal());

    }
```

```
}
```

```
/*
 * 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
 * sesnors. There will also be a video stream from the ROV.
 *
 * The system consists of a Raspberry Pi on the ROV that is connected to
several
 * Arduino microcontrollers that are connected to sensors and and programed
with
 * the control system.
 * The external computer which is on the surface vessel is connected to a GPS,
 * echo sounder and the RBPi. It will present and save data in addition to
 * handle user commands.
 */


import java.nio.ByteBuffer;
import java.util.Observable;

/**
 * Class to store values for the ROV application Bytearray/CharArrays created
to
 * be used to send information to GUI,Regulator, IO
 *
 * @author Kristian Homdrom,Marius Nonsvik
 */
public class StorageBox extends Observable {

    private byte ioByte;
    private volatile static ByteBuffer guiArray;
    private byte lights;
    private char[] c_arr;
    private static volatile String kpString, kiString, kdString,
pressDepthString, echoDepthString, setPointString, modeString, IP;
    private static char mod;
    private static float speed;
    private static boolean available = false;
    private static boolean run = false;

    //Set Gain values
```

```
    private static double kpLow, kpMed, kpHigh;
    private static double kiLow, kiMed, kiHigh;
    private static double kdLow, kdMed, kdHigh;

    /**
     * Constructor of the Storagebox
     */
    public StorageBox() {

        lights = 0;
        ioByte = 0;
        guiArray = ByteBuffer.allocate(42);
        guiArray.clear();
        setOceanDepthEcho(0);
        setMode((byte) 0);
        setDepthSetPoint(0);
        setPressure(0);
        setPressureDepth(0);
        setRoll(0);
        setPitch(0);
        setGains(0, 0, 0);
        setSpeed(0);

    }
   /**
    * Flag toii be used for checking if values have been written before
retrieving, Lock
    * @param b flag value param
    */
    public void setAvailable(boolean b) {
        available = b;
    }

    /**
     * Set proportianal gain values
     *
     * @param low low gain
     * @param medium medium gain
     * @param high high gain
     */
    public synchronized void setKpGain(double low, double medium, double high)
```

```
{
        kpLow = low;
        kpMed = medium;
        kpHigh = high;

    }

    /**
     * set integrator gain values
     *
     * @param low low gain
     * @param medium medium gain
     * @param high high gain
     */
    public synchronized void setKiGain(double low, double medium, double high)
{
        kiLow = low;
        kiMed = medium;
        kiHigh = high;

    }

    /**
     * Set derivative gain values
     *
     * @param low low gain
     * @param medium medium gain
     * @param high high gain
     */
    public synchronized void setKdGain(double low, double medium, double high)
{
        kdLow = low;
        kdMed = medium;
        kdHigh = high;

        available = true;

    }

    /**
     * Set the runstatus for the i2c
```

```
     *
     * @param run Set the runstatus for the i2c
     */
    public synchronized void setI2cRunStatus(boolean run) {
        this.run = run;
    }

    /**
     * set the status of the actuator
     *
     * @param status status of the actuatror running or not
     */
    public synchronized void setActuatorStatus(byte status) {

        guiArray.put(0, status);

    }

    /**
     * set the mode of the light
     *
     * @param lights mode of the lights, if turned on or not
     */
    public synchronized void setLights(byte lights) {
        System.out.println("light " + lights);
        this.lights = lights;
        setChanged();
        notifyObservers();
    }

    /**
     * set the pressure of the surroundings of the ROV
     *
     * @param press pressure around the ROV
     */
    public synchronized void setPressure(float press) {

        guiArray.putFloat(13, press);

    }
```

```
    /**
     * Sets the temperature of the surroundings
     *
     * @param temp Temperature of surroundings
     */
    public synchronized void setTemperature(float temp) {
        guiArray.putFloat(9, temp);
    }

    /**
     * set the depth of the ROV from Pressure
     *
     * @param depth depth of the ROV from the pressure sensor
     */
    public synchronized void setPressureDepth(float depth) {
        System.out.println("dybde inn " + depth);
        if ((depth > 0) && (depth < 51)) {
       /*     System.out.println("incoming depth "  + depth);

            Integer tempP = Math.round(depth*1000);
            System.out.println("runa depth " + tempP);
            if (tempP < 10000) {
                pressDepthString = "0" + tempP.toString();
                if (tempP == 0) {
                    pressDepthString = "00000";
                }
            } else {
                pressDepthString = tempP.toString();
            }
*/
            System.out.println("dybde satt " + depth);
            guiArray.putFloat(1, depth);
        }

    }

    /**
     * Set the depth of ROV from Echo sounder
     *
     * @param depth depth of based on the echo sounder
     */
```

```
    public synchronized void setOceanDepthEcho(float depth) {

        Integer tempEchoDepth = Math.round(depth * 1000);
        if (tempEchoDepth < 10000) {
            echoDepthString = "0" + tempEchoDepth.toString();

            if (tempEchoDepth == 0) {
                echoDepthString = "00000";

            }
        } else {
            echoDepthString = tempEchoDepth.toString();
        }

        guiArray.putFloat(5, depth);

    }

    /**
     * sets the target depth for the ROV to go to, based on command from GUI
     *
     * @param setPoint setpoint received from client.
     */
    public synchronized void setDepthSetPoint(float setPoint) {

        Integer tempDepthSet = Math.round(setPoint * 1000);

        if (tempDepthSet < 10000) {
            setPointString = "0" + tempDepthSet.toString();
            // in order to have the correct number of characters
            if (tempDepthSet == 0) {
                setPointString = "00000";
            }
        } else {
            setPointString = tempDepthSet.toString();
        }

        //  System.out.println("setpoint " + setPoint);
    }

    /**
```

```
     * Determines if the setpoint is meters from seabed, or depth
     *
     * @param mode mode of the setPoint
     */
    public synchronized void setMode(byte mode) {
        if (mode == 1) {
            mod = '1';
        } else {
            mod = '0';
        }

    }

    /**
     * set the speed of the ROV
     * @param speed speed as input
     */
    public synchronized void setSpeed(float speed) {

        this.speed = speed;
    }

    /**
     * Set the roll value of ROV
     *
     * @param roll roll value from IMU
     */
    public synchronized void setRoll(float roll) {

        guiArray.putFloat(17, roll);

    }

    /**
     * set the pitch value ofROV
     *
     * @param pitch pitch value from the IMu
     */
    public synchronized void setPitch(float pitch) {

        guiArray.putFloat(21, pitch);
```

```
    }

    /**
     * Sets the digital outputs of the ROV
     *
     * @param outputs value containing the output digial output values
     */
    public synchronized void setDigitalOutputs(byte outputs) {

        ioByte = outputs;
    }

    /**
     * Sets the value of the analog input corresponding with the index
parameter
     *
     * @param index Index of the analog sensor (index 0-3)
     * @param value Value of the analog sensor
     */
    public synchronized void setAnalogInputs(int index, float value) {
        guiArray.putFloat(26 + (index * 4), value);
    }

    /**
     * set the gain values of the PID regulator.
     *
     * @param kp propotianal gain of the PID regulator
     * @param ki integral gain of the PID regulator
     * @param kd derivative gain of the PID regulator
     */
    public synchronized void setGains(float kp, float ki, float kd) {

        kp = kp * 10;
        ki = ki * 100;
        kd = kd * 100;
        Short tempKp = (short) Math.round(kp);

        Short tempKi = (short) Math.round(ki);
        Short tempKd = (short) Math.round(kd);
```

```
        if (tempKp < 10) {
            kpString = "0" + tempKp.toString();

        } else {
            kpString = tempKp.toString();
        }
        if (tempKi < 10) {
            kiString = "0" + tempKi.toString();
        } else {
            kiString = tempKi.toString();
        }
        if (tempKd < 10) {
            kdString = "0" + tempKd.toString();
        } else {
            kdString = tempKd.toString();
        }

    }

    /**
     * set the status of teh leak sensor Put status into the guiarray to be
sent
     * ot the user interface
     *
     * @param isLeak set the status of leakage in sensopr
     */
    public synchronized void setLeakStatus(boolean isLeak) {
        // set leak to 0, if leak set variable to 1 to indicate leak
        byte leak = 0;
        if (isLeak == true) {
            leak = 1;

        }
        guiArray.put(25, leak);
    }
////////////////////////////////////////////////////////////////////////////
//////////////////////7
    // Get methods

    /**
     * return runstatus i2c
```

```
     *
     * @return return runstatus i2c
     */
    public synchronized boolean getI2CRunStatus() {
        return this.run;
    }

    /**
     * Retrieve the regulator buffer from the chararray
     *
     * @return return the regulatorArray which contains values used by the PID
     */
    public synchronized byte[] getRegulatorBuffer() {
        c_arr = buildCharArray();
        byte[] byteA = new byte[c_arr.length];
        for (int i = 0; i < c_arr.length; i++) {
            byteA[i] = (byte) (c_arr[i]);
        }
        return byteA;
    }

    /**
     * return the speed of the ROV
     *
     * @return return speed of ROV
     */
    public synchronized float getSpeed() {
        return this.speed;
    }

    /**
     * Build a total String of the values to be sent to ROV Convert them to
char
     * array
     *
     * @return return char array representing String values
     */
    public synchronized char[] buildCharArray() {
        // Use stringBuilder tool to build one String from many
        StringBuilder builder = new StringBuilder();
        builder.append(kpString);
```

```
        builder.append(kdString);
        builder.append(kiString);
//        builder.append(pressDepthString);
        builder.append(echoDepthString);
        builder.append(setPointString);
        builder.append(mod);

        String fullString = builder.toString();

        return fullString.toCharArray();

    }

    /**
     * Return the IO byte
     *
     * @return return the byte containing the values of connected senors and
     * outputs
     */
    public synchronized byte getIo() {
        return ioByte;

    }

    /**
     * Information about the ROV to be sent to the surface
     *
     * @return return the array of information to e sent to the GUI.
     */
    public synchronized ByteBuffer getGuiArray() {

        return guiArray;

    }

    /**
     * Return the status of the lights on the ROV
     *
     * @return Return the status of the lights on the ROV
     */
    public synchronized byte getLightStatus() {
```

```
        return lights;
    }

    /**
     * Get available status for fuzzy
     *
     * @return return avaialble status or fuzzy
     */
    public synchronized boolean getAvailable() {
        return available;
    }

    /**
     * Get kpGain as bytebuffer
     *
     * @return return KpGain as bytebuffer
     */
    public synchronized ByteBuffer getKpGain() {
        ByteBuffer bb = ByteBuffer.allocate(24);

        bb.putDouble(0, kpLow);
        bb.putDouble(8, kpMed);
        bb.putDouble(16, kpHigh);
        available = false;

        return bb;
    }

    /**
     * Get kiGain as bytebuffer
     *
     * @return return KiGain as bytebuffer
     */
    public synchronized ByteBuffer getKiGain() {
        ByteBuffer bb = ByteBuffer.allocate(24);

        bb.putDouble(0, kiLow);
        bb.putDouble(8, kiMed);
        bb.putDouble(16, kiHigh);
        available = false;
```

```
        return bb;
    }

    /**
     * Get KdGain as bytebuffer
     *
     * @return return KdGain as bytebuffer
     */
    public synchronized ByteBuffer getKdGain() {
        ByteBuffer bb = ByteBuffer.allocate(24);

        bb.putDouble(0, kdLow);
        bb.putDouble(8, kdMed);
        bb.putDouble(16, kdHigh);
        available = false;

        return bb;
    }

}
```

```
/*
 * 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
 * sesnors. There will also be a video stream from the ROV.
 *
 * The system consists of a Raspberry Pi on the ROV that is connected to
several
 * Arduino microcontrollers that are connected to sensors and and programed
with
 * the control system.
 * The external computer which is on the surface vessel is connected to a GPS,
 * echo sounder and the RBPi. It will present and save data in addition to
 * handle user commands.
 */


import ExternalCommunication.ConnectionServer;
import ExternalCommunication.StorageBox;
import I2CPackage.I2CBusComm;
import I2CPackage.IMUValues;
import I2CPackage.PressureValues;
import java.util.concurrent.Executors;
import java.util.concurrent.ScheduledExecutorService;
import java.util.concurrent.TimeUnit;

/**
 * Class that initalizes the ROV Java application, create objects and start
 * threads.
 *
 * @author Kristian Homdorm, Morten Solli,marnon
 */
public class Main {




    public static void main(String[] args) throws Exception {

        ScheduledExecutorService scheduleder =
Executors.newScheduledThreadPool(7);
```

```
        StorageBox sb = new StorageBox();
        ConnectionServer listener = new ConnectionServer(sb);

        I2CBusComm i2c = new I2CBusComm(sb);

        IMUValues imu = new IMUValues(i2c, sb);
       // PressureValues press = new PressureValues(i2c, sb);
        i2c.addObserver(imu);
       // i2c.addObserver(press);
        Gpio gpio = new Gpio(sb);
        sb.addObserver(gpio);
        scheduleder.scheduleAtFixedRate(i2c, 3000, 50, TimeUnit.MILLISECONDS);
        scheduleder.scheduleAtFixedRate(listener, 0, 60,
TimeUnit.MILLISECONDS);
        FuzzyLiteController sugeno = new FuzzyLiteController(sb);
        scheduleder.scheduleAtFixedRate(sugeno, 0, 50, TimeUnit.MILLISECONDS);
        scheduleder.scheduleAtFixedRate(gpio, 0, 20, TimeUnit.MILLISECONDS);

    }




    }
```

```
/*
 * 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
 * sesnors. There will also be a video stream from the ROV.
 *
 * The system consists of a Raspberry Pi on the ROV that is connected to
several
 * Arduino microcontrollers that are connected to sensors and and programed
with
 * the control system.
 * The external computer which is on the surface vessel is connected to a GPS,
 * echo sounder and the RBPi. It will present and save data in addition to
 * handle user commands.
 */


import java.text.SimpleDateFormat;
import java.util.Calendar;
import java.util.Date;
import java.util.TimeZone;

public class TimeUtil {

public static String decHoursToDMS(double decimalHours) {
String s = "";
int hours = (int)Math.floor(decimalHours);
double min = (decimalHours - hours) * 60D;
double sec = (min - Math.floor(min)) * 60D;
s = String.format("%02d:%02d:%02d", hours, (int)Math.floor(min),
(int)Math.round(sec));
return s;
}

public static Date getGMT() {
Date now = new Date();
return getGMT(now);
}

public static Date getGMT(Date d) {
Date now = d;
```

```
Date gmt = null;
String tzOffset = (new SimpleDateFormat("Z")).format(now);
//  System.out.println("tz:" + tzOffset);
int offset = 0;
try {
if (tzOffset.startsWith("+")) {
tzOffset = tzOffset.substring(1);
}
offset = Integer.parseInt(tzOffset);
} catch (NumberFormatException nfe) {
nfe.printStackTrace();
}
if (offset != 0) {
long value = offset / 100;
long longDate = now.getTime();
longDate -= value * 3_600_000L;
gmt = new Date(longDate);
} else {
gmt = now;
}
return gmt;
}

public static int getGMTOffset() {
Date now = new Date();
String tzOffset = (new SimpleDateFormat("Z")).format(now);
int offset = 0;
try {
if (tzOffset.startsWith("+"))
tzOffset = tzOffset.substring(1);
offset = Integer.parseInt(tzOffset);
} catch (NumberFormatException nfe) {
nfe.printStackTrace();
}
return offset / 100;
}

public static int getLocalGMTOffset() {
TimeZone tz = Calendar.getInstance().getTimeZone();
Date now = new Date();
SimpleDateFormat sdf = new SimpleDateFormat("Z");
```

```
sdf.setTimeZone(TimeZone.getDefault());
String tzOffset = sdf.format(now);
int offset = 0;
try {
if (tzOffset.startsWith("+"))
tzOffset = tzOffset.substring(1);
offset = Integer.parseInt(tzOffset);
} catch (NumberFormatException nfe) {
nfe.printStackTrace();
}
Calendar.getInstance().setTimeZone(tz);
return offset / 100;
}

public static String getTimeZoneLabel() {
return (new SimpleDateFormat("z")).format(new Date());
}

/**
 * @param howMuch in ms.
 */
public static void delay(long howMuch) {
try {
Thread.sleep(howMuch);
} catch (InterruptedException ie) {
ie.printStackTrace();
}
}
/*

public static void main(String args[]) {
SimpleDateFormat sdf = new SimpleDateFormat("EEE, d MMM yyyy HH:mm:ss");
BufferedReader stdin = new BufferedReader(new InputStreamReader(System.in));
String retString = "";
String prompt = "?> ";
System.err.print(prompt);
try {
retString = stdin.readLine();
} catch (Exception e) {
System.out.println(e);
}
```

```
System.out.println("Your GMT Offset:" + Integer.toString(getGMTOffset()) + "
hours");
System.out.println("Current Time is : " + (new Date()).toString());
System.out.println("GMT is          : " + sdf.format(getGMT()) + " GMT");
System.out.println("");
prompt = "Please enter a year [9999]       > ";
int year = 0;
int month = 0;
int day = 0;
int h = 0;
System.err.print(prompt);
try {
retString = stdin.readLine();
} catch (Exception e) {
System.out.println(e);
}
try {
year = Integer.parseInt(retString);
} catch (NumberFormatException numberformatexception) {
}
prompt = "Please enter a month (1-12) [99] > ";
System.err.print(prompt);
try {
retString = stdin.readLine();
} catch (Exception e) {
System.out.println(e);
}
try {
month = Integer.parseInt(retString);
} catch (NumberFormatException numberformatexception1) {
}
prompt = "Please enter a day (1-31) [99]   > ";
System.err.print(prompt);
try {
retString = stdin.readLine();
} catch (Exception e) {
System.out.println(e);
}
try {
day = Integer.parseInt(retString);
} catch (NumberFormatException numberformatexception2) {
```

```
}
prompt = "Please enter an hour (0-23) [99] > ";
System.err.print(prompt);
try {
retString = stdin.readLine();
} catch (Exception e) {
System.out.println(e);
}
try {
h = Integer.parseInt(retString);
} catch (NumberFormatException numberformatexception3) {
}
Calendar cal = Calendar.getInstance();
cal.set(year, month - 1, day, h, 0, 0);
System.out.println("You've entered:" + sdf.format(cal.getTime()));
int gmtOffset = 0;
prompt = "\nPlease enter the GMT offset for that date > ";
System.err.print(prompt);
try {
retString = stdin.readLine();
} catch (Exception e) {
System.out.println(e);
}
try {
gmtOffset = Integer.parseInt(retString);
} catch (NumberFormatException numberformatexception4) {
}
Date d = cal.getTime();
long lTime = d.getTime();
lTime -= 3_600_000L * gmtOffset;
System.out.println("GMT for your date:" + sdf.format(new Date(lTime)));

System.out.println();
Date now = new Date();
System.out.println("Date:" + now.toString());
System.out.println("GTM :" + (new SimpleDateFormat("yyyy MMMMM dd HH:mm:ss
'GMT'").format(getGMT(now))));

System.out.println("To DMS:" + decHoursToDMS(13.831260480533272));
}*/
}
```

```
/*
 * 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
 * sesnors. There will also be a video stream from the ROV.
 *
 * The system consists of a Raspberry Pi on the ROV that is connected to
several
 * Arduino microcontrollers that are connected to sensors and and programed
with
 * the control system.
 * The external computer which is on the surface vessel is connected to a GPS,
 * echo sounder and the RBPi. It will present and save data in addition to
 * handle user commands.
 */


import ExternalCommunication.StorageBox;
import java.util.Observable;
import java.util.Observer;
import java.util.logging.Level;
import java.util.logging.Logger;

/**
 * This class has responisblity of Converting  value obtained from IMU over
i2c bus to roll and pitch values using both gyroscope and accelerometer
 * By the usage of lowPass filters and a Complimentary filter.
 * Sensor fusing class
 * @author Kristian Homdrom
 */
public class IMUValues implements Observer {

    private float SENSORS_GRAVITY_STANDARD = 9.80665f;
    private final static float _lsm303Accel_MG_LSB = 0.0078F; // 1, 2, 4 or 12
mg per lsb

    private final static double GYRO_WEIGHT_VALUE = 0.40d;
    private final static double ACCEL_WEIGHT_VALUE = 0.40d;
    private I2CBusComm i2c;

    private static volatile double pitch = 0D, roll = 0D, heading = 0D;
```

```
    private static float accX = 0, accY = 0, accZ = 0;

    private static double accelPitch = 0, accelRoll = 0, accelYaw = 0;
    private static double xGyroComp = 0, yGyroComp = 0, zGyroComp = 0;
    // Gyro LP filter coefficents
    private static volatile double oldXGyro = 0, oldYGyro = 0, oldZGyro = 0,
gyroX = 0, gyroY = 0, gyroZ = 0;
    // Accel LP coeficcients
    private static volatile double oldXAccel = 0, oldYAccel = 0, oldZAccel =
0, accelX = 0, accelY = 0, accelZ = 0;
    private static volatile double tau = 50.0;
    private static volatile double dT;
    private static long t1 = 0, t2 = 0;
    private StorageBox sb;
   /**
     * Constructor for Pressure Values class
     *
     * @param i2c i2c link to the I2CBusComm class
     * @param sb StorageBox linkt to the Storagebox class
     */
    public IMUValues(I2CBusComm i2c,StorageBox sb) {
        this.i2c = i2c;
        this.sb=sb;

    }
    /**
     * Convert the the G values of the accelerometer to degrees using atan2
and
     * toDegrees Math functions
     */
    public void accelConvertToRollDegree() {
        //apply the mathematics to get pitch and roll

        accelPitch = Math.toDegrees(Math.atan(accelX / Math.sqrt((accelY *
accelY) + (accelZ * accelZ))));
        accelRoll = Math.toDegrees(Math.atan(accelY / Math.sqrt((accelX *
accelX) + (accelZ * accelZ))));

    }

    /**
```

```
     * Apply low pass filter to acceleromter Get rid of unwanted low changes (
     * etc, vibration)
     */
    private void accelLPFilter() {

        //compensate the value with the weight values
        // Simple lowpass filter
        accelX = accX * ACCEL_WEIGHT_VALUE + (oldXAccel * (1.0d -
ACCEL_WEIGHT_VALUE));
        oldXAccel = accelX;

        accelY = accY * ACCEL_WEIGHT_VALUE + (oldYAccel * (1.0d -
ACCEL_WEIGHT_VALUE));
        oldYAccel = accelY;

        accelZ = accZ * ACCEL_WEIGHT_VALUE + (oldZAccel * (1.0d -
ACCEL_WEIGHT_VALUE));
        oldZAccel = accelZ;
        accelConvertToRollDegree();
    }

    /**
     * Apply low pass filter to gyrovalues Get rid of unwanted low changes (
     * etc, vibration)
     */
    private void gyroLPFilter() {

        //compensate the value with the weight values
        // Simple lowpass filter
        gyroX = xGyroComp * GYRO_WEIGHT_VALUE + (oldXGyro * (1.0d -
GYRO_WEIGHT_VALUE));
        oldXGyro = gyroX;

        gyroY = yGyroComp * GYRO_WEIGHT_VALUE + (oldYGyro * (1.0d -
GYRO_WEIGHT_VALUE));
        oldYGyro = gyroY;

        gyroZ = zGyroComp * GYRO_WEIGHT_VALUE + (oldZGyro * (1.0d -
GYRO_WEIGHT_VALUE));
        oldZGyro = gyroZ;
        complimentaryFilter();
```

```
    }

    /**
     * Use a complimentary filter to get the roll and pitch values from IMU
     */
    private void complimentaryFilter() {
        // combine the two sensors of gyro and accel to get best value
        // T= timeConstant
        //T=alpha*dT/(1-alpha)
        t2 = System.currentTimeMillis();
        // Constant to specify how much of each sensor to use, see
Complimentary filter in thesis
        double alpha = 0.98;
        dT = (double) t2 - t1;

        // dT is time to get value from Gyro, while tau is the time set of the
thread to run at.
        // Calculate real roll
        roll = (alpha) * (roll + gyroX * (dT + tau) / 1000.0) + (1 - alpha) *
accelRoll;
       // Calucate real pitch
        pitch = (alpha) * (pitch - gyroY * (dT + tau) / 1000.0) + (1 - alpha)
* accelPitch;
       // send values
        sendPitch();
        sendRoll();

    }

    /**
     * send the pitch of the ROV
     *
     */
    public void sendPitch() {
      sb.setPitch((float)pitch);
    }

    /**
     * send the roll of the ROV
     *
     */
```

```
    public void sendRoll() {
    sb.setRoll((float)roll);
    }

    public double getHeading() {
        return this.heading;
    }
/**
 * Iterate bytearray to generate  number, little endian
 * @param list list to extract values from
 * @param idx index number retrieving number from
 * @return return number
 */
    private static int accel12(byte[] list, int idx) {

        int n = ((list[idx + 1] & 0xFF) << 8) | (list[idx] & 0xFF); // Low,
high bytes
        if (n > 32767) {
            n -= 65536;              // 2's complement signed
        }
        return n >> 4;                          // 12-bit resolution
    }
/**
 * Method for magnetcompass, NIU,
 * @param list
 * @param idx
 * @return
 */
    private static int mag16(byte[] list, int idx) {
        int n = ((list[idx] & 0xFF) << 8) | (list[idx + 1] & 0xFF);   // High,
low bytes
        return (n < 32768 ? n : n - 65536);                           // 2's
complement signed
    }

    @Override
    public void update(Observable o, Object o1) {

        //convert raw accelValues to G-values in accrod to earth gravity
        accX = (float) accel12(i2c.getAccel(), 0) * _lsm303Accel_MG_LSB *
SENSORS_GRAVITY_STANDARD;
```

```
        accY = (float) accel12(i2c.getAccel(), 2) * _lsm303Accel_MG_LSB *
SENSORS_GRAVITY_STANDARD;
        accZ = (float) accel12(i2c.getAccel(), 4) * _lsm303Accel_MG_LSB *
SENSORS_GRAVITY_STANDARD;
        //Filter out noise with low pass filter
        accelLPFilter();
        //start timer ,for gyro dT loop
        t1 = System.currentTimeMillis();
        double gyroArray[] = new double[3];
        try {
            //Get Gyro values
            gyroArray = i2c.getCalOutValue();

            xGyroComp = gyroArray[0];
            yGyroComp = gyroArray[1];
            zGyroComp = gyroArray[2];
        } catch (Exception ex) {
            Logger.getLogger(IMUValues.class.getName()).log(Level.SEVERE,
null, ex);
        }
      // Filter out noise with low pass filter
        gyroLPFilter();

    }
}
```

```
/*
 * 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
 * sesnors. There will also be a video stream from the ROV.
 *
 * The system consists of a Raspberry Pi on the ROV that is connected to
several
 * Arduino microcontrollers that are connected to sensors and and programed
with
 * the control system.
 * The external computer which is on the surface vessel is connected to a GPS,
 * echo sounder and the RBPi. It will present and save data in addition to
 * handle user commands.
 */


/**
 * This class has responsibility to decode commands sent from user interface
and
 * send values to the Storagebox
 *
 * @author Kristian Homdrom
 */
public class Parser {

    private final String SETPOINTID = "<Setpoint>";
    private final String LIGHTSID = "<Lights>";
    private final String MODEID = "<Mode>";
    private final String DIGITALOUTID = "<DigitalOut>";
    private final String KP = "<KP>";
    private final String KI = "<KI>";
    private final String KD = "<KD>";

    /**
     * Decode String sent from User interface Use switchCase to identyfy what
     * merthod in Storagebox to send to
     *
     * @param command Command sent from the user interface
     * @param sb Link to StorageBox to send the decoded commands
     */
```

```
    public void parseCommand(String command, StorageBox sb) {
        // split with positive workaround
        String[] parts = command.split("(?<=)");
        String[] gains = parts[1].split(" ");
        switch (parts[0]) {
            case SETPOINTID:
                sb.setDepthSetPoint(Float.parseFloat(parts[1]));
                break;
            case LIGHTSID:
                sb.setLights(Byte.valueOf(parts[1]));
                break;
            case MODEID:
                sb.setMode(Byte.valueOf(parts[1]));
                break;
            case DIGITALOUTID:
                sb.setDigitalOutputs(Byte.valueOf(parts[1]));
                break;
            case KP:
                sb.setKpGain(Float.parseFloat(gains[0]),
Float.parseFloat(gains[1]), Float.parseFloat(gains[2]));

                break;
            case KI:
                sb.setKiGain(Float.parseFloat(gains[0]),
Float.parseFloat(gains[1]), Float.parseFloat(gains[2]));
                break;
            case KD:
                sb.setKdGain(Float.parseFloat(gains[0]),
Float.parseFloat(gains[1]), Float.parseFloat(gains[2]));
                break;

        }
    }
}
```

```
/*
 * 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
 * sesnors. There will also be a video stream from the ROV.
 *
 * The system consists of a Raspberry Pi on the ROV that is connected to
several
 * Arduino microcontrollers that are connected to sensors and and programed
with
 * the control system.
 * The external computer which is on the surface vessel is connected to a GPS,
 * echo sounder and the RBPi. It will present and save data in addition to
 * handle user commands.
 */


import com.pi4j.io.gpio.GpioController;
import com.pi4j.io.gpio.GpioFactory;
import com.pi4j.io.gpio.GpioPinDigitalOutput;
import com.pi4j.io.gpio.PinState;
import com.pi4j.io.gpio.RaspiPin;
import java.util.Observable;
import java.util.Observer;
import ExternalCommunication.StorageBox;

import com.pi4j.io.gpio.GpioPinDigitalInput;
import com.pi4j.io.gpio.PinPullResistance;
import java.util.logging.Level;
import java.util.logging.Logger;

/**
 * Class that controls the digital GPIO pins of the raspberry Control the
light
 * of the ROV and check if there has been a leak in the control box
 *
 * @author marno/Kristian Homdrom
 */
public class Gpio implements Runnable, Observer {

    private final GpioController controller;
```

```
    private final GpioPinDigitalOutput ledPin;
    private final GpioPinDigitalInput leakPin;

    private byte lights = 0;
    private StorageBox sb;

    /**
     * Constructor which intilialize the chosen GPIO pins. Gpio pins get a
     * unique name and their reistiance set to either pull down or pull up.
     *
     * @param sb Storagebox parameter continued from startup to provide link
to
     * storagebox
     */
    public Gpio(StorageBox sb) {
        // Create a link to the GPIO outputs using PI4J library and WiringPi
        controller = GpioFactory.getInstance();
        // Create variables for the digital GPIO pins   // GPIO_04 = pin 16
        ledPin = controller.provisionDigitalOutputPin(RaspiPin.GPIO_04,
"PinLED", PinState.LOW);

        leakPin = controller.provisionDigitalInputPin(RaspiPin.GPIO_05, // PIN
NUMBER-18
                "LeakSensor", // PIN FRIENDLY NAME (optional)
                PinPullResistance.PULL_DOWN); // PIN RESISTANCE (optional)

        this.sb = sb;
    }

    /**
     * check the Gpio for water leakage If true set flag true if not reset.
     */
    public void checkForLeakage() {

        if (leakPin.getState().isHigh() == true) {
            sb.setLeakStatus(true);

        } else {
            sb.setLeakStatus(false);

        }
```

```
    }

    @Override
    public void update(Observable o, Object o1) {
        lights = sb.getLightStatus();
        if (lights == 0) {

            ledPin.low();

            try {
                Thread.sleep(10);
            } catch (InterruptedException ex) {
                Logger.getLogger(Gpio.class.getName()).log(Level.SEVERE, null,
ex);
            }

        } else {
            ledPin.high();

            try {
                Thread.sleep(10);
            } catch (InterruptedException ex) {
                Logger.getLogger(Gpio.class.getName()).log(Level.SEVERE, null,
ex);
            }

        }

    }

    @Override
    public void run() {
        checkForLeakage();
    }
}
```

```
/*
 * 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
 * sesnors. There will also be a video stream from the ROV.
 *
 * The system consists of a Raspberry Pi on the ROV that is connected to
several
 * Arduino microcontrollers that are connected to sensors and and programed
with
 * the control system.
 * The external computer which is on the surface vessel is connected to a GPS,
 * echo sounder and the RBPi. It will present and save data in addition to
 * handle user commands.
 */


import ExternalCommunication.StorageBox;
import com.pi4j.io.i2c.I2CBus;
import com.pi4j.io.i2c.I2CDevice;
import com.pi4j.io.i2c.I2CFactory;
import static I2CPackage.BitOps.twosComplementToByte;
import static I2CPackage.TimeUtil.delay;
import java.io.IOException;
import java.nio.ByteBuffer;
import java.nio.ByteOrder;
import java.util.Observable;
import java.util.logging.Level;
import java.util.logging.Logger;

/**
 * This class represents the master I2C device and intializes the I2C bus
 * located in the ROV Has as target to request and send information over i2c
to
 * the connected devices. Calibration of sensors also have to be done from
here
 * Snesor values are obtained then transferred to designated classes for
 * processing. Information to PID regulator are obtained from storagebox and
 * sent from here.
 *
 * @author Kristian Homdrom
```

```
 */
public class I2CBusComm extends Observable implements Runnable {

    // I2C adresses of standarad devices on the I2c bus
    public final static int LSM303_ADDRESS_ACCEL = 0x19; // 0011001x, 0x19
    public final static int LSM303_ADDRESS_MAG = (0x3C >> 1); // 0011110x,
0x1E <- that is an HMC5883L !
    public final static int L3GD20_ADRESS_GYRO = 0x6b;
    public final static int MS5837_ADDRESS = 0x76;
    public final static int REGULATOR_ADDRESS = 0x09;
    public final static int IO_ADDRESS = 0x07;
    public final static int ECHO_SOUNDER_ADDRESS = 0x04;

    //Register to write to to start the accelerometer
    public final static int LSM303_REGISTER_ACCEL_CTRL_REG1_A = 0x20; //
00000111   rw
    //Retgister tro write to to set the resolution of the accelerometer, high
or low
    public final static int LSM303_REGISTER_ACCEL_CTRL_REG4_A = 0x23; //
00000000   rw
    // Register of the accelerometer to write to to prepare to send x,y,z raw
data to the raspberry.
    public final static int LSM303_REGISTER_ACCEL_OUT_X_L_A = 0x28;

    // register of teh magnetometer to write set gain  to be used  (Gauss).
    public final static int LSM303_REGISTER_MAG_CRB_REG_M = 0x01;
    //Register of the magnetometer to start up the magnetometer
    public final static int LSM303_REGISTER_MAG_MR_REG_M = 0x02;
    // Register of the magnetometer wto write to o write to to prepare to send
x,y,z raw data to the raspberry.
    public final static int LSM303_REGISTER_MAG_OUT_X_H_M = 0x03;

    // Gyro Register for starting the x,y,z and power down modes.
    public final static int L3GD20_REGISTER_GYRO_CTRL_REG1 = 0x20;
    // Register to accessing gyro range settings
    public final static int L3GD20_REGISTER_GYRO_RANGE = 0x23;
    // 1, 2, 4 or 12 mg per lsb
    private final static float _lsm303Accel_MG_LSB = 0.001F;
    private static float _lsm303Mag_Gauss_LSB_XY = 1100.0F;  // Varies with
gain
    private static float _lsm303Mag_Gauss_LSB_Z = 980.0F;  // Varies with gain
```

```
    // gain setting for the magnetometer
    // Gain settings for setMagGain()
    public final static int LSM303_MAGGAIN_1_3 = 0x20; // +/- 1.3 //standanrd?
    public final static int LSM303_MAGGAIN_1_9 = 0x40; // +/- 1.9
    public final static int LSM303_MAGGAIN_2_5 = 0x60; // +/- 2.5
    public final static int LSM303_MAGGAIN_4_0 = 0x80; // +/- 4.0
    public final static int LSM303_MAGGAIN_4_7 = 0xA0; // +/- 4.7
    public final static int LSM303_MAGGAIN_5_6 = 0xC0; // +/- 5.6
    public final static int LSM303_MAGGAIN_8_1 = 0xE0; // +/- 8.1

    public final static int L3GD20_REG_R_OUT_X_L = 0x28; // X-axis angular
data rate LSB
    public final static int L3GD20_REG_R_OUT_X_H = 0x29; // X-axis angular
data rate MSB
    public final static int L3GD20_REG_R_OUT_Y_L = 0x2a; // Y-axis angular
data rate LSB
    public final static int L3GD20_REG_R_OUT_Y_H = 0x2b; // Y-axis angular
data rate MSB
    public final static int L3GD20_REG_R_OUT_Z_L = 0x2c; // Z-axis angular
data rate LSB
    public final static int L3GD20_REG_R_OUT_Z_H = 0x2d; // Z-axis angular
data rate MSB

    private final static int L3GD20_REG_R_STATUS_REG = 0x27; // Status
register
    public final static int L3GD20_MASK_STATUS_REG_ZYXDA = 0x08; // Z, Y, X
data available
    public final static int L3GD20_MASK_STATUS_REG_ZDA = 0x04; // Z data
available
    public final static int L3GD20_MASK_STATUS_REG_YDA = 0x02; // Y data
available
    public final static int L3GD20_MASK_STATUS_REG_XDA = 0x01; // X data
available
    //Enter the standard gravity force that will be working on the IMU--> // <
Earth's gravity in m/s^2
    private float SENSORS_GRAVITY_STANDARD = 9.80665f;
    private static double gyroGain = 0;
    // I2c bus to be used
    private I2CBus bus;
    private StorageBox sb;
```

```
    // I2C Devices to be used.
    private I2CDevice accelerometer = null, magnetometer = null, depthSensor =
null, gyro = null, regulator = null, iO = null, echoSounder = null;
    private byte[] accelData, magData, pressData, gyroData, echoData;
    private int[] calibration;
    private static volatile int counter = 0;

    // calibration params for pressure sensor;//PROM_READ
    private int C1, C2, C3, C4, C5, C6;

    private final static double SENOSR_TO_DPS_RANGE_250 = 0.00875;
    private final static double SENOSR_DPS_TO_RADS = 0.017453293d;

    // calibration
    // For calibration purposes
    private double meanX = 0;
    private double maxX = 0;
    private double minX = 0;
    private double meanY = 0;
    private double maxY = 0;
    private double minY = 0;
    private double meanZ = 0;
    private double maxZ = 0;
    private double minZ = 0;

    /**
     * Constructor of the I2CBusComm class Get every instance of connected i2c
     * devices and create variables for them to hold Send enable signals for
the
     * sensors connected so they can start taking values. Calibrate sensors
that
     * need it.
     *
     * @param sb StorageBox input to generate link to storagbox to place
valuies
     * obrtained
     */
    public I2CBusComm(StorageBox sb) {
        this.sb = sb;
        try {
            // set the bus to equal to bus 1 to be used as comm link
```

```
            bus = I2CFactory.getInstance(I2CBus.BUS_1);

            //Create a connection on the buiis for every device connected with
address
            accelerometer = bus.getDevice(LSM303_ADDRESS_ACCEL);

            //     magnetometer = bus.getDevice(LSM303_ADDRESS_MAG);
           // depthSensor = bus.getDevice(MS5837_ADDRESS);
            ;
            gyro = bus.getDevice(L3GD20_ADRESS_GYRO);

            // Arduinos
            regulator = bus.getDevice(REGULATOR_ADDRESS);
            iO = bus.getDevice(IO_ADDRESS);
          echoSounder = bus.getDevice(ECHO_SOUNDER_ADDRESS);

            // * Start sensing
            // Enable accelerometer
            Thread.sleep(10);
            if (accelerometer != null) {
                // starts the accelerometer
              accelerometer.write(LSM303_REGISTER_ACCEL_CTRL_REG1_A, (byte)
0x57); // Normal mode 100Hz
                // // Low Res. For Hi Res, write 0b00001000 , 0x08
             accelerometer.write(LSM303_REGISTER_ACCEL_CTRL_REG4_A, (byte)
0x17);//full scale reso-->+/- 4G
            }

            Thread.sleep(5);
            // Enable magnetometer
            if (magnetometer != null) {
                // start up the magnetometer
                magnetometer.write(LSM303_REGISTER_MAG_MR_REG_M, (byte) 0x00);
                // set the gain equal to standard value.
                int gain = LSM303_MAGGAIN_1_3;
                setMagGain(gain);

            }
            Thread.sleep(5);

            if (gyro != null) {
```

```
                //Start up gyro, enables x,y,z-axis and disables the power
down mode
                gyro.write(L3GD20_REGISTER_GYRO_CTRL_REG1, (byte) 0x0F);
//Normal mode,all axes enabled
                // Full scale range, 250 dps
                gyro.write(L3GD20_REGISTER_GYRO_RANGE, (byte) 0x00);
                gyroGain = SENOSR_TO_DPS_RANGE_250;
                calibrateGyro();
            }
            Thread.sleep(5);
            /*if (depthSensor != null) {
                calibratePressureSensor();

            }*/

            if (accelerometer == null) {
                System.out.println("error connectijng to accel");
            }
            if (regulator == null) {
                System.out.println("Error connecting to Regulator");
            }
            if (iO == null) {
                System.out.println("error conencting to IO");
            }
            if (echoSounder == null) {
                System.out.println("error connecting to echosounder");
            }

        } catch (Exception e) {
           Logger.getLogger(I2CBusComm.class.getName()).log(Level.SEVERE,
null, e);
        }
    }

    /**
     * Run method of tje thread, will run with set intervals given at start.
     */
    @Override
    public void run() {
        if (accelerometer != null) {
            readAccel();
```

```
        }

    /*      if (this.depthSensor != null) {
              readPressureSensor();

        } else {
            try {
                depthSensor = bus.getDevice(MS5837_ADDRESS);
            } catch (Exception e) {
                System.out.println(e.getMessage());
            }
        }
        */

        if (iO != null) {
            sendIOByte();
            getIOFeedback();

        } else {
            try {
               iO = bus.getDevice(IO_ADDRESS);

            } catch (Exception e) {
                System.out.println(e.getMessage());
            }
        }

        if (counter == 4) {
           if (echoSounder != null) {
                getEchoData();

            } else {
                try {
                    echoSounder = bus.getDevice(ECHO_SOUNDER_ADDRESS);
                } catch (Exception e) {
                    System.out.println(e.getMessage());
                }

            }
            if (regulator != null) {
```

```
                sendRegulatorData();
                regulatorFeedBack();
                counter = 0;
            } else {
                try {
                    regulator = bus.getDevice(REGULATOR_ADDRESS);
                } catch (Exception e) {
                    System.out.println(e.getMessage());
                }
            }

        }
        setChanged();

        notifyObservers();

        counter++;

    }

    /**
     * Set the gain of the magnetometer
     *
     * @param gain gain to be set for the magentometer
     * @throws IOException
     */
    private void setMagGain(int gain) throws IOException {
        magnetometer.write(LSM303_REGISTER_MAG_CRB_REG_M, (byte) gain);

        switch (gain) {
            case LSM303_MAGGAIN_1_3:
                _lsm303Mag_Gauss_LSB_XY = 1_100F;
                _lsm303Mag_Gauss_LSB_Z = 980F;
                break;
            case LSM303_MAGGAIN_1_9:
                _lsm303Mag_Gauss_LSB_XY = 855F;
                _lsm303Mag_Gauss_LSB_Z = 760F;
                break;
            case LSM303_MAGGAIN_2_5:
                _lsm303Mag_Gauss_LSB_XY = 670F;
                _lsm303Mag_Gauss_LSB_Z = 600F;
```

```
                break;
            case LSM303_MAGGAIN_4_0:
                _lsm303Mag_Gauss_LSB_XY = 450F;
                _lsm303Mag_Gauss_LSB_Z = 400F;
                break;
            case LSM303_MAGGAIN_4_7:
                _lsm303Mag_Gauss_LSB_XY = 400F;
                _lsm303Mag_Gauss_LSB_Z = 355F;
                break;
            case LSM303_MAGGAIN_5_6:
                _lsm303Mag_Gauss_LSB_XY = 330F;
                _lsm303Mag_Gauss_LSB_Z = 295F;
                break;
            case LSM303_MAGGAIN_8_1:
                _lsm303Mag_Gauss_LSB_XY = 230F;
                _lsm303Mag_Gauss_LSB_Z = 205F;
                break;
        }
    }

    /**
     * Method to calibrate the pressure sensor to get the correct readings
     *
          */
    public void calibratePressureSensor()  {
        // Reset command
        try {
             depthSensor.write((byte) 0x1E);
        Thread.sleep(200);
        byte[] data = new byte[2];
        // Read 12 bytes of calibration data
        // Read pressure sensitivity
        depthSensor.read(0xA2, data, 0, 2);
        C1 = ((data[0] & 0xFF) * 256 + (data[1] & 0xFF));
         Thread.sleep(200);
        // Read pressure offset
        depthSensor.read(0xA4, data, 0, 2);
        C2 = ((data[0] & 0xFF) * 256 + (data[1] & 0xFF));
             Thread.sleep(200);
        // Read temperature coefficient of pressure sensitivity
        depthSensor.read(0xA6, data, 0, 2);
```

```
        C3 = ((data[0] & 0xFF) * 256 + (data[1] & 0xFF));
             Thread.sleep(200);
        // Read temperature coefficient of pressure offset
        depthSensor.read(0xA8, data, 0, 2);
        C4 = ((data[0] & 0xFF) * 256 + (data[1] & 0xFF));
            Thread.sleep(200);
        // Read reference temperature
        depthSensor.read(0xAA, data, 0, 2);
        C5 = ((data[0] & 0xFF) * 256 + (data[1] & 0xFF));
           Thread.sleep(200);
        // Read temperature coefficient of the temperature
        depthSensor.read(0xAC, data, 0, 2);
        C6 = ((data[0] & 0xFF) * 256 + (data[1] & 0xFF));
             Thread.sleep(200);
        calibration = new int[6];
        calibration[0] = C1;
        calibration[1] = C2;
        calibration[2] = C3;
        calibration[3] = C4;
        calibration[4] = C5;
        calibration[5] = C6;

        } catch (Exception e) {
            System.out.println(e.getMessage());
            System.out.println("calibrering");
            calibratePressureSensor();
        }

    }
////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////
    // Send data to the Connected microcontrollers

    /**
     * Send the IO byte over the i2C to the IO device
     */
    public void sendIOByte() {


        try {
            iO.write(sb.getIo());
```

```
        } catch (IOException ex) {
            Logger.getLogger(I2CBusComm.class.getName()).log(Level.SEVERE,
null, ex);
        }

    }

    /**
     * Send the PID control data to the arduino regulator Retrieve the from
the
     */
    public void sendRegulatorData() {

        try {
            // Create byte array
            byte[] byteA = new byte[sb.getRegulatorBuffer().length];
            byteA = sb.getRegulatorBuffer();
            System.out.println("length " + byteA.length);
            // Send byteArray to the regulator
            regulator.write(byteA, 0, byteA.length);
            //Sleep to give Arduino time to read
            Thread.sleep(20);
        } catch (Exception ex) {
                Logger.getLogger(I2CBusComm.class.getName()).log(Level.SEVERE,
null, ex);

        }
    }

////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////
    //  Read the sensors connected to the I2C bus
    /**
     * Read pressure sensor raw values;
     *
     */
    public void readPressureSensor() {
        try {
            // Send pressure conversion(OSR = 256) command
            depthSensor.write((byte) 0x40);
            Thread.sleep(5);
```

```
            pressData = new byte[6];
            // Read digital pressure value
            // Read 3 bytes of data
            // D1 msb2, D1 msb1, D1 lsb
            depthSensor.read(0x00, pressData, 0, 3);

            // Send temperature conversion(OSR = 256) command
            depthSensor.write((byte) 0x50);
            Thread.sleep(5);

            // Read digital temperature value
            // Read 3 bytes of data

            depthSensor.read(0x00, pressData, 3, 3);

        } catch (Exception e) {
            System.out.println(e.getMessage());
        }

    }

    /**
     * Read the acceleromter for new data.
     */
    public synchronized void readAccel() {

        // values for every angle( x,y,z)  MSB and LSB ( most significant and
least significant)
        accelData = new byte[6];
        magData = new byte[6];

        try {
            if (accelerometer != null) {
                // write to the accelerometer to prepare for read action read
action
                accelerometer.write((byte) (LSM303_REGISTER_ACCEL_OUT_X_L_A |
0x80));

                int r = accelerometer.read(accelData, 0, 6);
```

```
                if (r != 6) {
                    System.out.println("Error reading accel data, < 6 bytes");
                }

            }
        } catch (Exception e) {
            System.out.println(e.getMessage());

        }
    }

    ////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////
/
    // Read data from arduino microcontrollers connected to the I2C bus
    /**
     * Retrieve the feedback fro mthe PID regulator
     */
    public void regulatorFeedBack() {

        try {
             byte b[] = new byte[5];
            regulator.read(b, 0, 5);

            Thread.sleep(10);
               ByteBuffer bb = ByteBuffer.wrap(b);
            bb.order(ByteOrder.LITTLE_ENDIAN);
            float tempF;

                tempF = bb.getFloat();
                sb.setPressureDepth(tempF);


              byte t = bb.get();

            sb.setActuatorStatus(t);

        } catch (Exception ex) {
                  Logger.getLogger(I2CBusComm.class.getName()).log(Level.SEVER
E, null, ex);
```

```
        }
    }

    /**
     * Retrieve the feedback of the io regulator
     *
     */
    public void getIOFeedback() {
        byte b[] = new byte[16];
        try {
            iO.read(b, 0, 16);
            Thread.sleep(10);
            ByteBuffer bb = ByteBuffer.wrap(b);
            bb.order(ByteOrder.LITTLE_ENDIAN);
            float tempF;
            for (int i = 0; i < 4; i++) {
                tempF = bb.getFloat();
                sb.setAnalogInputs(i, tempF);
            }


        } catch (Exception ex) {
            Logger.getLogger(I2CBusComm.class.getName()).log(Level.SEVERE,
null, ex);
        }
    }

    /**
     * Retrieve the echoSounder data
     */
    public void getEchoData() {
        byte b[] = new byte[8];
        try {
            // Read from echo sounder and place data in byte array( b), offset
0, amount of bytes to be read 16
            echoSounder.read(b, 0, 8);
            Thread.sleep(10);
            ByteBuffer bb = ByteBuffer.wrap(b);
            //Bit order to be read
            bb.order(ByteOrder.LITTLE_ENDIAN);
```

```
            float tempDepthF = bb.getFloat();
            float tempSpeedF = bb.getFloat();
            sb.setOceanDepthEcho(tempDepthF);
            sb.setSpeed(tempSpeedF);
            bb.clear();
        } catch (Exception e) {
            Logger.getLogger(I2CBusComm.class.getName()).log(Level.SEVERE,
null, e);
        }
    }

    ////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////////
/
    // return bytearrays of pressure, accelerometer, gyroscope,
PressureCalibration
    /**
     * Return the accelerometer raw data
     *
     * @return Return the accelerometer raw data
     */
    public byte[] getAccel() {
        return accelData;
    }

    /**
     * Return the pressure read data
     *
     * @return Return the pressure read data
     */
    public byte[] getPressVal() {
        return pressData;
    }

    /**
     * returns the raw values of the gyro in degrees/sec
     *
     * @return return the array of axis values of gyro (degree/sec)
     * @throws Exception falied to retrieve data
     */
    public double[] getCalOutValue() throws Exception {
```

```
        return new double[]{this.getCalOutXValue(), this.getCalOutYValue(),
this.getCalOutZValue()};
    }

    /**
     * Returns the calibration parameters for the pressure sensor
     *
     * @return Returns the calibration parameters for the pressure sensor
     */
    public int[] getPressureCalibrationData() {
        return calibration;
    }

////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////
    // methods  to simplify reading
    /**
     * Read from a selected GYRO register
     *
     * @param register register of the gyro to beread
     * @param mask something
     * @return return the bit values
     * @throws Exception failed to read register, throw exception
     */
    public int readFromRegister(int register, int mask) throws Exception {
        int current = readU8(register);
        return BitOps.getValueUnderMask(current, mask);
    }

    /*
 * Read an unsigned byte from the I2C device
     */
    private int readU8(int reg) throws Exception {
        int result = 0;
        try {
            result = this.gyro.read(reg);
        } catch (Exception ex) {
            ex.printStackTrace();
        }
        return result;
    }
```

```
    ////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////
/
    // Read values from gyro, x,y,z,
    /**
     * Read from gyro register x-axis values raw
     * @return return raw values x-axis multiplied by gain
     * @throws Exception  throws  exception if faliure
     */
    private double getRawOutXValue() throws Exception {
        int l = this.readFromRegister(L3GD20_REG_R_OUT_X_L, 0xff);
        int h_u2 = this.readFromRegister(L3GD20_REG_R_OUT_X_H, 0xff);
        int h = twosComplementToByte(h_u2);
        int value = 0;
        if (h < 0) {
            value = (h * 256 - l);
        } else {
            if (h >= 0) {
                value = (h * 256 + l);
            }
        }

        return value * this.gyroGain;
    }
/**
 * Read from Gyro register y-axis value
 * @return return raw values y-axis multplied by gain
 * @throws Exception  throws  exception if faliure
 */
    private double getRawOutYValue() throws Exception {
        int l = this.readFromRegister(L3GD20_REG_R_OUT_Y_L, 0xff);
        int h_u2 = this.readFromRegister(L3GD20_REG_R_OUT_Y_H, 0xff);
        int h = BitOps.twosComplementToByte(h_u2);
        int value = 0;
        if (h < 0) {
            value = (h * 256 - l);
        } else {
            if (h >= 0) {
                value = (h * 256 + l);
            }
```

```
        }
        return value * this.gyroGain;
    }
/**
 * Read from Gyro register z-axis value
 * @return return raw values z-axis multplied by gain
 * @throws Exception  throws  exception if faliure
 */
    private double getRawOutZValue() throws Exception {
        int l = this.readFromRegister(L3GD20_REG_R_OUT_Z_L, 0xff);
        int h_u2 = this.readFromRegister(L3GD20_REG_R_OUT_Z_H, 0xff);
        int h = BitOps.twosComplementToByte(h_u2);
        int value = 0;
        if (h < 0) {
            value = (h * 256 - l);
        } else {
            if (h >= 0) {
                value = (h * 256 + l);
            }
        }
        return value * this.gyroGain;
    }

    ////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////////
/
    // Calibration
    // GYRO calibrateed
    /**
     * geth the max value read from calibration durijng calibration
     *
     * @param da input to be estimated max value
     * @return return the max value estimated from gyro calibration
     */
    private static double getMax(double[] da) {
        double max = da[0];
        for (double d : da) {
            max = Math.max(max, d);
        }
        return max;
    }
```

```
    /**
     * get the min value read from the gyro
     *
     * @param da input to be used as estimation for the min value
     * @return return the min value of the calibration (X,Y,Z) of the gyro
     */
    private static double getMin(double[] da) {
        double min = da[0];
        for (double d : da) {
            min = Math.min(min, d);
        }
        return min;
    }

    /**
     * get the mean value of the calibration of GYRO
     *
     * @param da Values read from GYRo
     * @return return the mean values read from gyro
     */
    private static double getMean(double[] da) {
        double mean = 0;
        for (double d : da) {
            mean += d;
        }
        return mean / da.length;
    }

    /**
     * calibrate the gyroScope to get offset values to be used to reduce drift
     *
     * @throws Exception failed to calibrate gyro throw exception
     */
    public void calibrateGyro() throws Exception {
        this.calibrateX();
        this.calibrateY();
        this.calibrateZ();
    }

    /**
```

```
     * Calibrate the x axis of the Gyro
     *
     * @throws Exception failed to calibrate gyro x-axis throw exception
     */
    public void calibrateX() throws Exception {
        System.out.println("Calibrating X, please do not move the sensor...");
        double[] buff = new double[20];
        for (int i = 0; i < 20; i++) {
            while (this.getAxisDataAvailableValue()[0] == 0) {
                delay(1L);
            }
            buff[i] = this.getRawOutXValue();
        }
        this.meanX = getMean(buff);
        this.maxX = getMax(buff);
        this.minX = getMin(buff);
    }

    /**
     * calibrate the y axis of the gyroscope
     *
     * @throws Exception failed to calibrate gyro y-axis throw exception
     */
    public void calibrateY() throws Exception {
        System.out.println("Calibrating Y, please do not move the sensor...");
        double[] buff = new double[20];
        for (int i = 0; i < 20; i++) {
            while (this.getAxisDataAvailableValue()[1] == 0) {
                delay(1L);
            }
            buff[i] = this.getRawOutYValue();
        }
        this.meanY = getMean(buff);
        this.maxY = getMax(buff);
        this.minY = getMin(buff);
    }

    /**
     * calibrate the z axis of the gyroscope
     *
     * @throws Exception failed to calibrate gyro z-axis throw exception
```

```
     */
    public void calibrateZ() throws Exception {
        System.out.println("Calibrating Z, please do not move the sensor...");
        double[] buff = new double[20];
        for (int i = 0; i < 20; i++) {
            while (this.getAxisDataAvailableValue()[2] == 0) {
                delay(1L);
            }
            buff[i] = this.getRawOutZValue();
        }
        this.meanZ = getMean(buff);
        this.maxZ = getMax(buff);
        this.minZ = getMin(buff);
    }

    /**
     * retrieves the offset values read from the status register of the
     * gyroscope To be used to estimate min,mena,max offset of every axis
     *
     * @return retrieves the offset values read from the status register of
the
     * gyroscope,used to calculate min,mean,max (X,Y,Z)
     * @throws Exception fail to read values from gyro throw exception
     */
    public int[] getAxisDataAvailableValue() throws Exception {
        int zda = 0;
        int yda = 0;
        int xda = 0;
        if (this.readFromRegister(L3GD20_REG_R_STATUS_REG,
L3GD20_MASK_STATUS_REG_ZYXDA) == 0x01) {
            zda = this.readFromRegister(L3GD20_REG_R_STATUS_REG,
L3GD20_MASK_STATUS_REG_ZDA);
            yda = this.readFromRegister(L3GD20_REG_R_STATUS_REG,
L3GD20_MASK_STATUS_REG_YDA);
            xda = this.readFromRegister(L3GD20_REG_R_STATUS_REG,
L3GD20_MASK_STATUS_REG_XDA);
        }
        return new int[]{xda, yda, zda};
    }
////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////
```

```
//  Gyroscope get methods x,y,z. tuned for offset gained by the calibration.

    /**
     * Get the calculated degree/sec value of the gyro x axis, offset
subtracted
     *
     * @return Get the calculated degree/sec value of the gyro x axis, offset
     * subtracted
     * @throws Exception throws exception if failure to retrieve value
     */
    public double getCalOutXValue() throws Exception {
        double calX = 0d;
        double x = this.getRawOutXValue();
        if (x >= this.minX && x <= this.maxX) {
            calX = 0d;
        } else {
            calX = x - this.meanX;
        }
        return calX;
    }

    /**
     * Get the calculated degree/sec value of the gyro y axis, offset
subtracted
     *
     * @return Get the calculated degree/sec value of the gyro y axis, offset
     * subtracted
     * @throws  Exception throws exception if failure to retrieve value
     */
    public double getCalOutYValue() throws Exception {
        double calY = 0d;
        double y = this.getRawOutYValue();
        if (y >= this.minY && y <= this.maxY) {
            calY = 0d;
        } else {
            calY = y - this.meanY;
        }
        return calY;
    }

    /**
```

```
     * Get the calculated degree/sec value of the gyro x axis, offset
subtracted
     *
     * @return return the calculated degree/sec value of the gyro z axis,
offset
     * subtracted
     * @throws Exception throws exception if fail to read values
     */
    public double getCalOutZValue() throws Exception {
        double calZ = 0d;
        double z = this.getRawOutZValue();
        if (z >= this.minZ && z <= this.maxZ) {
            calZ = 0d;
        } else {
            calZ = z - this.meanZ;
        }
        return calZ;
    }

}
```

```
/*
 * 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
 * sesnors. There will also be a video stream from the ROV.
 *
 * The system consists of a Raspberry Pi on the ROV that is connected to
several
 * Arduino microcontrollers that are connected to sensors and and programed
with
 * the control system.
 * The external computer which is on the surface vessel is connected to a GPS,
 * echo sounder and the RBPi. It will present and save data in addition to
 * handle user commands.
 */


import ExternalCommunication.StorageBox;
import com.fuzzylite.*;
import com.fuzzylite.activation.General;
import com.fuzzylite.defuzzifier.WeightedAverage;
import com.fuzzylite.rule.Rule;
import com.fuzzylite.rule.RuleBlock;
import com.fuzzylite.term.Constant;
import com.fuzzylite.term.Trapezoid;
import com.fuzzylite.term.Triangle;
import com.fuzzylite.variable.InputVariable;
import com.fuzzylite.variable.OutputVariable;
import java.nio.ByteBuffer;

/**
 * Class to represent Fuzzy expert system of Sugeno type This class is used to
 * generate gain for the PID-controller based on the speed of the ROV.
 * Constructor
 *
 * @author Kristian Homdrom
 */
public class FuzzyLiteController implements Runnable {

    private static volatile InputVariable speedInput;
    private static volatile OutputVariable KpGain;
```

```
    private static volatile OutputVariable KdGain;
    private static volatile OutputVariable KiGain;
    private Engine engine;
    private StorageBox sb;
    private static boolean b = false;

    /**
     * Constructor for the fuzzyController Create the main frame of the fuzzy
     * controller inference engine. This is an empty shell that wioll be
filled
     * by standard fuzzy set, rules etc.
     *
     * @param sb reference to storagebox
     */
    public FuzzyLiteController(StorageBox sb) {
        this.sb = sb;
        engine = new Engine();
        engine.setName("fuzzyController");
        engine.setDescription("");

    }

    public void setParamsEngine() {
        setInputVariableParam();
        getSetKpGain();
        getSetKiGain();
        getSetKdGain();
        setKpRuleBlock();
        setKiRuleBlock();
        setKdRuleBlock();
        b = true;
        sb.setAvailable(false);
    }

    /**
     * This method sets the input parameters for the fuzzyController This will
     * be the range of the input, what membership function to be used The
     * memberships functions label and range. Then adds the input to the
     * inference engine
     */
    public void setInputVariableParam() {
```

```
        speedInput = new InputVariable();
        speedInput.setName("speedInput");
        speedInput.setDescription("");
        speedInput.setEnabled(true);
        speedInput.setRange(0.000, 8.000);
        speedInput.setLockValueInRange(false);
        speedInput.addTerm(new Trapezoid("slow", 0.000, 0.000, 2.000, 4.000));
        speedInput.addTerm(new Triangle("medium", 2.000, 4.000, 6.000));
        speedInput.addTerm(new Trapezoid("fast", 4.000, 6.000, 8.000, 8.000));
        engine.addInputVariable(speedInput);
    }
  /**
   * retrieve kp gain from storagebox, set by client
   *  -
   */
    public void getSetKpGain() {

        ByteBuffer bb = sb.getKpGain();
        double low = bb.getDouble(0);
        double medium = bb.getDouble(8);
        double high = bb.getDouble(16);

        KpGain(low, medium, high);
    }
  /**
   *  retrieve ki gain from storagebox, set by client-
   * Set the kigain of fuzzy system
   */
    public void getSetKiGain() {

        ByteBuffer bb = sb.getKiGain();
        double low = bb.getDouble(0);
        double medium = bb.getDouble(8);
        double high = bb.getDouble(16);

        KiGain(low, medium, high);
    }
/**
   *  retrieve kd gain from storagebox, set by client-
   * Set the kdgain of fuzzy system
   */
```

```
    public void getSetKdGain() {
        ByteBuffer bb = sb.getKdGain();
        double low = bb.getDouble(0);
        double medium = bb.getDouble(8);
        double high = bb.getDouble(16);

        KdGain(low, medium, high);
    }

    /**
     * This method sets the output paramters for the fuzzyController This
     * includes the name, range, inference method ( Sugeno is chosen),label
and
     * value of singletons at 1 Then adds these outputparamters to the
inference
     * engine
     *
     * @param low low value gain kp
     * @param medium medium value gain kp
     * @param high high value gain kp
     */
    public void KpGain(double low, double medium, double high) {
        if (low == 0) {
            low = 0.2;
            medium = 0.5;
            high = 1.0;
        }

        KpGain = new OutputVariable();
        KpGain.setName("KpGain");
        KpGain.setDescription("proportional gain of controller ");
        KpGain.setEnabled(true);
        KpGain.setRange(0.0000, 1.000);
        KpGain.setLockValueInRange(false);
        KpGain.setAggregation(null);
        KpGain.setDefuzzifier(new WeightedAverage("TakagiSugeno"));
        KpGain.setDefaultValue(Double.NaN);
        KpGain.setLockPreviousValue(false);
        //The gain will be set as constants, leading to no values going over
the limit
```

```
        KpGain.addTerm(new Constant("Low", low));
        KpGain.addTerm(new Constant("Medium", medium));
        KpGain.addTerm(new Constant("High", high));
        engine.addOutputVariable(KpGain);
    }

    /**
     * This method sets the output paramters for the fuzzyController This
     * includes the name, range, inference method ( Sugeno is chosen),label
and
     * value of singletons at 1 Then adds these outputparamters to the
inference
     * engine
     * @param low low value gain kd
     * @param medium medium value gain kd
     * @param high high value gain kd
     */
    public void KdGain(double low, double medium, double high) {
        if (low == 0) {
            low = 0.2401;
            medium = 0.2401;
            high = 0.2401;
        }
        KdGain = new OutputVariable();
        KdGain.setName("KdGain");
        KdGain.setDescription("Derivat gain of controller");
        KdGain.setEnabled(true);
        KdGain.setRange(0.0000, 0.5000);
        KdGain.setLockValueInRange(false);
        KdGain.setAggregation(null);
        KdGain.setDefuzzifier(new WeightedAverage("TakagiSugeno"));
        KdGain.setDefaultValue(Double.NaN);
        KdGain.setLockPreviousValue(false);
        //The gain will be set as constants, leading to no values going over
the limit
        KdGain.addTerm(new Constant("Low", low));
        KdGain.addTerm(new Constant("Medium", medium));
        KdGain.addTerm(new Constant("High", high));
        engine.addOutputVariable(KdGain);
    }
```

```
    /**
     * This method sets the output paramters for the fuzzyController This
     * includes the name, range, inference method ( Sugeno is chosen),label
and
     * value of singletons at 1 Then adds these outputparamters to the
inference
     * engine
     * @param low low value gain ki
     * @param medium medium value gain ki
     * @param high high value gain ki
     */
    public void KiGain(double low, double medium, double high) {
        if (low == 0) {
            low = 0.0127;
            medium = 0.0127;
            high = 0.0127;
        }
        KiGain = new OutputVariable();
        KiGain.setName("KiGain");
        KiGain.setDescription("Integral gain of  controller");
        KiGain.setEnabled(true);
        KiGain.setRange(0.0000, 0.5000);
        KiGain.setLockValueInRange(false);
        KiGain.setAggregation(null);
        KiGain.setDefuzzifier(new WeightedAverage("TakagiSugeno"));
        KiGain.setDefaultValue(Double.NaN);
        KiGain.setLockPreviousValue(false);
        //The gain will be set as constants, leading to no values going over
the limit
        KiGain.addTerm(new Constant("Low", low));
        KiGain.addTerm(new Constant("Medium", medium));
        KiGain.addTerm(new Constant("High", high));

        engine.addOutputVariable(KiGain);

    }

    /**
     * This method adds the ruleblock to the inference engine Parameters are
     * name of ruleblock, ruleconsequent and then adds the ruleblock to the
     * inference engine
```

```
     */
    public void setKpRuleBlock() {
        RuleBlock ruleBlock = new RuleBlock();
        ruleBlock.setName("Kp");
        ruleBlock.setDescription("RuleBlock for proportinal gain");
        ruleBlock.setEnabled(true);
        ruleBlock.setConjunction(null);
        ruleBlock.setDisjunction(null);
        ruleBlock.setImplication(null);
        ruleBlock.setActivation(new General());
        // Set the rulebase, antecedent, and consequent have to match params
set in setname
        ruleBlock.addRule(Rule.parse("if speedInput is slow then KpGain is
High", engine));
        ruleBlock.addRule(Rule.parse("if speedInput is medium then KpGain is
Medium", engine));
        ruleBlock.addRule(Rule.parse("if speedInput is fast then KpGain is
Low", engine));
        engine.addRuleBlock(ruleBlock);
    }

    /**
     * This method adds the ruleblock to the inference engine Parameters are
     * name of ruleblock, ruleconsequent and then adds the ruleblock to the
     * inference engine
     */
    public void setKiRuleBlock() {
        RuleBlock ruleBlock = new RuleBlock();
        ruleBlock.setName("Ki");
        ruleBlock.setDescription("RuleBlock for integral gain");
        ruleBlock.setEnabled(true);
        ruleBlock.setConjunction(null);
        ruleBlock.setDisjunction(null);
        ruleBlock.setImplication(null);
        ruleBlock.setActivation(new General());
        // Set the rulebase, antecedent, and consequent have to match params
set in setname
        ruleBlock.addRule(Rule.parse("if speedInput is slow then KiGain is
High", engine));
        ruleBlock.addRule(Rule.parse("if speedInput is medium then KiGain is
Medium", engine));
```

```
        ruleBlock.addRule(Rule.parse("if speedInput is fast then KiGain is
Low", engine));
        engine.addRuleBlock(ruleBlock);
    }

    /**
     * This method adds ruleblock to the inference engine Parameters are name
of
     * ruleblock, ruleconsequent and then adds the ruleblock to the inference
     * engine
     */
    public void setKdRuleBlock() {
        RuleBlock ruleBlock = new RuleBlock();
        ruleBlock.setName("Kd");
        ruleBlock.setDescription("RuleBlock for derivat gain");
        ruleBlock.setEnabled(true);
        ruleBlock.setConjunction(null);
        ruleBlock.setDisjunction(null);
        ruleBlock.setImplication(null);
        ruleBlock.setActivation(new General());
        // Set the rulebase, antecedent, and consequent have to match params
set in setname
        ruleBlock.addRule(Rule.parse("if speedInput is slow then KdGain is
High", engine));
        ruleBlock.addRule(Rule.parse("if speedInput is medium then KdGain is
Medium", engine));
        ruleBlock.addRule(Rule.parse("if speedInput is fast then KdGain is
Low", engine));
        engine.addRuleBlock(ruleBlock);
    }

    /**
     * Check if the infernce engine is ready for duty
     *
     * @return state of the engine
     */
    public boolean checkIfEngineReady() {
        StringBuilder status = new StringBuilder();
        if (!engine.isReady(status)) {
            throw new RuntimeException("[engine error] engine is not ready:n"
+ status);
```

```
        }
        return engine.isReady(status);
    }

    /**
     * Set the speedInput to the fuzzy inference engine to the parameter input
     * received from the speed of the ROv
     *
     * @param speed Speed is the param recveived from the ROV vessel on which
to
     * set the fuzzy inference engine
     */
    public void setSpeed(double speed) {
        this.speedInput.setValue(speed);
    }

    /**
     * Set the parameters of the gain values
     *
     * @param kp kp gain value of the fuzzy expert system
     * @param ki ki gain value of the fuzzy expert system
     * @param kd kd gain value of the fuzzy expert system
     */
    public void setKGain(double kp, double ki, double kd) {

        sb.setGains((float) kp, (float) ki, (float) kd);

    }

    @Override
    public void run() {
        if (b) {

            // Send the input value into the fuzzy expert system
            setSpeed(sb.getSpeed());
            // Run the fuzzy expert system based on the input
            engine.process();
            // extract new output gain from the process.
            setKGain(engine.getOutputValue("KpGain"),
engine.getOutputValue("KiGain"), engine.getOutputValue("KdGain"));
```

```
        }
        if (sb.getAvailable()) {

            setParamsEngine();
        }
    }
}
```

```
/*
 * 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
 * sesnors. There will also be a video stream from the ROV.
 *
 * The system consists of a Raspberry Pi on the ROV that is connected to
several
 * Arduino microcontrollers that are connected to sensors and and programed
with
 * the control system.
 * The external computer which is on the surface vessel is connected to a GPS,
 * echo sounder and the RBPi. It will present and save data in addition to
 * handle user commands.
 */


import com.pi4j.io.i2c.I2CDevice;

import java.io.IOException;

public class EndianReaders {

public enum Endianness {
LITTLE_ENDIAN,
BIG_ENDIAN
}

/**
 * Read an unsigned byte from the I2C device
     * @param device param of what i2cDevice
     * @param i2caddr i2cAddress of i2c device
     * @param reg register of i2cdevice to read
     * @param verbose status if read has been completed
     * @return  return valuie read from device's register
     * @throws java.io.IOException if fail to read throw exception
 */
public static int readU8(I2CDevice device, int i2caddr, int reg, boolean
verbose) throws IOException {
int result = 0;
```

```
result = device.read(reg);
if (verbose) {
System.out.println("I2C: Device " + i2caddr + " (0x" +
Integer.toHexString(i2caddr) +
") returned " + result + " (0x" + Integer.toHexString(result) +
") from reg " + reg + " (0x" + Integer.toHexString(reg) + ")");
}

return result; // & 0xFF;
}

/**
 * Read a signed byte from the I2C device
     * @param device param of what i2cDevice
     * @param i2caddr i2cAddress of i2c device
     * @param reg register of i2cdevice to read
     * @param verbose status if read has been completed
     * @return  return valuie read from device's register
     * @throws java.io.IOException if fail to read throw exception
 */
public static int readS8(I2CDevice device, int i2caddr, int reg, boolean
verbose) throws IOException {
int result = 0;

result = device.read(reg); // & 0x7F;
if (result > 127)
result -= 256;
if (verbose) {
System.out.println("I2C: Device " + i2caddr + " returned " + result + " from
reg " + reg);
}
return result; // & 0xFF;
}

public static int readU16LE(I2CDevice device, int i2caddr, int register,
boolean verbose) throws IOException {
return EndianReaders.readU16(device, i2caddr, register,
Endianness.LITTLE_ENDIAN, verbose);
}

public static int readU16BE(I2CDevice device, int i2caddr, int register,
```

```
boolean verbose) throws IOException {
return EndianReaders.readU16(device, i2caddr, register, Endianness.BIG_ENDIAN,
verbose);
}

public static int readU16(I2CDevice device, int i2caddr, int register,
Endianness endianness, boolean verbose) throws IOException {
int hi = EndianReaders.readU8(device, i2caddr, register, verbose);
int lo = EndianReaders.readU8(device, i2caddr, register + 1, verbose);
return ((endianness == Endianness.BIG_ENDIAN) ? (hi << 8) + lo : (lo << 8) +
hi); // & 0xFFFF;
}

public static int readS16(I2CDevice device, int i2caddr, int register,
Endianness endianness, boolean verbose) throws IOException {
int hi = 0, lo = 0;
if (endianness == Endianness.BIG_ENDIAN) {
hi = EndianReaders.readS8(device, i2caddr, register, verbose);
lo = EndianReaders.readU8(device, i2caddr, register + 1, verbose);
} else {
lo = EndianReaders.readU8(device, i2caddr, register, verbose);
hi = EndianReaders.readS8(device, i2caddr, register + 1, verbose);
}
return ((hi << 8) + lo); // & 0xFFFF;
}

public static int readS16LE(I2CDevice device, int i2caddr, int register,
boolean verbose) throws IOException {
return EndianReaders.readS16(device, i2caddr, register,
Endianness.LITTLE_ENDIAN, verbose);
}

public static int readS16BE(I2CDevice device, int i2caddr, int register,
boolean verbose) throws IOException {
return EndianReaders.readS16(device, i2caddr, register, Endianness.BIG_ENDIAN,
verbose);
}
}
```

```
/*
 * 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
 * sesnors. There will also be a video stream from the ROV.
 *
 * The system consists of a Raspberry Pi on the ROV that is connected to
several
 * Arduino microcontrollers that are connected to sensors and and programed
with
 * the control system.
 * The external computer which is on the surface vessel is connected to a GPS,
 * echo sounder and the RBPi. It will present and save data in addition to
 * handle user commands.
 */


import java.io.IOException;
import java.net.ServerSocket;
import java.net.Socket;
import java.util.concurrent.Executor;
import java.util.concurrent.Executors;

/**
 * This class is responsible for the TCP connections. It listens to the TCP
port
 * and when an TCP request from a client is received, this class will creat a
 * clientHandler and give it a thread.
 *
 * @author MorSol
 */
public class ConnectionServer implements Runnable {

    private static final int LISTENING_PORT = 5001; // Port for TCP-connection
    private StorageBox sb;
    private Executor service;
    private ServerSocket serverSocket;


    /**
     * Constructer for the Connection server class, takes in executor from
```

```
main
     * class.
     * @param sb link to storagebox
     */
    public ConnectionServer(StorageBox sb)  {
        try {
                    this.sb = sb;
        service = Executors.newCachedThreadPool();
        serverSocket = new ServerSocket(LISTENING_PORT);
        } catch (Exception e) {
            System.out.println(e.getMessage());
        }


    }

    /**
     * Run methode for this thread. Is executed when running this thread. In
     * this method the listening socket is created. Then a while loop is
     * entered, this is where new connections are accepted and given it's own
     * clientHandler thread.
     */
    @Override
    public void run() {
        // Open socket on server for listening to inncomming connectios.

        try {

             Socket socket = serverSocket.accept();
            service.execute(new ClientHandler(socket, sb));

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

    }

}
```

```
/*
 * 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
 * sesnors. There will also be a video stream from the ROV.
 *
 * The system consists of a Raspberry Pi on the ROV that is connected to
several
 * Arduino microcontrollers that are connected to sensors and and programed
with
 * the control system.
 * The external computer which is on the surface vessel is connected to a GPS,
 * echo sounder and the RBPi. It will present and save data in addition to
 * handle user commands.
 */


import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.OutputStreamWriter;
import java.io.PrintWriter;
import java.net.InetAddress;
import java.net.Socket;
import java.net.UnknownHostException;
import java.nio.ByteBuffer;
import java.util.logging.Level;
import java.util.logging.Logger;

/**
 * This class is responsible for handeling TCP-clientsockets. It takes in the
 * socket of the connected client then reads incomming messages and write
 * outgoing commands. In the event of communication loss, the socket should
 * close and the thread should terminate.
 *
 * @author MorSol
 */
public class ClientHandler implements Runnable {

    Socket socket;                  // Client connection socket.
    boolean runClientHandler;       // True if clientHandler shall run.
```

```
    BufferedReader in = null;       // Input reader.
    PrintWriter out = null;         // Output writer.
    private StorageBox sb;
    private long timer;
    private boolean localhost = false;
    private Parser parser;

    /**
     * Constructer for the clientHandler class which functions as the
connection
     * to the client. Takes in the connection socket from the connectionSrver.
     *
     * @param socket Socket from incoming connection to communicate with
client
     * @param sb Storagebox as param for storing/retrieving data
     */
    public ClientHandler(Socket socket, StorageBox sb) {
        this.sb = sb;
        System.out.println("Creating clientHandler");
        //   this.sc = clientSocket;
        //   this.sc=sc;
        this.socket = socket;     // Set socket to socket received from
connectionServer
        timer = System.currentTimeMillis();
        runClientHandler = true;        // Set run to true
        parser = new Parser();
        try {
            // Creating inputstream.
            in = new BufferedReader(new
InputStreamReader(socket.getInputStream()));
            // Creating outputstream.
            out = new PrintWriter(new
OutputStreamWriter(socket.getOutputStream()));
        } catch (IOException ex) {
            System.out.println("Could not create BufferedReader and/or
PrintWriter");
        }
    }

    /**
     * Run methode for this thread. Executed when the thread is running. As
```

```
long
     * as a client is connected to the socket the while loop will run and look
     * for incomming String messages. If a message is recived it will be
     * processed. In the event of the client shuting down, loss of connection
or
     * reciving "quit", the server will terminate the connection and end the
     * tread.
     */
    @Override
    public void run() {

        try {
            System.out.println("Accept client address: " +
socket.getInetAddress().
                    getCanonicalHostName());
            if (socket.getInetAddress() == InetAddress.getLocalHost()) {
                localhost = true;

            } else {

            }
            // Runs while loop as long as runClientHandler = true
            while (runClientHandler && localhost == false) {

                if (!socket.isConnected()) {
                    runClientHandler = false;
                    break;
                }
                // Read the command over fro mthe TCP-connection
                readCommand();
                if (System.currentTimeMillis() > timer + 100) {
                    sendData();
                    timer = System.currentTimeMillis();
                }
            }
            // Method to break run method
            while (runClientHandler && localhost) {
                if (!socket.isConnected()) {
                    runClientHandler = false;
                    break;
                }
```

```
                // Read the command over fro mthe TCP-connection
                readCommand();
            }
        } catch (UnknownHostException e) {
            System.out.println(e.getMessage());
            Logger.getLogger(ClientHandler.class.getName()).log(Level.SEVERE,
null, e);
        } finally {
            try {
                // Closing the input/output -stream and the client socket
                System.out.println("Closing client socket for: " +
socket.getInetAddress());
                in.close();         // Close innputstream
                out.close();        // Close outputstream
                socket.close();     // Close client socket
            } catch (IOException ex) {
                System.out.println(ex.getMessage());
            }
        }
    }

    /**
     * Send byteArray of informatiom regarding ROV to the external GUI over
TCP
     * connection
     */
    public void sendData() {
        // Retrieve the ByteByuffer of information from StorageBox
        ByteBuffer bb = sb.getGuiArray();
        // Create a byte array to send over TCP-connection
        byte[] b = new byte[42];
        // Place data from Bytebuffer into bytearray
        for (int i = 0; i < sb.getGuiArray().capacity() - 1; i++) {

            b[i] = bb.get(i);
        }
        // Try to send the byte array to Remote host over socket
        try {
            socket.getOutputStream().write(b);
        } catch (IOException ex) {
            Logger.getLogger(ClientHandler.class.getName()).log(Level.SEVERE,
```

```
null, ex);
        }
    }

    /**
     * Read the incoming message from the TCP conenction, then send them to
     * Parser to be evaluated
     */
    public void readCommand() {
        try {
            if (socket.getInputStream().available() > 0) {
                String clientInput = in.readLine();
                System.out.println(socket.getInetAddress() + " - " +
clientInput);

                parser.parseCommand(clientInput, sb);
                if (clientInput.equalsIgnoreCase("quit")) {

                    // Stops thread is "quit" is received on the input stream.
                    System.out.println("Stopping thread for client: "
                            + socket.getInetAddress());
                    runClientHandler = false;
                }
                // Sets the runClientHander to false if a massage containing
null
                // is received on the iput. This happens when the client
closes it's
                // connection. This results the condition for the while loop
is set
                // to false, thus ending the while loop.
                if (clientInput == null) {
                    System.out.println("Null received, looks like the client
is offline");
                    runClientHandler = false;
                }
            }
        } catch (IOException ex) {
            System.out.println(ex.getMessage());
        }
    }
```

```
    /**
     * Sends strings from the socket to the connected client.
     *
     * @param stringCommand, String
     */
    public void sendCommand(String stringCommand) {
        out.write(stringCommand);
        out.flush();
    }

}
```

```
/*
 * 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
 * sesnors. There will also be a video stream from the ROV.
 *
 * The system consists of a Raspberry Pi on the ROV that is connected to
several
 * Arduino microcontrollers that are connected to sensors and and programed
with
 * the control system.
 * The external computer which is on the surface vessel is connected to a GPS,
 * echo sounder and the RBPi. It will present and save data in addition to
 * handle user commands.
 */


public class BitOps {
public static boolean checkBit(int value, int position) {
int mask = 1 << position;
return ((value & mask) == mask);
}

public static int setBit(int value, int position) {
return (value | (1 << position));
}

public static int clearBit(int value, int position) {
return (value & ~(1 << position));
}

public static int flipBit(int value, int position) {
return (value ^ (1 << position));
}

public static boolean checkBits(int value, int mask) {
return ((value & mask) == mask);
}

public static int setBits(int value, int mask) {
return (value | mask);
```

```
}

public static int clearBits(int value, int mask) {
return (value & (~mask));
}

public static int flipBits(int value, int mask) {
return value ^ mask;
}

public static int setValueUnderMask(int valueToSet, int currentValue, int
mask) {
int currentValueCleared = clearBits(currentValue, mask);
int i = 0;
while (mask % 2 == 0 && mask != 0x00) {
mask >>= 1;
i++;
}
return setBits(valueToSet << i, currentValueCleared);
}

public static int getValueUnderMask(int currentValue, int mask) {
int currentValueCleared = clearBits(currentValue, ~mask); // clear bits not
under mask
int i = 0;
while (mask % 2 == 0 && mask != 0x00) {
mask >>= 1;
i++;
}
return currentValueCleared >> i;
}

public static int twosComplementToByte(int value) {
if (value >= 0 && value <= 0x7f)
return value;
else
return value - 0x100;
}

public static int twosComplementToCustom(int value, int signBitPosition) {
if (value >= 0 && value <= (1 << signBitPosition) - 1)
```

```
return value;
else
return value - (2 << signBitPosition);
}
}
```

I.4 Java Client Application Code

```
/*
 * To change this license header, choose License Headers in Project
Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package ntnusubsea.gui;

import java.awt.image.BufferedImage;
import java.util.concurrent.Executors;
import java.util.concurrent.ScheduledExecutorService;
import java.util.concurrent.TimeUnit;
import javax.swing.SwingUtilities;

/**
 * Main class that launches the application and schedules
 * the different threads
 *
 * @author Marius Nonsvik
 */
public class NTNUSubseaGUI {

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        Data data = new Data();
        EchoSounderFrame sonar = new EchoSounderFrame(data);
        DataLogger logger = new DataLogger(data);
        TCPClient client = new TCPClient(data);
        IOControlFrame io = new IOControlFrame(data, client);
        ROVFrame frame = new ROVFrame(sonar, data, client, io);
        VideoEncoder encoder = new VideoEncoder(data);
        NmeaReceiver nmea = new NmeaReceiver(data);
        UDPClient stream = new UDPClient(data);
        BufferedImage banan;
        ScheduledExecutorService executor =
        Executors.newScheduledThreadPool(8);
        SwingUtilities.invokeLater(frame);
        SwingUtilities.invokeLater(sonar);
        SwingUtilities.invokeLater(io);
```

```
        sonar.setVisible(false);
        data.addObserver(sonar);
        data.addObserver(logger);
        data.addObserver(frame);
        data.addObserver(encoder);
        data.addObserver(io);
        executor.scheduleAtFixedRate(logger,
                3000, 1000, TimeUnit.MILLISECONDS);
        executor.scheduleAtFixedRate(encoder,
                20, 40, TimeUnit.MILLISECONDS);
        executor.scheduleAtFixedRate(nmea,
                0, 1000, TimeUnit.MILLISECONDS);
        executor.scheduleAtFixedRate(client,
                0, 100, TimeUnit.MILLISECONDS);
        executor.scheduleAtFixedRate(stream,
                0, 20, TimeUnit.MILLISECONDS);
        Runtime.getRuntime()
                .addShutdownHook(new Thread(new Runnable() {
                    @Override
                    public void run() {
                        executor.shutdown();
                        encoder.finishVideo();
                    }
                },
                        "Shutdown-thread"));
    }
}
```

```
/*
 * To change this license header, choose License Headers in Project
Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package ntnusubsea.gui;

import java.awt.Graphics;
import java.awt.Graphics2D;
import java.awt.GraphicsConfiguration;
import java.awt.GraphicsDevice;
import java.awt.GraphicsEnvironment;
import java.awt.HeadlessException;
import java.awt.Image;
import java.awt.RenderingHints;
import java.awt.Transparency;
import java.awt.image.BufferedImage;
import java.awt.image.ColorModel;
import java.awt.image.PixelGrabber;
import javax.swing.ImageIcon;

/**
 * This class contains utilities to change and handle images.
 * @author Marius Nonsvik
 */
public class ImageUtils {

    /**
     * Rotates an image the given amount of degrees
     * @param img Image to rotate
     * @param degree Number of degrees to rotate
     * @return The rotated image
     */
    public static Image rotateImage(Image img,double degree){
        BufferedImage bufImg = toBufferedImage(img);
        double angle = Math.toRadians(degree);
        return tilt(bufImg,angle);
    }

    /**
```

```
     * Tilts an image in a given angle
     * @param image Image to tilt
     * @param angle Angle to tilt the image
     * @return The tilted image
     */
    public static BufferedImage tilt(BufferedImage image, double angle) {
        double sin = Math.abs(Math.sin(angle)), cos =
Math.abs(Math.cos(angle));
        int w = image.getWidth(), h = image.getHeight();
        int neww = (int)Math.floor(w*cos+h*sin), newh =
(int)Math.floor(h*cos+w*sin);
        GraphicsConfiguration gc = getDefaultConfiguration();
        BufferedImage result = gc.createCompatibleImage(neww, newh,
Transparency.TRANSLUCENT);
        Graphics2D g = result.createGraphics();
       // g.translate((neww-w)/2, (newh-h)/2);
        g.rotate(angle, w/2, h/2);
        g.drawRenderedImage(image, null);
        g.dispose();
        return result;
    }

    /**
     * Returns the default configuration of the graphics device
     * @return Default configuration of the graphics device
     */
    public static GraphicsConfiguration getDefaultConfiguration() {
        GraphicsEnvironment ge =
GraphicsEnvironment.getLocalGraphicsEnvironment();
        GraphicsDevice gd = ge.getDefaultScreenDevice();
        return gd.getDefaultConfiguration();
    }

    /**
     * Converts an image to a bufferedImage
     * @param image Image to convert
     * @return The corresponding bufferedImage
     */
    public static BufferedImage toBufferedImage(Image image) {
        if (image instanceof BufferedImage) {
            return (BufferedImage)image;
```

```
        }

        // This code ensures that all the pixels in the image are loaded
        image = new ImageIcon(image).getImage();

        // Determine if the image has transparent pixels; for this method's
        // implementation, see e661 Determining If an Image Has Transparent
Pixels
        boolean hasAlpha = hasAlpha(image);

        // Create a buffered image with a format that's compatible with the
screen
        BufferedImage bimage = null;
        GraphicsEnvironment ge =
GraphicsEnvironment.getLocalGraphicsEnvironment();
        try {
            // Determine the type of transparency of the new buffered image
            int transparency = Transparency.OPAQUE;
            if (hasAlpha) {
                transparency = Transparency.BITMASK;
            }

            // Create the buffered image
            GraphicsDevice gs = ge.getDefaultScreenDevice();
            GraphicsConfiguration gc = gs.getDefaultConfiguration();
            bimage = gc.createCompatibleImage(
                image.getWidth(null), image.getHeight(null), transparency);
        } catch (HeadlessException e) {
            // The system does not have a screen
        }

        if (bimage == null) {
            // Create a buffered image using the default color model
            int type = BufferedImage.TYPE_INT_RGB;
            if (hasAlpha) {
                type = BufferedImage.TYPE_INT_ARGB;
            }
            bimage = new BufferedImage(image.getWidth(null),
image.getHeight(null), type);
        }
```

```
        // Copy image to buffered image
        Graphics g = bimage.createGraphics();

        // Paint the image onto the buffered image
        g.drawImage(image, 0, 0, null);
        g.dispose();

        return bimage;
    }

    // http://www.exampledepot.com/egs/java.awt.image/HasAlpha.html
    // This method returns true if the specified image has transparent pixels
    public static boolean hasAlpha(Image image) {
        // If buffered image, the color model is readily available
        if (image instanceof BufferedImage) {
            BufferedImage bimage = (BufferedImage)image;
            return bimage.getColorModel().hasAlpha();
        }

        // Use a pixel grabber to retrieve the image's color model;
        // grabbing a single pixel is usually sufficient
         PixelGrabber pg = new PixelGrabber(image, 0, 0, 1, 1, false);
        try {
            pg.grabPixels();
        } catch (InterruptedException e) {
        }

        // Get the image's color model
        ColorModel cm = pg.getColorModel();
        return cm.hasAlpha();
    }


        /**
     * Resizes a BufferedImage. Used to make an image fit it's container.
     *
     * @param image BufferedImage to resize
     * @param width Desired width of the image
     * @param height Desired height of the image
     * @return The resulting BufferedImage in the new size
     */
        public static BufferedImage resize(BufferedImage image, int width, int
```

```
height) {
        BufferedImage bi = new BufferedImage(width, height,
BufferedImage.TRANSLUCENT);
        Graphics2D g2d = (Graphics2D) bi.createGraphics();
        g2d.addRenderingHints(new RenderingHints(RenderingHints.KEY_RENDERING,
RenderingHints.VALUE_RENDER_QUALITY));
        g2d.drawImage(image, 0, 0, width, height, null);
        g2d.dispose();
        return bi;
    }
}
```

```
/*
 * To change this license header, choose License Headers in Project
Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package ntnusubsea.gui;

import java.awt.Graphics;
import java.awt.image.BufferedImage;
import javax.swing.JPanel;

/**
 * This class is an extended version of JPanel, with the added methods
required
 * to display BufferedImages in the panel.
 *
 * @author Marius Nonsvik
 */
public class ImagePanel extends JPanel {

    private BufferedImage image;
    private int width, height;

    /**
     * Constructor used to create the Sheet object
     *
     * @param width The width of the sheet
     * @param height The height of the sheet
     */
    public ImagePanel(int width, int height) {
        setSize(width, height);
    }

    /**
     * This methods updates the sheet to display the image used as a input
     * parameter
     *
     * @param img Image to display in the component
     */
    public void paintSheet(BufferedImage img) {
```

```
        image = null;
        image = img;
        repaint();
    }

    /**
     * Uses the the paintComponent method of the super class and makes the
     * component compatible with bufferedImage
     *
     * @param g A graphics context onto which a bufferedImage can be drawn
     */
    @Override
    protected void paintComponent(Graphics g) {
        super.paintComponent(g);
        g.drawImage(image, 0, 0, this);
    }
}
```

```
/*
 * To change this license header, choose License Headers in Project
Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
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;

/**
 * This frame lets the user control the different inputs and outputs. Sends a
 * new command whenever a button is pressed.
 * @author Marius Nonsvik
 */
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.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();
        jPanelChannel7 = new javax.swing.JPanel();
        jLabelChannel7Header = new javax.swing.JLabel();
        jToggleButtonChannel7 = new javax.swing.JToggleButton();
        jLabelChannel7Value = new javax.swing.JLabel();
        jLabelOnChannel7 = new javax.swing.JLabel();
        jLabelOffChannel7 = new javax.swing.JLabel();
        jLabelIndicatorChannel7 = new javax.swing.JLabel();
        jPanelChannel8 = new javax.swing.JPanel();
        jLabelChannel8Header = new javax.swing.JLabel();
```

```
        jToggleButtonChannel8 = new javax.swing.JToggleButton();
        jLabelChannel8Value = new javax.swing.JLabel();
        jLabelOnChannel8 = new javax.swing.JLabel();
        jLabelOffChannel8 = new javax.swing.JLabel();
        jLabelIndicatorChannel8 = new javax.swing.JLabel();
        jPanelChannel1 = new javax.swing.JPanel();
        jLabelChannel1Header = new javax.swing.JLabel();
        jLabelChannel1Value = new javax.swing.JLabel();
        jToggleButtonChannel1 = new javax.swing.JToggleButton();
        jLabelIndicatorChannel1 = new javax.swing.JLabel();
        jLabelOnChannel1 = new javax.swing.JLabel();
        jLabelOffChannel1 = new javax.swing.JLabel();
        jPanelChannel2 = new javax.swing.JPanel();
        jLabelChannel2Header = new javax.swing.JLabel();
        jLabelChannel2Value = new javax.swing.JLabel();
        jToggleButtonChannel2 = new javax.swing.JToggleButton();
        jLabelIndicatorChannel2 = new javax.swing.JLabel();
        jLabelOnChannel2 = new javax.swing.JLabel();
        jLabelOffChannel2 = new javax.swing.JLabel();
        jPanelChannel3 = new javax.swing.JPanel();
        jLabelChannel3Header = new javax.swing.JLabel();
        jLabelChannel3Value = new javax.swing.JLabel();
        jToggleButtonChannel3 = new javax.swing.JToggleButton();
        jLabelIndicatorChannel3 = new javax.swing.JLabel();
        jLabelOnChannel3 = new javax.swing.JLabel();
        jLabelOffChannel3 = new javax.swing.JLabel();
        jPanelChannel4 = new javax.swing.JPanel();
        jLabelChannel4Header = new javax.swing.JLabel();
        jLabelChannel4Value = new javax.swing.JLabel();
        jToggleButtonChannel4 = new javax.swing.JToggleButton();
        jLabelIndicatorChannel4 = new javax.swing.JLabel();
        jLabelOnChannel4 = new javax.swing.JLabel();
        jLabelOffChannel4 = new javax.swing.JLabel();
        jSeparator1 = new javax.swing.JSeparator();

        setDefaultCloseOperation(javax.swing.WindowConstants.DISPOSE_ON_CLOSE)
;
        setTitle("I/O Controller");
        setBackground(new java.awt.Color(28, 28, 28));
        setForeground(new java.awt.Color(28, 28, 28));
```

```
        jPanel9.setBackground(new java.awt.Color(28, 28, 28));

        jPanelChannel5.setBackground(new java.awt.Color(28, 28, 28));
        jPanelChannel5.setPreferredSize(new java.awt.Dimension(160, 120));

        jLabelChannel5Header.setBackground(new java.awt.Color(28, 28, 28));
        jLabelChannel5Header.setFont(new java.awt.Font("Tahoma", 0, 14)); //
NOI18N
        jLabelChannel5Header.setForeground(new java.awt.Color(255, 255, 255));
        jLabelChannel5Header.setText("Channel 5 (Output)");

        jToggleButtonChannel5.setBackground(new java.awt.Color(28, 28, 28));
        jToggleButtonChannel5.setForeground(new java.awt.Color(28, 28, 28));
        jToggleButtonChannel5.setIcon(new
javax.swing.ImageIcon(getClass().getResource("/ntnusubsea/gui/Images/Toggle_of
f.png"))); // NOI18N
        jToggleButtonChannel5.setBorder(null);
        jToggleButtonChannel5.setEnabled(false);
        jToggleButtonChannel5.setSelectedIcon(new
javax.swing.ImageIcon(getClass().getResource("/ntnusubsea/gui/Images/Toggle_on
.png"))); // NOI18N
        jToggleButtonChannel5.addActionListener(new
java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jToggleButtonChannel5ActionPerformed(evt);
            }
        });

        jLabelChannel5Value.setBackground(new java.awt.Color(28, 28, 28));
        jLabelChannel5Value.setFont(new java.awt.Font("Tahoma", 0, 14)); //
NOI18N
        jLabelChannel5Value.setForeground(new java.awt.Color(255, 255, 255));
        jLabelChannel5Value.setText("jLabel2");

        jLabelOnChannel5.setBackground(new java.awt.Color(28, 28, 28));
        jLabelOnChannel5.setForeground(new java.awt.Color(255, 255, 255));
        jLabelOnChannel5.setText("On");

        jLabelOffChannel5.setBackground(new java.awt.Color(28, 28, 28));
        jLabelOffChannel5.setForeground(new java.awt.Color(255, 255, 255));
        jLabelOffChannel5.setText("Off");
```

```
        jLabelIndicatorChannel5.setBackground(new java.awt.Color(28, 28, 28));
        jLabelIndicatorChannel5.setIcon(new
javax.swing.ImageIcon(getClass().getResource("/ntnusubsea/gui/Images/IO_Off.pn
g"))); // NOI18N
        jLabelIndicatorChannel5.setOpaque(true);

        javax.swing.GroupLayout jPanelChannel5Layout = new
javax.swing.GroupLayout(jPanelChannel5);
        jPanelChannel5.setLayout(jPanelChannel5Layout);
        jPanelChannel5Layout.setHorizontalGroup(
            jPanelChannel5Layout.createParallelGroup(javax.swing.GroupLayout.
Alignment.LEADING)
            .addComponent(jLabelChannel5Header,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)
            .addGroup(jPanelChannel5Layout.createSequentialGroup()
                .addComponent(jToggleButtonChannel5,
javax.swing.GroupLayout.PREFERRED_SIZE, 46,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UN
RELATED)
                .addComponent(jLabelIndicatorChannel5,
javax.swing.GroupLayout.PREFERRED_SIZE, 54,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RE
LATED)
                .addGroup(jPanelChannel5Layout.createParallelGroup(javax.swing
.GroupLayout.Alignment.LEADING)
                    .addComponent(jLabelOnChannel5,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)
                    .addComponent(jLabelOffChannel5,
javax.swing.GroupLayout.DEFAULT_SIZE, 50, Short.MAX_VALUE)))
            .addComponent(jLabelChannel5Value,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)
        );
        jPanelChannel5Layout.setVerticalGroup(
            jPanelChannel5Layout.createParallelGroup(javax.swing.GroupLayout.
Alignment.LEADING)
```

```
            .addGroup(jPanelChannel5Layout.createSequentialGroup()
                .addComponent(jLabelChannel5Header,
javax.swing.GroupLayout.PREFERRED_SIZE, 28,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RE
LATED)
                .addComponent(jLabelChannel5Value,
javax.swing.GroupLayout.PREFERRED_SIZE, 50,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RE
LATED)
                .addGroup(jPanelChannel5Layout.createParallelGroup(javax.swing
.GroupLayout.Alignment.LEADING)
                    .addGroup(jPanelChannel5Layout.createSequentialGroup()
                        .addGap(5, 5, 5)
                        .addComponent(jLabelOnChannel5,
javax.swing.GroupLayout.PREFERRED_SIZE, 20,
javax.swing.GroupLayout.PREFERRED_SIZE)
                        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlac
ement.RELATED)
                        .addComponent(jLabelOffChannel5,
javax.swing.GroupLayout.PREFERRED_SIZE, 20,
javax.swing.GroupLayout.PREFERRED_SIZE))
                    .addComponent(jToggleButtonChannel5,
javax.swing.GroupLayout.PREFERRED_SIZE, 63,
javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addComponent(jLabelIndicatorChannel5,
javax.swing.GroupLayout.PREFERRED_SIZE, 63,
javax.swing.GroupLayout.PREFERRED_SIZE))
                .addContainerGap())
        );

        jPanelChannel6.setBackground(new java.awt.Color(28, 28, 28));

        jLabelChannel6Header.setBackground(new java.awt.Color(28, 28, 28));
        jLabelChannel6Header.setFont(new java.awt.Font("Tahoma", 0, 14)); //
NOI18N
        jLabelChannel6Header.setForeground(new java.awt.Color(255, 255, 255));
        jLabelChannel6Header.setText("Channel 6 (Output)");

        jToggleButtonChannel6.setBackground(new java.awt.Color(28, 28, 28));
```

```
        jToggleButtonChannel6.setForeground(new java.awt.Color(28, 28, 28));
        jToggleButtonChannel6.setIcon(new
javax.swing.ImageIcon(getClass().getResource("/ntnusubsea/gui/Images/Toggle_of
f.png"))); // NOI18N
        jToggleButtonChannel6.setBorder(null);
        jToggleButtonChannel6.setEnabled(false);
        jToggleButtonChannel6.setSelectedIcon(new
javax.swing.ImageIcon(getClass().getResource("/ntnusubsea/gui/Images/Toggle_on
.png"))); // NOI18N
        jToggleButtonChannel6.addActionListener(new
java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jToggleButtonChannel6ActionPerformed(evt);
            }
        });

        jLabelChannel6Value.setBackground(new java.awt.Color(28, 28, 28));
        jLabelChannel6Value.setFont(new java.awt.Font("Tahoma", 0, 14)); //
NOI18N
        jLabelChannel6Value.setForeground(new java.awt.Color(255, 255, 255));
        jLabelChannel6Value.setText("jLabel2");

        jLabelOnChannel6.setBackground(new java.awt.Color(28, 28, 28));
        jLabelOnChannel6.setForeground(new java.awt.Color(255, 255, 255));
        jLabelOnChannel6.setText("On");

        jLabelOffChannel6.setBackground(new java.awt.Color(28, 28, 28));
        jLabelOffChannel6.setForeground(new java.awt.Color(255, 255, 255));
        jLabelOffChannel6.setText("Off");

        jLabelIndicatorChannel6.setBackground(new java.awt.Color(28, 28, 28));
        jLabelIndicatorChannel6.setIcon(new
javax.swing.ImageIcon(getClass().getResource("/ntnusubsea/gui/Images/IO_Off.pn
g"))); // NOI18N
        jLabelIndicatorChannel6.setText("jLabel26");
        jLabelIndicatorChannel6.setOpaque(true);

        javax.swing.GroupLayout jPanelChannel6Layout = new
javax.swing.GroupLayout(jPanelChannel6);
        jPanelChannel6.setLayout(jPanelChannel6Layout);
        jPanelChannel6Layout.setHorizontalGroup(
```

```
            jPanelChannel6Layout.createParallelGroup(javax.swing.GroupLayout.
Alignment.LEADING)
            .addComponent(jLabelChannel6Header,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)
            .addGroup(jPanelChannel6Layout.createSequentialGroup()
                .addComponent(jToggleButtonChannel6,
javax.swing.GroupLayout.PREFERRED_SIZE, 46,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UN
RELATED)
                .addComponent(jLabelIndicatorChannel6,
javax.swing.GroupLayout.PREFERRED_SIZE, 58,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RE
LATED)
                .addGroup(jPanelChannel6Layout.createParallelGroup(javax.swing
.GroupLayout.Alignment.LEADING)
                    .addComponent(jLabelOffChannel6,
javax.swing.GroupLayout.DEFAULT_SIZE, 50, 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.RE
LATED)
                .addComponent(jLabelChannel6Value,
javax.swing.GroupLayout.PREFERRED_SIZE, 50,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RE
LATED)
```

```
                .addGroup(jPanelChannel6Layout.createParallelGroup(javax.swing
.GroupLayout.Alignment.LEADING)
                    .addGroup(jPanelChannel6Layout.createParallelGroup(javax.s
wing.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.Component
Placement.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(28, 28, 28));

        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(28, 28, 28));
        jToggleButtonChannel7.setForeground(new java.awt.Color(28, 28, 28));
        jToggleButtonChannel7.setIcon(new
javax.swing.ImageIcon(getClass().getResource("/ntnusubsea/gui/Images/Toggle_of
f.png"))); // NOI18N
        jToggleButtonChannel7.setBorder(null);
        jToggleButtonChannel7.setEnabled(false);
        jToggleButtonChannel7.setSelectedIcon(new
javax.swing.ImageIcon(getClass().getResource("/ntnusubsea/gui/Images/Toggle_on
```

```
.png"))); // 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(28, 28, 28));
        jLabelIndicatorChannel7.setIcon(new
javax.swing.ImageIcon(getClass().getResource("/ntnusubsea/gui/Images/IO_Off.pn
g"))); // 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()
                .addComponent(jToggleButtonChannel7,
javax.swing.GroupLayout.PREFERRED_SIZE, 46,
javax.swing.GroupLayout.PREFERRED_SIZE)
```

```
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UN
RELATED)
                .addComponent(jLabelIndicatorChannel7,
javax.swing.GroupLayout.PREFERRED_SIZE, 58,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RE
LATED)
                .addGroup(jPanelChannel7Layout.createParallelGroup(javax.swing
.GroupLayout.Alignment.LEADING)
                    .addComponent(jLabelOffChannel7,
javax.swing.GroupLayout.DEFAULT_SIZE, 51, 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)
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RE
LATED)
                .addComponent(jLabelChannel7Value,
javax.swing.GroupLayout.PREFERRED_SIZE, 50,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RE
LATED)
                .addGroup(jPanelChannel7Layout.createParallelGroup(javax.swing
.GroupLayout.Alignment.LEADING)
                    .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
jPanelChannel7Layout.createSequentialGroup()
                        .addComponent(jLabelOnChannel7,
javax.swing.GroupLayout.PREFERRED_SIZE, 20,
javax.swing.GroupLayout.PREFERRED_SIZE)
                        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlac
ement.RELATED)
```

```
                        .addComponent(jLabelOffChannel7,
javax.swing.GroupLayout.PREFERRED_SIZE, 20,
javax.swing.GroupLayout.PREFERRED_SIZE)
                        .addGap(11, 11, 11))
                    .addComponent(jToggleButtonChannel7,
javax.swing.GroupLayout.Alignment.TRAILING,
javax.swing.GroupLayout.PREFERRED_SIZE, 63,
javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addComponent(jLabelIndicatorChannel7,
javax.swing.GroupLayout.Alignment.TRAILING,
javax.swing.GroupLayout.PREFERRED_SIZE, 63,
javax.swing.GroupLayout.PREFERRED_SIZE)))
        );

        jPanelChannel8.setBackground(new java.awt.Color(28, 28, 28));

        jLabelChannel8Header.setBackground(new java.awt.Color(28, 28, 28));
        jLabelChannel8Header.setFont(new java.awt.Font("Tahoma", 0, 14)); //
NOI18N
        jLabelChannel8Header.setForeground(new java.awt.Color(255, 255, 255));
        jLabelChannel8Header.setText("Channel 8 (Output)");

        jToggleButtonChannel8.setBackground(new java.awt.Color(28, 28, 28));
        jToggleButtonChannel8.setForeground(new java.awt.Color(28, 28, 28));
        jToggleButtonChannel8.setIcon(new
javax.swing.ImageIcon(getClass().getResource("/ntnusubsea/gui/Images/Toggle_of
f.png"))); // NOI18N
        jToggleButtonChannel8.setBorder(null);
        jToggleButtonChannel8.setEnabled(false);
        jToggleButtonChannel8.setSelectedIcon(new
javax.swing.ImageIcon(getClass().getResource("/ntnusubsea/gui/Images/Toggle_on
.png"))); // NOI18N
        jToggleButtonChannel8.addActionListener(new
java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jToggleButtonChannel8ActionPerformed(evt);
            }
        });

        jLabelChannel8Value.setBackground(new java.awt.Color(28, 28, 28));
        jLabelChannel8Value.setFont(new java.awt.Font("Tahoma", 0, 14)); //
```

```
NOI18N
        jLabelChannel8Value.setForeground(new java.awt.Color(255, 255, 255));
        jLabelChannel8Value.setText("jLabel2");

        jLabelOnChannel8.setBackground(new java.awt.Color(28, 28, 28));
        jLabelOnChannel8.setForeground(new java.awt.Color(255, 255, 255));
        jLabelOnChannel8.setText("On");

        jLabelOffChannel8.setBackground(new java.awt.Color(28, 28, 28));
        jLabelOffChannel8.setForeground(new java.awt.Color(255, 255, 255));
        jLabelOffChannel8.setText("Off");

        jLabelIndicatorChannel8.setBackground(new java.awt.Color(28, 28, 28));
        jLabelIndicatorChannel8.setIcon(new
javax.swing.ImageIcon(getClass().getResource("/ntnusubsea/gui/Images/IO_Off.pn
g"))); // NOI18N
        jLabelIndicatorChannel8.setOpaque(true);

        javax.swing.GroupLayout jPanelChannel8Layout = new
javax.swing.GroupLayout(jPanelChannel8);
        jPanelChannel8.setLayout(jPanelChannel8Layout);
        jPanelChannel8Layout.setHorizontalGroup(
            jPanelChannel8Layout.createParallelGroup(javax.swing.GroupLayout.
Alignment.LEADING)
            .addComponent(jLabelChannel8Header,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)
            .addGroup(jPanelChannel8Layout.createSequentialGroup()
                .addComponent(jToggleButtonChannel8,
javax.swing.GroupLayout.PREFERRED_SIZE, 46,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UN
RELATED)
                .addComponent(jLabelIndicatorChannel8,
javax.swing.GroupLayout.PREFERRED_SIZE, 58,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RE
LATED)
                .addGroup(jPanelChannel8Layout.createParallelGroup(javax.swing
.GroupLayout.Alignment.LEADING)
                    .addComponent(jLabelOffChannel8,
```

```
javax.swing.GroupLayout.DEFAULT_SIZE, 42, Short.MAX_VALUE)
                    .addComponent(jLabelOnChannel8,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)))
            .addComponent(jLabelChannel8Value,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)
        );
        jPanelChannel8Layout.setVerticalGroup(
            jPanelChannel8Layout.createParallelGroup(javax.swing.GroupLayout.
Alignment.LEADING)
            .addGroup(jPanelChannel8Layout.createSequentialGroup()
                .addComponent(jLabelChannel8Header,
javax.swing.GroupLayout.PREFERRED_SIZE, 28,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RE
LATED)
                .addComponent(jLabelChannel8Value,
javax.swing.GroupLayout.PREFERRED_SIZE, 50,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RE
LATED)
                .addGroup(jPanelChannel8Layout.createParallelGroup(javax.swing
.GroupLayout.Alignment.LEADING)
                    .addComponent(jLabelIndicatorChannel8,
javax.swing.GroupLayout.Alignment.TRAILING,
javax.swing.GroupLayout.PREFERRED_SIZE, 63,
javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addComponent(jToggleButtonChannel8,
javax.swing.GroupLayout.Alignment.TRAILING,
javax.swing.GroupLayout.PREFERRED_SIZE, 63,
javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
jPanelChannel8Layout.createSequentialGroup()
                        .addComponent(jLabelOnChannel8,
javax.swing.GroupLayout.PREFERRED_SIZE, 20,
javax.swing.GroupLayout.PREFERRED_SIZE)
                        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlac
ement.RELATED)
                        .addComponent(jLabelOffChannel8,
javax.swing.GroupLayout.PREFERRED_SIZE, 20,
```

```
javax.swing.GroupLayout.PREFERRED_SIZE)
                        .addGap(11, 11, 11))))
        );

        jPanelChannel1.setBackground(new java.awt.Color(28, 28, 28));
        jPanelChannel1.setPreferredSize(new java.awt.Dimension(160, 148));

        jLabelChannel1Header.setBackground(new java.awt.Color(28, 28, 28));
        jLabelChannel1Header.setFont(new java.awt.Font("Tahoma", 0, 14)); //
NOI18N
        jLabelChannel1Header.setForeground(new java.awt.Color(255, 255, 255));
        jLabelChannel1Header.setText("Channel 1 (Input)");

        jLabelChannel1Value.setBackground(new java.awt.Color(28, 28, 28));
        jLabelChannel1Value.setFont(new java.awt.Font("Tahoma", 0, 14)); //
NOI18N
        jLabelChannel1Value.setForeground(new java.awt.Color(255, 255, 255));
        jLabelChannel1Value.setText("jLabel2");

        jToggleButtonChannel1.setBackground(new java.awt.Color(28, 28, 28));
        jToggleButtonChannel1.setForeground(new java.awt.Color(28, 28, 28));
        jToggleButtonChannel1.setIcon(new
javax.swing.ImageIcon(getClass().getResource("/ntnusubsea/gui/Images/Toggle_of
f.png"))); // NOI18N
        jToggleButtonChannel1.setBorder(null);
        jToggleButtonChannel1.setEnabled(false);
        jToggleButtonChannel1.setSelectedIcon(new
javax.swing.ImageIcon(getClass().getResource("/ntnusubsea/gui/Images/Toggle_on
.png"))); // NOI18N
        jToggleButtonChannel1.addActionListener(new
java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jToggleButtonChannel1ActionPerformed(evt);
            }
        });

        jLabelIndicatorChannel1.setBackground(new java.awt.Color(28, 28, 28));
        jLabelIndicatorChannel1.setIcon(new
javax.swing.ImageIcon(getClass().getResource("/ntnusubsea/gui/Images/IO_Off.pn
g"))); // NOI18N
        jLabelIndicatorChannel1.setOpaque(true);
```

```
        jLabelOnChannel1.setBackground(new java.awt.Color(28, 28, 28));
        jLabelOnChannel1.setForeground(new java.awt.Color(255, 255, 255));
        jLabelOnChannel1.setText("On");

        jLabelOffChannel1.setBackground(new java.awt.Color(28, 28, 28));
        jLabelOffChannel1.setForeground(new java.awt.Color(255, 255, 255));
        jLabelOffChannel1.setText("Off");

        javax.swing.GroupLayout jPanelChannel1Layout = new
javax.swing.GroupLayout(jPanelChannel1);
        jPanelChannel1.setLayout(jPanelChannel1Layout);
        jPanelChannel1Layout.setHorizontalGroup(
            jPanelChannel1Layout.createParallelGroup(javax.swing.GroupLayout.
Alignment.LEADING)
            .addComponent(jLabelChannel1Header,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)
            .addComponent(jLabelChannel1Value,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)
            .addGroup(jPanelChannel1Layout.createSequentialGroup()
                .addComponent(jToggleButtonChannel1,
javax.swing.GroupLayout.PREFERRED_SIZE, 46,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RE
LATED)
                .addComponent(jLabelIndicatorChannel1,
javax.swing.GroupLayout.PREFERRED_SIZE, 54,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RE
LATED)
                .addGroup(jPanelChannel1Layout.createParallelGroup(javax.swing
.GroupLayout.Alignment.LEADING)
                    .addComponent(jLabelOffChannel1,
javax.swing.GroupLayout.DEFAULT_SIZE, 47, Short.MAX_VALUE)
                    .addComponent(jLabelOnChannel1,
javax.swing.GroupLayout.DEFAULT_SIZE, 47, Short.MAX_VALUE))
                .addContainerGap())
        );
        jPanelChannel1Layout.setVerticalGroup(
```

```
            jPanelChannel1Layout.createParallelGroup(javax.swing.GroupLayout.
Alignment.LEADING)
            .addGroup(jPanelChannel1Layout.createSequentialGroup()
                .addGroup(jPanelChannel1Layout.createParallelGroup(javax.swing
.GroupLayout.Alignment.TRAILING)
                    .addComponent(jLabelIndicatorChannel1,
javax.swing.GroupLayout.PREFERRED_SIZE, 63,
javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addGroup(jPanelChannel1Layout.createSequentialGroup()
                        .addComponent(jLabelChannel1Header,
javax.swing.GroupLayout.PREFERRED_SIZE, 28,
javax.swing.GroupLayout.PREFERRED_SIZE)
                        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlac
ement.RELATED)
                        .addComponent(jLabelChannel1Value,
javax.swing.GroupLayout.PREFERRED_SIZE, 60,
javax.swing.GroupLayout.PREFERRED_SIZE)
                        .addGroup(jPanelChannel1Layout.createParallelGroup(jav
ax.swing.GroupLayout.Alignment.LEADING)
                            .addGroup(jPanelChannel1Layout.createSequentialGro
up()
                                .addPreferredGap(javax.swing.LayoutStyle.Compo
nentPlacement.UNRELATED)
                                .addComponent(jToggleButtonChannel1,
javax.swing.GroupLayout.PREFERRED_SIZE, 63,
javax.swing.GroupLayout.PREFERRED_SIZE))
                            .addGroup(jPanelChannel1Layout.createSequentialGro
up()
                                .addGap(21, 21, 21)
                                .addComponent(jLabelOnChannel1,
javax.swing.GroupLayout.PREFERRED_SIZE, 20,
javax.swing.GroupLayout.PREFERRED_SIZE)
                                .addPreferredGap(javax.swing.LayoutStyle.Compo
nentPlacement.RELATED)
                                .addComponent(jLabelOffChannel1,
javax.swing.GroupLayout.PREFERRED_SIZE, 20,
javax.swing.GroupLayout.PREFERRED_SIZE)))))
                .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE))
        );
```

```
        jPanelChannel2.setBackground(new java.awt.Color(28, 28, 28));
        jPanelChannel2.setPreferredSize(new java.awt.Dimension(160, 148));

        jLabelChannel2Header.setBackground(new java.awt.Color(28, 28, 28));
        jLabelChannel2Header.setFont(new java.awt.Font("Tahoma", 0, 14)); //
NOI18N
        jLabelChannel2Header.setForeground(new java.awt.Color(255, 255, 255));
        jLabelChannel2Header.setText("Channel 2 (Input)");

        jLabelChannel2Value.setBackground(new java.awt.Color(28, 28, 28));
        jLabelChannel2Value.setFont(new java.awt.Font("Tahoma", 0, 14)); //
NOI18N
        jLabelChannel2Value.setForeground(new java.awt.Color(255, 255, 255));
        jLabelChannel2Value.setText("jLabel2");

        jToggleButtonChannel2.setBackground(new java.awt.Color(28, 28, 28));
        jToggleButtonChannel2.setForeground(new java.awt.Color(28, 28, 28));
        jToggleButtonChannel2.setIcon(new
javax.swing.ImageIcon(getClass().getResource("/ntnusubsea/gui/Images/Toggle_of
f.png"))); // NOI18N
        jToggleButtonChannel2.setBorder(null);
        jToggleButtonChannel2.setEnabled(false);
        jToggleButtonChannel2.setSelectedIcon(new
javax.swing.ImageIcon(getClass().getResource("/ntnusubsea/gui/Images/Toggle_on
.png"))); // NOI18N
        jToggleButtonChannel2.addActionListener(new
java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jToggleButtonChannel2ActionPerformed(evt);
            }
        });

        jLabelIndicatorChannel2.setBackground(new java.awt.Color(28, 28, 28));
        jLabelIndicatorChannel2.setIcon(new
javax.swing.ImageIcon(getClass().getResource("/ntnusubsea/gui/Images/IO_Off.pn
g"))); // NOI18N
        jLabelIndicatorChannel2.setOpaque(true);

        jLabelOnChannel2.setBackground(new java.awt.Color(28, 28, 28));
        jLabelOnChannel2.setForeground(new java.awt.Color(255, 255, 255));
        jLabelOnChannel2.setText("On");
```

```
        jLabelOffChannel2.setBackground(new java.awt.Color(28, 28, 28));
        jLabelOffChannel2.setForeground(new java.awt.Color(255, 255, 255));
        jLabelOffChannel2.setText("Off");

        javax.swing.GroupLayout jPanelChannel2Layout = new
javax.swing.GroupLayout(jPanelChannel2);
        jPanelChannel2.setLayout(jPanelChannel2Layout);
        jPanelChannel2Layout.setHorizontalGroup(
            jPanelChannel2Layout.createParallelGroup(javax.swing.GroupLayout.
Alignment.LEADING)
            .addComponent(jLabelChannel2Header,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)
            .addComponent(jLabelChannel2Value,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)
            .addGroup(jPanelChannel2Layout.createSequentialGroup()
                .addComponent(jToggleButtonChannel2,
javax.swing.GroupLayout.PREFERRED_SIZE, 46,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RE
LATED)
                .addComponent(jLabelIndicatorChannel2,
javax.swing.GroupLayout.PREFERRED_SIZE, 54,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RE
LATED)
                .addGroup(jPanelChannel2Layout.createParallelGroup(javax.swing
.GroupLayout.Alignment.LEADING)
                    .addGroup(jPanelChannel2Layout.createSequentialGroup()
                        .addComponent(jLabelOnChannel2,
javax.swing.GroupLayout.DEFAULT_SIZE, 47, Short.MAX_VALUE)
                        .addGap(16, 16, 16))
                    .addGroup(jPanelChannel2Layout.createSequentialGroup()
                        .addComponent(jLabelOffChannel2,
javax.swing.GroupLayout.DEFAULT_SIZE, 53, Short.MAX_VALUE)
                        .addContainerGap())))
        );
        jPanelChannel2Layout.setVerticalGroup(
            jPanelChannel2Layout.createParallelGroup(javax.swing.GroupLayout.
```

```
Alignment.LEADING)
            .addGroup(jPanelChannel2Layout.createSequentialGroup()
                .addComponent(jLabelChannel2Header,
javax.swing.GroupLayout.PREFERRED_SIZE, 28,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RE
LATED)
                .addComponent(jLabelChannel2Value,
javax.swing.GroupLayout.PREFERRED_SIZE, 60,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addGroup(jPanelChannel2Layout.createParallelGroup(javax.swing
.GroupLayout.Alignment.LEADING)
                    .addGroup(jPanelChannel2Layout.createSequentialGroup()
                        .addGap(19, 19, 19)
                        .addComponent(jLabelOnChannel2,
javax.swing.GroupLayout.PREFERRED_SIZE, 20,
javax.swing.GroupLayout.PREFERRED_SIZE)
                        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlac
ement.RELATED)
                        .addComponent(jLabelOffChannel2,
javax.swing.GroupLayout.PREFERRED_SIZE, 20,
javax.swing.GroupLayout.PREFERRED_SIZE))
                    .addGroup(jPanelChannel2Layout.createSequentialGroup()
                        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlac
ement.UNRELATED)
                        .addGroup(jPanelChannel2Layout.createParallelGroup(jav
ax.swing.GroupLayout.Alignment.LEADING)
                            .addComponent(jLabelIndicatorChannel2,
javax.swing.GroupLayout.PREFERRED_SIZE, 63,
javax.swing.GroupLayout.PREFERRED_SIZE)
                            .addComponent(jToggleButtonChannel2,
javax.swing.GroupLayout.PREFERRED_SIZE, 63,
javax.swing.GroupLayout.PREFERRED_SIZE))))
                .addContainerGap(23, Short.MAX_VALUE))
        );

        jPanelChannel3.setBackground(new java.awt.Color(28, 28, 28));
        jPanelChannel3.setPreferredSize(new java.awt.Dimension(160, 148));

        jLabelChannel3Header.setBackground(new java.awt.Color(28, 28, 28));
        jLabelChannel3Header.setFont(new java.awt.Font("Tahoma", 0, 14)); //
```

```
NOI18N
        jLabelChannel3Header.setForeground(new java.awt.Color(255, 255, 255));
        jLabelChannel3Header.setText("Channel 3 (Input)");

        jLabelChannel3Value.setBackground(new java.awt.Color(28, 28, 28));
        jLabelChannel3Value.setFont(new java.awt.Font("Tahoma", 0, 14)); //
NOI18N
        jLabelChannel3Value.setForeground(new java.awt.Color(255, 255, 255));
        jLabelChannel3Value.setText("jLabel2");

        jToggleButtonChannel3.setBackground(new java.awt.Color(28, 28, 28));
        jToggleButtonChannel3.setForeground(new java.awt.Color(28, 28, 28));
        jToggleButtonChannel3.setIcon(new
javax.swing.ImageIcon(getClass().getResource("/ntnusubsea/gui/Images/Toggle_of
f.png"))); // NOI18N
        jToggleButtonChannel3.setBorder(null);
        jToggleButtonChannel3.setEnabled(false);
        jToggleButtonChannel3.setSelectedIcon(new
javax.swing.ImageIcon(getClass().getResource("/ntnusubsea/gui/Images/Toggle_on
.png"))); // NOI18N
        jToggleButtonChannel3.addActionListener(new
java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jToggleButtonChannel3ActionPerformed(evt);
            }
        });

        jLabelIndicatorChannel3.setBackground(new java.awt.Color(28, 28, 28));
        jLabelIndicatorChannel3.setIcon(new
javax.swing.ImageIcon(getClass().getResource("/ntnusubsea/gui/Images/IO_Off.pn
g"))); // NOI18N
        jLabelIndicatorChannel3.setOpaque(true);

        jLabelOnChannel3.setBackground(new java.awt.Color(28, 28, 28));
        jLabelOnChannel3.setForeground(new java.awt.Color(255, 255, 255));
        jLabelOnChannel3.setText("On");

        jLabelOffChannel3.setBackground(new java.awt.Color(28, 28, 28));
        jLabelOffChannel3.setForeground(new java.awt.Color(255, 255, 255));
        jLabelOffChannel3.setText("Off");
```

```
        javax.swing.GroupLayout jPanelChannel3Layout = new
javax.swing.GroupLayout(jPanelChannel3);
        jPanelChannel3.setLayout(jPanelChannel3Layout);
        jPanelChannel3Layout.setHorizontalGroup(
            jPanelChannel3Layout.createParallelGroup(javax.swing.GroupLayout.
Alignment.LEADING)
            .addComponent(jLabelChannel3Header,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)
            .addComponent(jLabelChannel3Value,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)
            .addGroup(jPanelChannel3Layout.createSequentialGroup()
                .addComponent(jToggleButtonChannel3,
javax.swing.GroupLayout.PREFERRED_SIZE, 46,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RE
LATED)
                .addComponent(jLabelIndicatorChannel3,
javax.swing.GroupLayout.PREFERRED_SIZE, 54,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RE
LATED)
                .addGroup(jPanelChannel3Layout.createParallelGroup(javax.swing
.GroupLayout.Alignment.LEADING)
                    .addGroup(jPanelChannel3Layout.createSequentialGroup()
                        .addComponent(jLabelOnChannel3,
javax.swing.GroupLayout.DEFAULT_SIZE, 47, Short.MAX_VALUE)
                        .addGap(14, 14, 14))
                    .addGroup(jPanelChannel3Layout.createSequentialGroup()
                        .addComponent(jLabelOffChannel3,
javax.swing.GroupLayout.DEFAULT_SIZE, 51, Short.MAX_VALUE)
                        .addContainerGap())))
        );
        jPanelChannel3Layout.setVerticalGroup(
            jPanelChannel3Layout.createParallelGroup(javax.swing.GroupLayout.
Alignment.LEADING)
            .addGroup(jPanelChannel3Layout.createSequentialGroup()
                .addComponent(jLabelChannel3Header,
javax.swing.GroupLayout.PREFERRED_SIZE, 28,
javax.swing.GroupLayout.PREFERRED_SIZE)
```

```
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RE
LATED)
                .addComponent(jLabelChannel3Value,
javax.swing.GroupLayout.PREFERRED_SIZE, 60,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addGroup(jPanelChannel3Layout.createParallelGroup(javax.swing
.GroupLayout.Alignment.LEADING)
                    .addGroup(jPanelChannel3Layout.createSequentialGroup()
                        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlac
ement.UNRELATED)
                        .addGroup(jPanelChannel3Layout.createParallelGroup(jav
ax.swing.GroupLayout.Alignment.TRAILING)
                            .addComponent(jToggleButtonChannel3,
javax.swing.GroupLayout.PREFERRED_SIZE, 63,
javax.swing.GroupLayout.PREFERRED_SIZE)
                            .addComponent(jLabelIndicatorChannel3,
javax.swing.GroupLayout.PREFERRED_SIZE, 63,
javax.swing.GroupLayout.PREFERRED_SIZE)))
                    .addGroup(jPanelChannel3Layout.createSequentialGroup()
                        .addGap(19, 19, 19)
                        .addComponent(jLabelOnChannel3,
javax.swing.GroupLayout.PREFERRED_SIZE, 20,
javax.swing.GroupLayout.PREFERRED_SIZE)
                        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlac
ement.RELATED)
                        .addComponent(jLabelOffChannel3,
javax.swing.GroupLayout.PREFERRED_SIZE, 20,
javax.swing.GroupLayout.PREFERRED_SIZE)))
                .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE))
        );

        jPanelChannel4.setBackground(new java.awt.Color(28, 28, 28));
        jPanelChannel4.setPreferredSize(new java.awt.Dimension(160, 148));

        jLabelChannel4Header.setBackground(new java.awt.Color(28, 28, 28));
        jLabelChannel4Header.setFont(new java.awt.Font("Tahoma", 0, 14)); //
NOI18N
        jLabelChannel4Header.setForeground(new java.awt.Color(255, 255, 255));
        jLabelChannel4Header.setText("Channel 4 (Input)");
```

```
        jLabelChannel4Value.setBackground(new java.awt.Color(28, 28, 28));
        jLabelChannel4Value.setFont(new java.awt.Font("Tahoma", 0, 14)); //
NOI18N
        jLabelChannel4Value.setForeground(new java.awt.Color(255, 255, 255));
        jLabelChannel4Value.setText("jLabel2");

        jToggleButtonChannel4.setBackground(new java.awt.Color(28, 28, 28));
        jToggleButtonChannel4.setForeground(new java.awt.Color(28, 28, 28));
        jToggleButtonChannel4.setIcon(new
javax.swing.ImageIcon(getClass().getResource("/ntnusubsea/gui/Images/Toggle_of
f.png"))); // NOI18N
        jToggleButtonChannel4.setBorder(null);
        jToggleButtonChannel4.setEnabled(false);
        jToggleButtonChannel4.setSelectedIcon(new
javax.swing.ImageIcon(getClass().getResource("/ntnusubsea/gui/Images/Toggle_on
.png"))); // NOI18N
        jToggleButtonChannel4.addActionListener(new
java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jToggleButtonChannel4ActionPerformed(evt);
            }
        });

        jLabelIndicatorChannel4.setBackground(new java.awt.Color(28, 28, 28));
        jLabelIndicatorChannel4.setIcon(new
javax.swing.ImageIcon(getClass().getResource("/ntnusubsea/gui/Images/IO_Off.pn
g"))); // NOI18N
        jLabelIndicatorChannel4.setOpaque(true);

        jLabelOnChannel4.setBackground(new java.awt.Color(28, 28, 28));
        jLabelOnChannel4.setForeground(new java.awt.Color(255, 255, 255));
        jLabelOnChannel4.setText("On");

        jLabelOffChannel4.setBackground(new java.awt.Color(28, 28, 28));
        jLabelOffChannel4.setForeground(new java.awt.Color(255, 255, 255));
        jLabelOffChannel4.setText("Off");

        javax.swing.GroupLayout jPanelChannel4Layout = new
javax.swing.GroupLayout(jPanelChannel4);
        jPanelChannel4.setLayout(jPanelChannel4Layout);
        jPanelChannel4Layout.setHorizontalGroup(
```

```
            jPanelChannel4Layout.createParallelGroup(javax.swing.GroupLayout.
Alignment.LEADING)
            .addComponent(jLabelChannel4Header,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)
            .addComponent(jLabelChannel4Value,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)
            .addGroup(jPanelChannel4Layout.createSequentialGroup()
                .addComponent(jToggleButtonChannel4,
javax.swing.GroupLayout.PREFERRED_SIZE, 46,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RE
LATED)
                .addComponent(jLabelIndicatorChannel4,
javax.swing.GroupLayout.PREFERRED_SIZE, 54,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RE
LATED, javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                .addGroup(jPanelChannel4Layout.createParallelGroup(javax.swing
.GroupLayout.Alignment.LEADING)
                    .addComponent(jLabelOffChannel4,
javax.swing.GroupLayout.PREFERRED_SIZE, 50,
javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addComponent(jLabelOnChannel4,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE))
                .addContainerGap())
        );
        jPanelChannel4Layout.setVerticalGroup(
            jPanelChannel4Layout.createParallelGroup(javax.swing.GroupLayout.
Alignment.LEADING)
            .addGroup(jPanelChannel4Layout.createSequentialGroup()
                .addComponent(jLabelChannel4Header,
javax.swing.GroupLayout.PREFERRED_SIZE, 28,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RE
LATED)
                .addComponent(jLabelChannel4Value,
javax.swing.GroupLayout.PREFERRED_SIZE, 60,
javax.swing.GroupLayout.PREFERRED_SIZE)
```

```
                .addGroup(jPanelChannel4Layout.createParallelGroup(javax.swing
.GroupLayout.Alignment.LEADING)
                    .addGroup(jPanelChannel4Layout.createSequentialGroup()
                        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlac
ement.UNRELATED)
                        .addGroup(jPanelChannel4Layout.createParallelGroup(jav
ax.swing.GroupLayout.Alignment.TRAILING)
                            .addComponent(jToggleButtonChannel4,
javax.swing.GroupLayout.PREFERRED_SIZE, 63,
javax.swing.GroupLayout.PREFERRED_SIZE)
                            .addComponent(jLabelIndicatorChannel4,
javax.swing.GroupLayout.PREFERRED_SIZE, 63,
javax.swing.GroupLayout.PREFERRED_SIZE)))
                    .addGroup(jPanelChannel4Layout.createSequentialGroup()
                        .addGap(19, 19, 19)
                        .addComponent(jLabelOnChannel4,
javax.swing.GroupLayout.PREFERRED_SIZE, 20,
javax.swing.GroupLayout.PREFERRED_SIZE)
                        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlac
ement.RELATED)
                        .addComponent(jLabelOffChannel4,
javax.swing.GroupLayout.PREFERRED_SIZE, 20,
javax.swing.GroupLayout.PREFERRED_SIZE)))
                .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE))
        );

        jSeparator1.setBackground(new java.awt.Color(28, 28, 28));

        javax.swing.GroupLayout jPanel9Layout = new
javax.swing.GroupLayout(jPanel9);
        jPanel9.setLayout(jPanel9Layout);
        jPanel9Layout.setHorizontalGroup(
            jPanel9Layout.createParallelGroup(javax.swing.GroupLayout.Alignmen
t.LEADING)
            .addGroup(jPanel9Layout.createSequentialGroup()
                .addContainerGap(11, Short.MAX_VALUE)
                .addGroup(jPanel9Layout.createParallelGroup(javax.swing.GroupL
ayout.Alignment.LEADING)
                    .addGroup(jPanel9Layout.createSequentialGroup()
                        .addComponent(jPanelChannel1,
```

```
javax.swing.GroupLayout.DEFAULT_SIZE, 167, Short.MAX_VALUE)
                        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlac
ement.UNRELATED, javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                        .addComponent(jPanelChannel2,
javax.swing.GroupLayout.DEFAULT_SIZE, 173, Short.MAX_VALUE)
                        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlac
ement.UNRELATED, javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                        .addComponent(jPanelChannel3,
javax.swing.GroupLayout.DEFAULT_SIZE, 171, Short.MAX_VALUE)
                        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlac
ement.UNRELATED, javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                        .addComponent(jPanelChannel4,
javax.swing.GroupLayout.PREFERRED_SIZE, 164,
javax.swing.GroupLayout.PREFERRED_SIZE))
                    .addGroup(jPanel9Layout.createSequentialGroup()
                        .addComponent(jPanelChannel5,
javax.swing.GroupLayout.PREFERRED_SIZE, 166,
javax.swing.GroupLayout.PREFERRED_SIZE)
                        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlac
ement.UNRELATED, 12, Short.MAX_VALUE)
                        .addComponent(jPanelChannel6,
javax.swing.GroupLayout.PREFERRED_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
                        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlac
ement.UNRELATED, 12, Short.MAX_VALUE)
                        .addComponent(jPanelChannel7,
javax.swing.GroupLayout.PREFERRED_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
                        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlac
ement.UNRELATED, 12, Short.MAX_VALUE)
                        .addComponent(jPanelChannel8,
javax.swing.GroupLayout.PREFERRED_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)))
                .addContainerGap(13, Short.MAX_VALUE))
            .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
jPanel9Layout.createSequentialGroup()
                .addContainerGap()
                .addComponent(jSeparator1)
                .addContainerGap())
        );
        jPanel9Layout.setVerticalGroup(
```

```
            jPanel9Layout.createParallelGroup(javax.swing.GroupLayout.Alignmen
t.LEADING)
            .addGroup(jPanel9Layout.createSequentialGroup()
                .addGap(10, 10, 10)
                .addGroup(jPanel9Layout.createParallelGroup(javax.swing.GroupL
ayout.Alignment.TRAILING, false)
                    .addComponent(jPanelChannel3,
javax.swing.GroupLayout.DEFAULT_SIZE, 191, Short.MAX_VALUE)
                    .addComponent(jPanelChannel2,
javax.swing.GroupLayout.Alignment.LEADING,
javax.swing.GroupLayout.DEFAULT_SIZE, 191, Short.MAX_VALUE)
                    .addComponent(jPanelChannel1,
javax.swing.GroupLayout.Alignment.LEADING,
javax.swing.GroupLayout.DEFAULT_SIZE, 191, Short.MAX_VALUE)
                    .addComponent(jPanelChannel4,
javax.swing.GroupLayout.DEFAULT_SIZE, 191, Short.MAX_VALUE))
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RE
LATED, javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                .addComponent(jSeparator1,
javax.swing.GroupLayout.PREFERRED_SIZE, 10,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UN
RELATED)
                .addGroup(jPanel9Layout.createParallelGroup(javax.swing.GroupL
ayout.Alignment.TRAILING)
                    .addComponent(jPanelChannel7,
javax.swing.GroupLayout.Alignment.LEADING,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)
                    .addComponent(jPanelChannel6,
javax.swing.GroupLayout.Alignment.LEADING,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)
                    .addComponent(jPanelChannel5,
javax.swing.GroupLayout.DEFAULT_SIZE, 176, Short.MAX_VALUE)
                    .addComponent(jPanelChannel8,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE))
                .addGap(10, 10, 10))
        );
```

```
        javax.swing.GroupLayout layout = new
javax.swing.GroupLayout(getContentPane());
        getContentPane().setLayout(layout);
        layout.setHorizontalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADI
NG)
            .addComponent(jPanel9, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
        );
        layout.setVerticalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADI
NG)
            .addComponent(jPanel9, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
        );

        pack();
    }// </editor-fold>

    private void
jToggleButtonChannel5ActionPerformed(java.awt.event.ActionEvent evt) {
        try {
            if (jToggleButtonChannel5.isSelected()) {
                setBit(4, 1);
                //client.sendCommand(DIGITALOUTID + outputValue);
                jLabelIndicatorChannel5.setIcon(new ImageIcon(ImageIO.read(new
File("src/ntnusubsea/gui/Images/IO_On.png"))));
            } else {
                setBit(4, 0);
                //client.sendCommand(DIGITALOUTID + outputValue);
                jLabelIndicatorChannel5.setIcon(new ImageIcon(ImageIO.read(new
File("src/ntnusubsea/gui/Images/IO_Off.png"))));
            }
            System.out.println(outputValue);
        } catch (IOException ex) {
            System.out.println(ex.getMessage());
        }
    }

    private void
jToggleButtonChannel6ActionPerformed(java.awt.event.ActionEvent evt) {
```

```
        try {
            if (jToggleButtonChannel6.isSelected()) {
                setBit(5, 1);
                //client.sendCommand(DIGITALOUTID + outputValue);
                jLabelIndicatorChannel6.setIcon(new ImageIcon(ImageIO.read(new
File("src/ntnusubsea/gui/Images/IO_On.png"))));
            } else {
                setBit(5, 0);
                //client.sendCommand(DIGITALOUTID + outputValue);
                jLabelIndicatorChannel6.setIcon(new ImageIcon(ImageIO.read(new
File("src/ntnusubsea/gui/Images/IO_Off.png"))));
            }
            System.out.println(outputValue);
        } catch (IOException ex) {
            System.out.println(ex.getMessage());
        }
    }

    private void
jToggleButtonChannel7ActionPerformed(java.awt.event.ActionEvent evt) {
        try {
            if (jToggleButtonChannel7.isSelected()) {
                setBit(6, 1);
                //client.sendCommand(DIGITALOUTID + outputValue);
                jLabelIndicatorChannel7.setIcon(new ImageIcon(ImageIO.read(new
File("src/ntnusubsea/gui/Images/IO_On.png"))));
            } else {
                setBit(6, 0);
                //client.sendCommand(DIGITALOUTID + outputValue);
                jLabelIndicatorChannel7.setIcon(new ImageIcon(ImageIO.read(new
File("src/ntnusubsea/gui/Images/IO_Off.png"))));
            }
            System.out.println(outputValue);
        } catch (IOException ex) {
            System.out.println(ex.getMessage());
        }
    }

    private void
jToggleButtonChannel8ActionPerformed(java.awt.event.ActionEvent evt) {
        try {
```

```
            if (jToggleButtonChannel8.isSelected()) {
                setBit(7, 1);
                //client.sendCommand(DIGITALOUTID + outputValue);
                jLabelIndicatorChannel8.setIcon(new ImageIcon(ImageIO.read(new
File("src/ntnusubsea/gui/Images/IO_On.png"))));
            } else {
                setBit(7, 0);
                //client.sendCommand(DIGITALOUTID + outputValue);
                jLabelIndicatorChannel8.setIcon(new ImageIcon(ImageIO.read(new
File("src/ntnusubsea/gui/Images/IO_Off.png"))));
            }
            System.out.println(outputValue);
        } catch (IOException ex) {
            System.out.println(ex.getMessage());
        }
    }

    private void
jToggleButtonChannel1ActionPerformed(java.awt.event.ActionEvent evt) {
        try {
            if (jToggleButtonChannel1.isSelected()) {
                setBit(0, 1);
                //client.sendCommand(DIGITALOUTID + outputValue);
                jLabelIndicatorChannel1.setIcon(new ImageIcon(ImageIO.read(new
File("src/ntnusubsea/gui/Images/IO_On.png"))));
            } else {
                setBit(0, 0);
                //client.sendCommand(DIGITALOUTID + outputValue);
                jLabelIndicatorChannel1.setIcon(new ImageIcon(ImageIO.read(new
File("src/ntnusubsea/gui/Images/IO_Off.png"))));
            }
            System.out.println(outputValue);
        } catch (IOException ex) {
            System.out.println(ex.getMessage());
        }
    }

    private void
jToggleButtonChannel2ActionPerformed(java.awt.event.ActionEvent evt) {
        try {
            if (jToggleButtonChannel2.isSelected()) {
```

```
                setBit(1, 1);
                //client.sendCommand(DIGITALOUTID + outputValue);
                jLabelIndicatorChannel2.setIcon(new ImageIcon(ImageIO.read(new
File("src/ntnusubsea/gui/Images/IO_On.png"))));
            } else {
                setBit(1, 0);
                //client.sendCommand(DIGITALOUTID + outputValue);
                jLabelIndicatorChannel2.setIcon(new ImageIcon(ImageIO.read(new
File("src/ntnusubsea/gui/Images/IO_Off.png"))));
            }
            System.out.println(outputValue);
        } catch (IOException ex) {
            System.out.println(ex.getMessage());
        }
    }

    private void
jToggleButtonChannel3ActionPerformed(java.awt.event.ActionEvent evt) {
        try {
            if (jToggleButtonChannel3.isSelected()) {
                setBit(2, 1);
                //client.sendCommand(DIGITALOUTID + outputValue);
                jLabelIndicatorChannel3.setIcon(new ImageIcon(ImageIO.read(new
File("src/ntnusubsea/gui/Images/IO_On.png"))));
            } else {
                setBit(2, 0);
                //client.sendCommand(DIGITALOUTID + outputValue);
                jLabelIndicatorChannel3.setIcon(new ImageIcon(ImageIO.read(new
File("src/ntnusubsea/gui/Images/IO_Off.png"))));
            }
            System.out.println(outputValue);
        } catch (IOException ex) {
            System.out.println(ex.getMessage());
        }
    }

    private void
jToggleButtonChannel4ActionPerformed(java.awt.event.ActionEvent evt) {
        try {
            if (jToggleButtonChannel4.isSelected()) {
                setBit(3, 1);
```

```
                //client.sendCommand(DIGITALOUTID + outputValue);
                jLabelIndicatorChannel4.setIcon(new ImageIcon(ImageIO.read(new
File("src/ntnusubsea/gui/Images/IO_On.png"))));
            } else {
                setBit(3, 0);
                //client.sendCommand(DIGITALOUTID + outputValue);
                jLabelIndicatorChannel4.setIcon(new ImageIcon(ImageIO.read(new
File("src/ntnusubsea/gui/Images/IO_Off.png"))));
            }
            System.out.println(outputValue);
        } catch (IOException ex) {
            System.out.println(ex.getMessage());
        }
    }

    public void enableIO() {
        jToggleButtonChannel1.setEnabled(true);
        jToggleButtonChannel2.setEnabled(true);
        jToggleButtonChannel3.setEnabled(true);
        jToggleButtonChannel4.setEnabled(true);
        jToggleButtonChannel5.setEnabled(true);
        jToggleButtonChannel6.setEnabled(true);
        jToggleButtonChannel7.setEnabled(true);
        jToggleButtonChannel8.setEnabled(true);
    }

    public void disableIO() {
        jToggleButtonChannel1.setEnabled(false);
        jToggleButtonChannel2.setEnabled(false);
        jToggleButtonChannel3.setEnabled(false);
        jToggleButtonChannel4.setEnabled(false);
        jToggleButtonChannel5.setEnabled(false);
        jToggleButtonChannel6.setEnabled(false);
        jToggleButtonChannel7.setEnabled(false);
        jToggleButtonChannel8.setEnabled(false);
    }

    // Variables declaration - do not modify
    private javax.swing.JLabel jLabelChannel1Header;
    private javax.swing.JLabel jLabelChannel1Value;
    private javax.swing.JLabel jLabelChannel2Header;
```

```
    private javax.swing.JLabel jLabelChannel2Value;
    private javax.swing.JLabel jLabelChannel3Header;
    private javax.swing.JLabel jLabelChannel3Value;
    private javax.swing.JLabel jLabelChannel4Header;
    private javax.swing.JLabel jLabelChannel4Value;
    private javax.swing.JLabel jLabelChannel5Header;
    private javax.swing.JLabel jLabelChannel5Value;
    private javax.swing.JLabel jLabelChannel6Header;
    private javax.swing.JLabel jLabelChannel6Value;
    private javax.swing.JLabel jLabelChannel7Header;
    private javax.swing.JLabel jLabelChannel7Value;
    private javax.swing.JLabel jLabelChannel8Header;
    private javax.swing.JLabel jLabelChannel8Value;
    private javax.swing.JLabel jLabelIndicatorChannel1;
    private javax.swing.JLabel jLabelIndicatorChannel2;
    private javax.swing.JLabel jLabelIndicatorChannel3;
    private javax.swing.JLabel jLabelIndicatorChannel4;
    private javax.swing.JLabel jLabelIndicatorChannel5;
    private javax.swing.JLabel jLabelIndicatorChannel6;
    private javax.swing.JLabel jLabelIndicatorChannel7;
    private javax.swing.JLabel jLabelIndicatorChannel8;
    private javax.swing.JLabel jLabelOffChannel1;
    private javax.swing.JLabel jLabelOffChannel2;
    private javax.swing.JLabel jLabelOffChannel3;
    private javax.swing.JLabel jLabelOffChannel4;
    private javax.swing.JLabel jLabelOffChannel5;
    private javax.swing.JLabel jLabelOffChannel6;
    private javax.swing.JLabel jLabelOffChannel7;
    private javax.swing.JLabel jLabelOffChannel8;
    private javax.swing.JLabel jLabelOnChannel1;
    private javax.swing.JLabel jLabelOnChannel2;
    private javax.swing.JLabel jLabelOnChannel3;
    private javax.swing.JLabel jLabelOnChannel4;
    private javax.swing.JLabel jLabelOnChannel5;
    private javax.swing.JLabel jLabelOnChannel6;
    private javax.swing.JLabel jLabelOnChannel7;
    private javax.swing.JLabel jLabelOnChannel8;
    private javax.swing.JPanel jPanel9;
    private javax.swing.JPanel jPanelChannel1;
    private javax.swing.JPanel jPanelChannel2;
    private javax.swing.JPanel jPanelChannel3;
```

```
    private javax.swing.JPanel jPanelChannel4;
    private javax.swing.JPanel jPanelChannel5;
    private javax.swing.JPanel jPanelChannel6;
    private javax.swing.JPanel jPanelChannel7;
    private javax.swing.JPanel jPanelChannel8;
    private javax.swing.JSeparator jSeparator1;
    private javax.swing.JToggleButton jToggleButtonChannel1;
    private javax.swing.JToggleButton jToggleButtonChannel2;
    private javax.swing.JToggleButton jToggleButtonChannel3;
    private javax.swing.JToggleButton jToggleButtonChannel4;
    private javax.swing.JToggleButton jToggleButtonChannel5;
    private javax.swing.JToggleButton jToggleButtonChannel6;
    private javax.swing.JToggleButton jToggleButtonChannel7;
    private javax.swing.JToggleButton jToggleButtonChannel8;
    // End of variables declaration

    @Override
    public void run() {
        this.setVisible(false);
    }

    @Override
    public void update(Observable o, Object arg) {
        jLabelChannel1Value.setText(data.getChannel(1));
        jLabelChannel2Value.setText(data.getChannel(2));
        jLabelChannel3Value.setText(data.getChannel(3));
        jLabelChannel4Value.setText(data.getChannel(4));
    }

    private void setBit(int bit, int value) {
        if (value == 0) {
            outputValue = outputValue & ~(1 << bit);
        } else {
            outputValue = outputValue | (1 << bit);
        }
    }
}
```

```
/*
 * To change this license header, choose License Headers in Project
Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package ntnusubsea.gui;

import java.awt.Color;
import java.util.Observable;
import java.util.Observer;
import javax.swing.JOptionPane;

/**
 * Frame to display a graph panel
 * @author Marius Nonsvik
 */
public class EchoSounderFrame extends javax.swing.JFrame implements Runnable,
Observer {

    private Data data;
    private GraphPanel graph;

    /**
     * Creates new form SonarFrame
     * @param data Data containing shared variables
     */
    public EchoSounderFrame(Data data) {
        initComponents();
        this.data = data;
        graph = new GraphPanel();
        graph.setSize(jPanel1.getWidth(), jPanel1.getHeight());
        graph.setBackground(Color.BLACK);
        graph.setForeground(Color.WHITE);
        graph.setVisible(true);
        jPanel1.add(graph);
        this.add(jPanel1);
        this.pack();
    }

    /**
```

```
     * Draws and displays the graphs
     */
    public void showGraph() {
        graph.createAndShowGraph();
    }

    /**
     * |- Not finished -|
     * Refreshes the graphs with the given depth and position of the 2nd
     * graph in the x axis x-axis
     * @param position X position of second graph
     * @param depth Depth of the graphs
     */
    public void refreshGraph(int position, int depth) {
        graph.drawGraph(position, depth);
    }

    /**
     * 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() {

        jPanel1 = new javax.swing.JPanel();
        jMenuBar1 = new javax.swing.JMenuBar();
        jMenu1 = new javax.swing.JMenu();
        jMenuItem1 = new javax.swing.JMenuItem();
        jMenu2 = new javax.swing.JMenu();
        jMenuItem2 = new javax.swing.JMenuItem();

        setDefaultCloseOperation(javax.swing.WindowConstants.DISPOSE_ON_CLOSE)
;
        setTitle("Echo sounder");
        setPreferredSize(new java.awt.Dimension(800, 600));

        jPanel1.setLayout(new java.awt.BorderLayout());
        getContentPane().add(jPanel1, java.awt.BorderLayout.CENTER);
```

```
        jMenu1.setText("File");

        jMenuItem1.setText("Exit");
        jMenuItem1.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jMenuItem1ActionPerformed(evt);
            }
        });
        jMenu1.add(jMenuItem1);

        jMenuBar1.add(jMenu1);

        jMenu2.setText("Tools");

        jMenuItem2.setText("Calibrate");
        jMenuItem2.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jMenuItem2ActionPerformed(evt);
            }
        });
        jMenu2.add(jMenuItem2);

        jMenuBar1.add(jMenu2);

        setJMenuBar(jMenuBar1);

        pack();
    }// </editor-fold>

    private void jMenuItem1ActionPerformed(java.awt.event.ActionEvent evt) {
        this.dispose();
    }

    private void jMenuItem2ActionPerformed(java.awt.event.ActionEvent evt) {
        String cableLength = (String) JOptionPane.showInputDialog(this, "Enten
current cable length (Meters)", "Calibration", JOptionPane.PLAIN_MESSAGE,
null, null, "100.000");
        try {
            System.out.println(Float.valueOf(cableLength));
        } catch (NumberFormatException | NullPointerException ex) {
```

```
            System.out.println("Invalid or no input");
        }
    }

    @Override
    public void run() {
        showGraph();
    }


    // Variables declaration - do not modify
    private javax.swing.JMenu jMenu1;
    private javax.swing.JMenu jMenu2;
    private javax.swing.JMenuBar jMenuBar1;
    private javax.swing.JMenuItem jMenuItem1;
    private javax.swing.JMenuItem jMenuItem2;
    private javax.swing.JPanel jPanel1;
    // End of variables declaration

    @Override
    public void update(Observable o, Object arg) {
     //int depth = Math.round(data.getDepth() * 100);
    // refreshGraph(0, depth);
    }
}
```

```
/*
 * To change this license header, choose License Headers in Project
Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package ntnusubsea.gui;

import java.awt.BasicStroke;
import java.awt.Color;
import java.awt.Graphics;
import java.awt.Graphics2D;
import java.awt.Point;
import java.awt.RenderingHints;
import java.awt.Stroke;
import java.util.ArrayList;
import java.util.List;
import java.util.Random;
import javax.swing.*;

/**
 * A class that represents a panel where graphs can we drawn to display data
 * to the user in a visual manner.
 * @author Marius Nonsvik
 */
@SuppressWarnings("serial")
public class GraphPanel extends JPanel {

    private final int MAX_SCORE = 65536;
    private final int BORDER_GAP = 30;
    private final Stroke GRAPH_STROKE = new BasicStroke(2);
    private final int GRAPH_START_POINT = 12;
    private final int HATCH_CNT = 10;
    private ArrayList<Integer> scores;
    private ArrayList<Integer> scores2;
    private int maxDataPoints = 5000;
    private int maxScore = 5;

    /**
     * Creates an instance of GraphPanel
     */
```

```
    public GraphPanel() {
        this.scores = new ArrayList<Integer>();
        this.scores2 = new ArrayList<Integer>();
    }

    @Override
    protected void paintComponent(Graphics g) {
        super.paintComponent(g);
        Graphics2D g2 = (Graphics2D) g;
        g2.setRenderingHint(RenderingHints.KEY_ANTIALIASING,
RenderingHints.VALUE_ANTIALIAS_ON);

        double xScale = ((double) getWidth() - 2 * BORDER_GAP) /
(scores2.size() - 1);
        double yScale = ((double) getHeight() - 2 * BORDER_GAP) / (MAX_SCORE -
1);

        List<Point> graphPoints = new ArrayList<Point>();
        for (int i = 0; i < scores.size(); i++) {
            int x1 = (int) (i * xScale + BORDER_GAP);
            int y1 = (int) ((MAX_SCORE - scores.get(i)) * yScale +
BORDER_GAP);
            graphPoints.add(new Point(x1, y1));
        }

        List<Point> graphPoints2 = new ArrayList<Point>();
        for (int i = 0; i < scores2.size(); i++) {
            int x1 = (int) (i * xScale + BORDER_GAP);
            int y1 = (int) ((MAX_SCORE - scores2.get(i)) * yScale +
BORDER_GAP);
            graphPoints2.add(new Point(x1, y1));
        }

        // create x and y axes
        g2.drawLine(BORDER_GAP, getHeight() - BORDER_GAP, BORDER_GAP,
BORDER_GAP);
        g2.drawLine(BORDER_GAP, getHeight() - BORDER_GAP, getWidth() -
BORDER_GAP, getHeight() - BORDER_GAP);

        // create hatch marks for y axis.
        for (int i = 0; i < HATCH_CNT; i++) {
```

```
            int x0 = BORDER_GAP;
            int x1 = GRAPH_START_POINT + BORDER_GAP;
            int y0 = getHeight() - (((i + 1) * (getHeight() - BORDER_GAP * 2))
/ HATCH_CNT + BORDER_GAP);
            int y1 = y0;
            g2.drawLine(x0, y0, x1, y1);
        }

        // and for x axis
        for (int i = 0; i < HATCH_CNT; i++) {
            int x0 = getWidth() - (((i + 1) * (getWidth() - BORDER_GAP * 2)) /
HATCH_CNT + BORDER_GAP);
            int x1 = x0;
            int y0 = getHeight() - GRAPH_START_POINT - BORDER_GAP;
            int y1 = getHeight() - BORDER_GAP;
            g2.drawLine(x0, y0, x1, y1);

        }

        Stroke oldStroke = g2.getStroke();
        g2.setColor(Color.GREEN);
        g2.setStroke(GRAPH_STROKE);
        for (int i = 0; i < graphPoints.size() - 1; i++) {
            int x1 = graphPoints.get(i).x;
            int y1 = graphPoints.get(i).y;
            int x2 = graphPoints.get(i + 1).x;
            int y2 = graphPoints.get(i + 1).y;
            g2.drawLine(x1, y1, x2, y2);
        }
        g2.setColor(Color.YELLOW);
        for (int i = 0; i < graphPoints2.size() - 1; i++) {
            int x1 = graphPoints2.get(i).x;
            int y1 = graphPoints2.get(i).y;
            int x2 = graphPoints2.get(i + 1).x;
            int y2 = graphPoints2.get(i + 1).y;
            g2.drawLine(x1, y1, x2, y2);
        }
        g2.setStroke(oldStroke);

    }
```

```
    /**
     * Creates and displays the graph
     */
    public void createAndShowGraph() {
        Random random = new Random();
        for (int i = 0; i < maxDataPoints - 2000; i++) {
            scores.add(random.nextInt(maxScore) + 11);
        }
        for (int i = 0; i < maxDataPoints; i++) {
            scores2.add(random.nextInt(maxScore) + 1);
        }
        revalidate();
        repaint();

    }

    /**
     * Draws one new point in the graph and moves the old data 1 step further
     * back.
     * @param position Position of the ROV compared to the vessel
     * @param depth Depth of the ROV
     */
    public void drawGraph(int position, int depth) {
        // int lastNumber = scores.set(maxDataPoints-200 - 1, depth + 11);
        int lastNumber = scores.set(maxDataPoints - 2001,
scores2.get(maxDataPoints - 2000)+10000);
        for (int i = maxDataPoints - 2000 - 2; i >= 0; i--) {
            lastNumber = scores.set(i, lastNumber);
        }
        int lastNumber2 = scores2.set(maxDataPoints - 1, depth + 1);
        for (int i = maxDataPoints - 2; i >= 0; i--) {
            lastNumber2 = scores2.set(i, lastNumber2);
        }
        repaint();
    }
}
```

```
/*
 * To change this license header, choose License Headers in Project
Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package ntnusubsea.gui;

import java.io.BufferedWriter;
import java.io.File;
import java.io.FileWriter;
import java.time.LocalDateTime;
import java.util.Observable;
import java.util.Observer;

/**
 * This class creates a text file and logs the data every second. The text
file
 * contains time stamps and the desired data.
 *
 * @author Marius Nonsvik
 */
public class DataLogger implements Observer, Runnable {

    private String dataString = " NaN";
    private Data data;
    private LocalDateTime now;
    File file = new File("ROV Data Log.txt");

    /**
     * Creates an instance of DataLogger and saves the current timestamp
     *
     * @param data The Data object containing the data to be logged
     */
    public DataLogger(Data data) {
        now = LocalDateTime.now();
        this.data = data;
        newSession();
    }

    /**
```

```
     * Writes a string the the text file
     *
     * @param data String to write to the text file
     */
    public void writeToFile(String data) {
        BufferedWriter writer = null;
        try {
            writer = new BufferedWriter(new FileWriter(file, true));
            writer.write(data + System.getProperty("line.separator"));
        } catch (Exception e) {
            System.out.println(e.getMessage());
        } finally {
            try {
                writer.close();
            } catch (Exception e) {
                System.out.println(e.getMessage());
            }
        }
    }

    /**
     * Writes a header in the text file to indicate a new logging session
     */
    public void newSession() {
        String seperator = System.getProperty("line.separator");
        String startString = seperator + seperator + "----- New session!
-----" + seperator;
        writeToFile(startString);
        String day, month, year;
        if (now.getDayOfMonth() < 10) {
            day = "0" + Integer.toString(now.getDayOfMonth());
        } else {
            day = Integer.toString(now.getDayOfMonth());
        }
        if (now.getMonthValue() < 10) {
            month = "0" + Integer.toString(now.getMonthValue());
        } else {
            month = Integer.toString(now.getMonthValue());
        }
        year = Integer.toString(now.getYear());
        String date = day + "." + month + "." + year;
```

```
        writeToFile("-----" + date + "-----" + seperator);
        String channelLabels = "               ";
        dataString = " | Depth: " + String.valueOf(data.getDepth());
        channelLabels += fixedLengthString("| Depth", dataString.length());
        dataString += " | Heading: " + String.valueOf(data.getHeading());
        channelLabels += fixedLengthString("| Heading", dataString.length() +
15 - channelLabels.length());
        dataString += " | Latitude: " + String.valueOf(data.getLatitude());
        channelLabels += fixedLengthString("| Latitude", dataString.length() +
15 - channelLabels.length());
        dataString += " | Longitude: " + String.valueOf(data.getLongitude());
        channelLabels += fixedLengthString("| Longitude", dataString.length()
+ 15 - channelLabels.length());
        dataString += " | " + data.getChannel(1);
        channelLabels += fixedLengthString("| Channel 1", dataString.length()
+ 15 - channelLabels.length());
        dataString += " | " + data.getChannel(2);
        channelLabels += fixedLengthString("| Channel 2", dataString.length()
+ 15 - channelLabels.length());
        dataString += " | " + data.getChannel(3);
        channelLabels += fixedLengthString("| Channel 3", dataString.length()
+ 15 - channelLabels.length());
        dataString += " | " + data.getChannel(4);
        channelLabels += fixedLengthString("| Channel 4", dataString.length()
+ 15 - channelLabels.length());
        writeToFile(channelLabels);
    }

    @Override
    public void update(Observable o, Object arg) {
        dataString = " | Depth: " + String.valueOf(data.getDepth());
        dataString += " | Heading: " + String.valueOf(data.getHeading());
        dataString += " | Latitude: " + String.valueOf(data.getLatitude());
        dataString += " | Longitude: " + String.valueOf(data.getLongitude());
        dataString += " | " + data.getChannel(1);
        dataString += " | " + data.getChannel(2);
        dataString += " | " + data.getChannel(3);
        dataString += " | " + data.getChannel(4);
    }

    @Override
```

```
    public void run() {
        now = LocalDateTime.now();
        String second, minute, hour;
        if (now.getSecond() < 10) {
            second = "0" + Integer.toString(now.getSecond());
        } else {
            second = Integer.toString(now.getSecond());
        }
        if (now.getMinute() < 10) {
            minute = "0" + Integer.toString(now.getMinute());
        } else {
            minute = Integer.toString(now.getMinute());
        }
        if (now.getHour() < 10) {
            hour = "0" + Integer.toString(now.getHour());
        } else {
            hour = Integer.toString(now.getHour());
        }
        String time = hour + ":" + minute + ":" + second;
        writeToFile("Time: " + time + dataString);
    }

    /**
     * Returns a string at a fixed length
     * @param string String to change length of
     * @param length Wanted length of string
     * @return String at new fixed length
     */
    public static String fixedLengthString(String string, int length) {
        return String.format("%-" + length + "s", string);
    }

}
```

```
/*
 * To change this license header, choose License Headers in Project
Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package ntnusubsea.gui;

import java.awt.image.BufferedImage;
import java.io.BufferedReader;
import java.io.FileReader;
import java.io.IOException;
import java.util.ArrayList;
import java.util.Observable;
import java.util.logging.Level;
import java.util.logging.Logger;

/**
 * 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. Data is made thread safe by the use of
 * synchronized methods.
 *
 * @author Marius Nonsvik
 */
public final class Data extends Observable {

    private float depth = 0;
    private float seafloorRov = 0;
    private float seafloorBoat = 0;
    private float pitchAngle = 0;
    private float wingAngle = 0;
    private float rollAngle = 0;
    private float heading = 0;
    private float latitude = 0;
    private float longitude = 0;
    private float temperature = 0;
    private float speed;
    private float pressure = 0;
    private float channel1 = 0;
```

```
    private float channel2 = 0;
    private float channel3 = 0;
    private float channel4 = 0;
    private byte actuatorStatus = 0;
    private byte leakStatus = 0;
    private ArrayList<String> labels = new ArrayList();
    private float[] channelValues = new float[4];
    private String defaultIP = "";
    private BufferedImage videoImage;
    private String Kp;
    private String Ki;
    private String Kd;
    private long timer = System.currentTimeMillis();

    /**
     * Creates an object of the class Data.
     */
    public Data() {
        try {
            BufferedReader br = new BufferedReader(new FileReader("ROV
Options.txt"));
            defaultIP = br.readLine();
            labels.add(0, br.readLine());
            labels.add(1, br.readLine());
            labels.add(2, br.readLine());
            labels.add(3, br.readLine());
            labels.add(4, br.readLine());
            labels.add(5, br.readLine());
            labels.add(6, br.readLine());
            labels.add(7, br.readLine());
            Kp = br.readLine();
            Ki = br.readLine();
            Kd = br.readLine();
            channelValues[0] = channel1;
            channelValues[1] = channel2;
            channelValues[2] = channel3;
            channelValues[3] = channel4;
//            videoImage = ImageIO.read(new
File("C:\\Users\\marno\\OneDrive\\Documents\\NetBeansProjects\\NTNUSubsea
GUI\\src\\ntnusubsea\\gui\\Images\\rsz_rovside.png"));
        } catch (IOException ex) {
```

```
            Logger.getLogger(Data.class.getName()).log(Level.SEVERE, null,
ex);
        }

    }

    /**
     * Sets the default connection IP
     *
     * @param ip The default connection IP
     */
    public synchronized void setDefaultIP(String ip) {
        defaultIP = ip;
        setChanged();
        notifyObservers();
    }

    /**
     * Returns the default connection IP
     *
     * @return The default connection IP
     */
    public synchronized String getDefaultIP() {
        return defaultIP;
    }

    public synchronized String getKp() {
        return Kp;
    }

    public synchronized String getKi() {
        return Ki;
    }

    public synchronized String getKd() {
        return Kd;
    }

    /**
     * Sets the label of all the different I/O channels and notifies observers
     *
```

```
     * @param c1 Channel 1 label
     * @param c2 Channel 2 label
     * @param c3 Channel 3 label
     * @param c4 Channel 4 label
     * @param c5 Channel 5 label
     * @param c6 Channel 6 label
     * @param c7 Channel 7 label
     * @param c8 Channel 8 label
     */
    public synchronized void setIOLabels(String c1, String c2, String c3,
String c4, String c5, String c6, String c7, String c8) {
        labels.set(0, c1);
        labels.set(1, c2);
        labels.set(2, c3);
        labels.set(3, c4);
        labels.set(4, c5);
        labels.set(5, c6);
        labels.set(6, c7);
        labels.set(7, c8);
        setChanged();
        notifyObservers();
    }

    /**
     * Returns a string containing the label of the channel. If the channel is
     * an input, the string also contains its measure value. (Index 1-8)
     *
     * @param channel Index of channel to return
     * @return String containing label and value
     */
    public synchronized String getChannel(int channel) {
        if (channel > 0 && channel < 9) {
            if (channel < 5) {
                String channelString = labels.get(channel - 1) + ": ";
                channelString += channelValues[channel - 1];
                return channelString;
            } else {
                return labels.get(channel - 1);
            }
        } else {
            return null;
```

```
        }
    }

    /**
     * Returns the value of the channel as a float. (Index 1-4)
     *
     * @param channel Index of channel
     * @return Value of the channel as float
     */
    public synchronized float getChannelValue(int channel) {
        if (channel < 0 && channel > 5) {
            return channelValues[channel - 1];
        } else {
            return (float) 0.001;
        }
    }

    /**
     * Sets the value of one of the inputs and notifies observers (Index 1-4).
     *
     * @param value Value of the channel
     * @param channel Index of the channel
     */
    public synchronized void setChannel(float value, int channel) {
        if (channel < 0 && channel > 5) {
            channelValues[channel - 1] = value;
        }
        setChanged();
        notifyObservers();
    }

    /**
     * Updates the current pitch angle of the ROV and notifies observers
     *
     * @param angle Current pitch angle of the ROV
     */
    public synchronized void setPitchAngle(float angle) {
        pitchAngle = angle;
        setChanged();
        notifyObservers();
    }
```

```
    /**
     * Retrieves the current pitch angle
     *
     * @return Current pitch angle of the ROV
     */
    public synchronized float getPitchAngle() {
        return pitchAngle;
    }

    /**
     * Updates the current roll angle of the ROV
     *
     * @param angle Current roll angle of the ROV
     */
    public synchronized void setRollAngle(float angle) {
        rollAngle = angle;
        setChanged();
        notifyObservers();
    }

    /**
     * Retrieves the current roll angle
     *
     * @return Current roll angle of the ROV
     */
    public synchronized float getRollAngle() {
        return rollAngle;
    }

    /**
     * Updates the current wing angle of the ROV and notifies observers
     *
     * @param angle Current wing angle of the ROV
     */
    public synchronized void setWingAngle(float angle) {
        wingAngle = angle;
        setChanged();
        notifyObservers();
    }
```

```
    /**
     * Retrieves the current pitch angle
     *
     * @return Current wing angle of the ROV
     */
    public synchronized float getWingAngle() {
        return wingAngle;
    }

    /**
     * Updates the current heading of the ROV and notifies observers
     *
     * @param heading Current heading of the ROV
     */
    public synchronized void setHeading(float heading) {
        this.heading = heading;
        setChanged();
        notifyObservers();
    }

    /**
     * Retrieves the current heading
     *
     * @return Current heading of the ROV
     */
    public synchronized float getHeading() {
        return heading;
    }

    /**
     * Updates the current latitude of the ROV and notifies observers
     *
     * @param latitude Current latitude of the ROV
     */
    public synchronized void setLatitude(float latitude) {
        this.latitude = latitude;
        setChanged();
        notifyObservers();
    }

    /**
```

```
     * Retrieves the current latitude
     *
     * @return Current latitude of the ROV
     */
    public synchronized float getLatitude() {
        return latitude;
    }

    /**
     * Updates the current longitude of the ROV and notifies observers
     *
     * @param longitude Current longitude of the ROV
     */
    public synchronized void setLongitude(float longitude) {
        this.longitude = longitude;
        setChanged();
        notifyObservers();
    }

    /**
     * Retrieves the current longitude
     *
     * @return Current longitude of the ROV
     */
    public synchronized float getLongitude() {
        return longitude;
    }

    /**
     * Updates the current depth of the ROV and notifies observers
     *
     * @param depth Current depth of the ROV
     */
    public synchronized void setDepth(float depth) {
        this.depth = depth;
        setChanged();
        notifyObservers();
    }

    /**
     * Retrieves the current depth
```

```
     *
     * @return Current depth of the ROV
     */
    public synchronized float getDepth() {
        return depth;
    }

    /**
     * Updates the current depth beneath the ROV
     *
     * @param depth Depth beneath the ROV
     */
    public synchronized void setSeafloorRov(float depth) {
        seafloorRov = depth;
        setChanged();
        notifyObservers();
    }

    /**
     * Returns the depth beneath the ROV
     *
     * @return Depth beneath the ROV
     */
    public synchronized float getSeafloorRov() {
        return seafloorRov;
    }

    /**
     * Updates the current depth beneath the vessel
     *
     * @param depth Depth beneath the vessel
     */
    public synchronized void setSeafloorBoat(float depth) {
        seafloorBoat = depth;
        setChanged();
        notifyObservers();
    }

    /**
     * Returns the depth beneath the vessel
     *
```

```
     * @return Depth beneath the vessel
     */
    public synchronized float getSeafloorBoat() {
        return seafloorBoat;
    }

    /**
     * Updates the image of the video stream and notifies observers
     *
     * @param image New image in the video stream
     */
    public synchronized void setVideoImage(BufferedImage image) {
        videoImage = null;
        videoImage = image;
        setChanged();
        notifyObservers();
    }

    /**
     * Updates the status of the actuators. 1 if they are currently running
and
     * 0 if they are currently idle.
     *
     * @param status Current status of the actuators
     */
    public synchronized void setActuatorStatus(byte status) {
        this.actuatorStatus = status;
        setChanged();
        notifyObservers();
    }

    /**
     * Returns the status of the actuators. true if they are currently running
     * and false if they are currently idle.
     *
     * @return Current status of the actuators
     */
    public synchronized boolean getActuatorStatus() {
        if (actuatorStatus == 1) {
            return true;
        } else {
```

```
            return false;
        }
    }

    /**
     * Updates the leak status in the ROV. 1 if a leak is detected, 0 if no
leak
     * is detected.
     *
     * @param leak Current leak status of the ROV
     */
    public synchronized void setLeakStatus(byte leak) {
        leakStatus = leak;
        setChanged();
        notifyObservers();
    }

    /**
     * Returns the leak status of the ROV. Returns true if a leak is detected,
     * false if no leak is detected
     *
     * @return Current leak status of the ROV
     */
    public synchronized boolean getLeakStatus() {
        if (leakStatus == 1) {
            return true;
        } else {
            return false;
        }
    }

    /**
     * Updates the temperature of the water
     *
     * @param temp Temperature of the water
     */
    public synchronized void setTemperature(float temp) {
        temperature = temp;
        setChanged();
        notifyObservers();
    }
```

```
    /**
     * Returns the current temperature of the water
     *
     * @return Temperature of the water
     */
    public synchronized float getTemperature() {
        return temperature;
    }

    /**
     * Updates the pressure surrounding the ROV
     *
     * @param pres Pressure surrounding the ROV
     */
    public synchronized void setPressure(float pres) {
        pressure = pres;
        setChanged();
        notifyObservers();
    }

    /**
     * Returns the current pressure around the ROV
     *
     * @return Current pressure around the ROV
     */
    public synchronized float getPressure() {
        return pressure;
    }

    public synchronized void setSpeed(float speed) {
        this.speed = speed;
        setChanged();
        notifyObservers();
    }

    public synchronized float getSpeed() {
        return speed;
    }

    /**
```

```
     * Returns the current image in the video stream
     *
     * @return Current image in the video stream
     */
    public synchronized BufferedImage getVideoImage() {
        return videoImage;
    }
}
```

```
/*
 * To change this license header, choose License Headers in Project
Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package ntnusubsea.gui;

import com.fazecast.jSerialComm.SerialPort;
import com.fazecast.jSerialComm.SerialPortDataListener;
import com.fazecast.jSerialComm.SerialPortEvent;
import java.nio.ByteBuffer;
import java.nio.ByteOrder;

/**
 * Detects a connected com-port and reads incoming Nmea data.
 * @author Kristian Homdrom and Marius Nonsvik
 */
public class NmeaReceiver implements Runnable {

    private boolean connection;
    private SerialPort port;
    private Data data;

    public NmeaReceiver(Data data) {
        // Create new Arduino Serial object, used to cennect to arduino with
comport and baudrate set.
        //  ardConn = new Arduino(wantedPort, BAUD_RATE);
        if (SerialPort.getCommPorts().length > 0) {
            port = SerialPort.getCommPorts()[0];
            port.setBaudRate(115200);
        }
        this.data = data;
    }

    /**
     * open the serial connection to the arduino
     */
    public synchronized void openConnection() {
        try {
            if (!this.port.isOpen()) {
```

```
                this.port.openPort();
                connection = true;
                // if port failed to open get message
            }
        } catch (Exception e) {
            System.out.println(e.getMessage());
        }
    }

    /**
     * check if Serial port is open
     *
     * @return state of the serialport
     */
    public boolean getOpen() {
        return this.port.isOpen();
    }

    @Override
    public void run() {
        openConnection();
        if (getOpen());
        {
            //   port.getFlowControlSettings()
            port.addDataListener(new SerialPortDataListener() {
                @Override
                public int getListeningEvents() {
                    return SerialPort.LISTENING_EVENT_DATA_AVAILABLE;
                }

                @Override
                public void serialEvent(SerialPortEvent event) {
                    if (event.getEventType() !=
SerialPort.LISTENING_EVENT_DATA_AVAILABLE) {
                        return;
                    }
                    //    System.out.println("bytesav " +
port.bytesAvailable() );
                    if (port.bytesAvailable() > 6) {
                        byte[] newData = new byte[port.bytesAvailable()];
                        int numRead = port.readBytes(newData, newData.length);
```

```
                        System.out.println("numread " + numRead);
                        if (numRead == 24) {
                            getValues(newData);
                            System.out.println("tt");
                        } else {
                            port.readBytes(newData, port.bytesAvailable());
                        }

                    }

                }
            });

        }
    }

    public void getValues(byte[] b) {
        ByteBuffer bb = ByteBuffer.wrap(b);
        bb.order(ByteOrder.LITTLE_ENDIAN);
        int index = 0;
        try {
            data.setLatitude(roundToDecimalPlaces(bb.getFloat(4), 2));
            data.setLongitude(roundToDecimalPlaces(bb.getFloat(8), 2));
            data.setHeading(roundToDecimalPlaces(bb.getFloat(12), 2));
            data.setSeafloorRov(roundToDecimalPlaces(bb.getFloat(16), 2));
            data.setSpeed(roundToDecimalPlaces(bb.getFloat(20), 2));
        } catch (Exception e) {

        }
    }

    private float roundToDecimalPlaces(float value, int decimalPlaces) {
        float shift = (float) Math.pow(10, decimalPlaces);
        return Math.round(value * shift) / shift;
    }
}
```

```
/*
 * To change this license header, choose License Headers in Project
Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package ntnusubsea.gui;

import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.File;
import java.io.FileReader;
import java.io.FileWriter;
import java.util.ArrayList;

/**
 * This class allows the user to change options between every time the
program
 * is loaded. It reads and writes from the ROV Options text file from the
program
 * folder
 *
 * @author Marius Nonsvik
 */
public class OptionsFrame extends javax.swing.JFrame {

    File file = new File("ROV Options.txt");
    String defaultIP;
    ArrayList channels = new ArrayList<String>();
    private Data data;

    /**
     * Creates new form OptionsFrame
     * @param data Data containing shared variables.
     */
    public OptionsFrame(Data data) {
        initComponents();
        this.data = data;
        getOptionsFromFile();
    }
```

```
    /**
     * 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() {

        jLabelIPHeader = new javax.swing.JLabel();
        jButtonOK = new javax.swing.JButton();
        jButtonApply = new javax.swing.JButton();
        jButtonCanel = new javax.swing.JButton();
        jTextFieldIP = new javax.swing.JTextField();
        jSeparator1 = new javax.swing.JSeparator();
        jLabelChannelsHeader = new javax.swing.JLabel();
        jLabel1 = new javax.swing.JLabel();
        jLabel2 = new javax.swing.JLabel();
        jLabel3 = new javax.swing.JLabel();
        jLabel4 = new javax.swing.JLabel();
        jLabel5 = new javax.swing.JLabel();
        jLabel6 = new javax.swing.JLabel();
        jLabel7 = new javax.swing.JLabel();
        jLabel8 = new javax.swing.JLabel();
        jTextFieldChannel1 = new javax.swing.JTextField();
        jTextFieldChannel2 = new javax.swing.JTextField();
        jTextFieldChannel4 = new javax.swing.JTextField();
        jTextFieldChannel3 = new javax.swing.JTextField();
        jTextFieldChannel5 = new javax.swing.JTextField();
        jTextFieldChannel6 = new javax.swing.JTextField();
        jTextFieldChannel7 = new javax.swing.JTextField();
        jTextFieldChannel8 = new javax.swing.JTextField();

        setDefaultCloseOperation(javax.swing.WindowConstants.DISPOSE_ON_CLOSE)
;
        setTitle("Options");
        setResizable(false);

        jLabelIPHeader.setFont(new java.awt.Font("Tahoma", 0, 14)); // NOI18N
        jLabelIPHeader.setText("Default IP");
```

```
        jButtonOK.setText("OK");
        jButtonOK.setPreferredSize(new java.awt.Dimension(65, 25));
        jButtonOK.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jButtonOKActionPerformed(evt);
            }
        });

        jButtonApply.setText("Apply");
        jButtonApply.setPreferredSize(new java.awt.Dimension(65, 25));
        jButtonApply.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jButtonApplyActionPerformed(evt);
            }
        });

        jButtonCanel.setText("Cancel");
        jButtonCanel.setPreferredSize(new java.awt.Dimension(65, 25));
        jButtonCanel.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jButtonCanelActionPerformed(evt);
            }
        });

        jTextFieldIP.setText("123.123.123.123");
        jTextFieldIP.setToolTipText("");

        jLabelChannelsHeader.setFont(new java.awt.Font("Tahoma", 0, 14)); //
NOI18N
        jLabelChannelsHeader.setText("Custom I/O");

        jLabel1.setText("<html>Channel 1<br/>Intput");

        jLabel2.setText("<html>Channel 2<br/>Input");

        jLabel3.setText("<html>Channel 5<br/>Output");

        jLabel4.setText("<html>Channel 3<br/>Input");

        jLabel5.setText("<html>Channel 4<br/>Input");
```

```
        jLabel6.setText("<html>Channel 7<br/>Output");

        jLabel7.setText("<html>Channel 6<br/>Output");

        jLabel8.setText("<html>Channel 8<br/>Output");

        jTextFieldChannel1.setText("Channel 1");

        jTextFieldChannel2.setText("Channel 2");

        jTextFieldChannel4.setText("Channel 4");

        jTextFieldChannel3.setText("Channel 3");

        jTextFieldChannel5.setText("Channel 5");
        jTextFieldChannel5.addActionListener(new
java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jTextFieldChannel5ActionPerformed(evt);
            }
        });

        jTextFieldChannel6.setText("Channel 6");
        jTextFieldChannel6.addActionListener(new
java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jTextFieldChannel6ActionPerformed(evt);
            }
        });

        jTextFieldChannel7.setText("Channel 7");

        jTextFieldChannel8.setText("Channel 8");

        javax.swing.GroupLayout layout = new
javax.swing.GroupLayout(getContentPane());
        getContentPane().setLayout(layout);
        layout.setHorizontalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADI
NG)
```

```
            .addGroup(layout.createSequentialGroup()
                .addContainerGap()
                .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.
Alignment.LEADING)
                    .addComponent(jSeparator1)
                    .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
layout.createSequentialGroup()
                        .addGap(0, 0, Short.MAX_VALUE)
                        .addComponent(jButtonOK,
javax.swing.GroupLayout.PREFERRED_SIZE, 75,
javax.swing.GroupLayout.PREFERRED_SIZE)
                        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlac
ement.RELATED)
                        .addComponent(jButtonApply,
javax.swing.GroupLayout.PREFERRED_SIZE, 75,
javax.swing.GroupLayout.PREFERRED_SIZE)
                        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlac
ement.RELATED)
                        .addComponent(jButtonCanel,
javax.swing.GroupLayout.PREFERRED_SIZE, 75,
javax.swing.GroupLayout.PREFERRED_SIZE))
                    .addGroup(layout.createSequentialGroup()
                        .addComponent(jTextFieldChannel5,
javax.swing.GroupLayout.PREFERRED_SIZE, 100,
javax.swing.GroupLayout.PREFERRED_SIZE)
                        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlac
ement.RELATED)
                        .addComponent(jTextFieldChannel6,
javax.swing.GroupLayout.DEFAULT_SIZE, 100, Short.MAX_VALUE)
                        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlac
ement.RELATED)
                        .addComponent(jTextFieldChannel7,
javax.swing.GroupLayout.DEFAULT_SIZE, 100, Short.MAX_VALUE)
                        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlac
ement.RELATED)
                        .addComponent(jTextFieldChannel8,
javax.swing.GroupLayout.DEFAULT_SIZE, 100, Short.MAX_VALUE))
                    .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
layout.createSequentialGroup()
                        .addGroup(layout.createParallelGroup(javax.swing.Group
Layout.Alignment.LEADING)
```

```
                            .addGroup(javax.swing.GroupLayout.Alignment.TRAILI
NG, layout.createSequentialGroup()
                                .addGroup(layout.createParallelGroup(javax.swi
ng.GroupLayout.Alignment.LEADING)
                                    .addComponent(jTextFieldIP,
javax.swing.GroupLayout.PREFERRED_SIZE, 140,
javax.swing.GroupLayout.PREFERRED_SIZE)
                                    .addComponent(jLabelIPHeader,
javax.swing.GroupLayout.PREFERRED_SIZE, 92,
javax.swing.GroupLayout.PREFERRED_SIZE)
                                    .addComponent(jLabelChannelsHeader,
javax.swing.GroupLayout.PREFERRED_SIZE, 80,
javax.swing.GroupLayout.PREFERRED_SIZE))
                                .addGap(0, 0, Short.MAX_VALUE))
                            .addGroup(javax.swing.GroupLayout.Alignment.TRAILI
NG, layout.createSequentialGroup()
                                .addGroup(layout.createParallelGroup(javax.swi
ng.GroupLayout.Alignment.TRAILING, false)
                                    .addComponent(jLabel1,
javax.swing.GroupLayout.DEFAULT_SIZE, 100, Short.MAX_VALUE)
                                    .addComponent(jLabel3)
                                    .addComponent(jTextFieldChannel1))
                                .addPreferredGap(javax.swing.LayoutStyle.Compo
nentPlacement.RELATED)
                                .addGroup(layout.createParallelGroup(javax.swi
ng.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.Compo
nentPlacement.RELATED)
                                .addGroup(layout.createParallelGroup(javax.swi
ng.GroupLayout.Alignment.TRAILING, false)
                                    .addComponent(jLabel6,
javax.swing.GroupLayout.DEFAULT_SIZE, 100, Short.MAX_VALUE)
                                    .addComponent(jLabel4)
                                    .addComponent(jTextFieldChannel3))))
                        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlac
```

```
ement.RELATED)
                        .addGroup(layout.createParallelGroup(javax.swing.Group
Layout.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())
        );
        layout.setVerticalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADI
NG)
            .addGroup(layout.createSequentialGroup()
                .addContainerGap()
                .addComponent(jLabelIPHeader,
javax.swing.GroupLayout.PREFERRED_SIZE, 24,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RE
LATED)
                .addComponent(jTextFieldIP,
javax.swing.GroupLayout.PREFERRED_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UN
RELATED)
                .addComponent(jSeparator1,
javax.swing.GroupLayout.PREFERRED_SIZE, 10,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RE
LATED)
                .addComponent(jLabelChannelsHeader,
javax.swing.GroupLayout.PREFERRED_SIZE, 28,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RE
LATED)
                .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.
Alignment.BASELINE)
                    .addComponent(jLabel1,
javax.swing.GroupLayout.PREFERRED_SIZE, 30,
javax.swing.GroupLayout.PREFERRED_SIZE)
```

```
                    .addComponent(jLabel2,
javax.swing.GroupLayout.PREFERRED_SIZE, 30,
javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addComponent(jLabel4,
javax.swing.GroupLayout.PREFERRED_SIZE, 30,
javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addComponent(jLabel5,
javax.swing.GroupLayout.PREFERRED_SIZE, 30,
javax.swing.GroupLayout.PREFERRED_SIZE))
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RE
LATED)
                .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.
Alignment.BASELINE)
                    .addComponent(jTextFieldChannel1,
javax.swing.GroupLayout.PREFERRED_SIZE, 32,
javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addComponent(jTextFieldChannel2,
javax.swing.GroupLayout.PREFERRED_SIZE, 32,
javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addComponent(jTextFieldChannel4,
javax.swing.GroupLayout.PREFERRED_SIZE, 32,
javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addComponent(jTextFieldChannel3,
javax.swing.GroupLayout.PREFERRED_SIZE, 32,
javax.swing.GroupLayout.PREFERRED_SIZE))
                .addGap(47, 47, 47)
                .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.
Alignment.BASELINE)
                    .addComponent(jLabel3,
javax.swing.GroupLayout.PREFERRED_SIZE, 30,
javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addComponent(jLabel7,
javax.swing.GroupLayout.PREFERRED_SIZE, 30,
javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addComponent(jLabel6,
javax.swing.GroupLayout.PREFERRED_SIZE, 30,
javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addComponent(jLabel8,
javax.swing.GroupLayout.PREFERRED_SIZE, 30,
javax.swing.GroupLayout.PREFERRED_SIZE))
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RE
```

```
LATED)
                .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.
Alignment.BASELINE)
                    .addComponent(jTextFieldChannel5,
javax.swing.GroupLayout.PREFERRED_SIZE, 32,
javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addComponent(jTextFieldChannel6,
javax.swing.GroupLayout.PREFERRED_SIZE, 32,
javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addComponent(jTextFieldChannel7,
javax.swing.GroupLayout.PREFERRED_SIZE, 32,
javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addComponent(jTextFieldChannel8,
javax.swing.GroupLayout.PREFERRED_SIZE, 32,
javax.swing.GroupLayout.PREFERRED_SIZE))
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RE
LATED, 59, Short.MAX_VALUE)
                .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.
Alignment.BASELINE)
                    .addComponent(jButtonOK,
javax.swing.GroupLayout.PREFERRED_SIZE, 25,
javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addComponent(jButtonApply,
javax.swing.GroupLayout.PREFERRED_SIZE, 25,
javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addComponent(jButtonCanel,
javax.swing.GroupLayout.PREFERRED_SIZE, 25,
javax.swing.GroupLayout.PREFERRED_SIZE))
                .addContainerGap())
        );

        pack();
    }// </editor-fold>

    private void jButtonOKActionPerformed(java.awt.event.ActionEvent evt) {
        BufferedWriter writer = null;
        try {
            writer = new BufferedWriter(new FileWriter(file, false));
            writer.write(jTextFieldIP.getText() +
System.getProperty("line.separator"));
            writer.write(jTextFieldChannel1.getText() +
```

```
System.getProperty("line.separator"));
            writer.write(jTextFieldChannel2.getText() +
System.getProperty("line.separator"));
            writer.write(jTextFieldChannel3.getText() +
System.getProperty("line.separator"));
            writer.write(jTextFieldChannel4.getText() +
System.getProperty("line.separator"));
            writer.write(jTextFieldChannel5.getText() +
System.getProperty("line.separator"));
            writer.write(jTextFieldChannel6.getText() +
System.getProperty("line.separator"));
            writer.write(jTextFieldChannel7.getText() +
System.getProperty("line.separator"));
            writer.write(jTextFieldChannel8.getText() +
System.getProperty("line.separator"));
            data.setDefaultIP(jTextFieldIP.getText());
            data.setIOLabels(jTextFieldChannel1.getText(),
jTextFieldChannel2.getText(), jTextFieldChannel3.getText(),
jTextFieldChannel4.getText(), jTextFieldChannel5.getText(),
jTextFieldChannel6.getText(), jTextFieldChannel7.getText(),
jTextFieldChannel8.getText());
        } catch (Exception e) {
            System.out.println(e.getMessage());
        } finally {
            try {
                writer.close();
            } catch (Exception e) {
                System.out.println(e.getMessage());
            }
        }
        this.dispose();
    }

    private void jButtonApplyActionPerformed(java.awt.event.ActionEvent evt) {
        BufferedWriter writer = null;
        try {
            writer = new BufferedWriter(new FileWriter(file, false));
            writer.write(jTextFieldIP.getText() +
System.getProperty("line.separator"));
            writer.write(jTextFieldChannel1.getText() +
System.getProperty("line.separator"));
```

```
            writer.write(jTextFieldChannel2.getText() +
System.getProperty("line.separator"));
            writer.write(jTextFieldChannel3.getText() +
System.getProperty("line.separator"));
            writer.write(jTextFieldChannel4.getText() +
System.getProperty("line.separator"));
            writer.write(jTextFieldChannel5.getText() +
System.getProperty("line.separator"));
            writer.write(jTextFieldChannel6.getText() +
System.getProperty("line.separator"));
            writer.write(jTextFieldChannel7.getText() +
System.getProperty("line.separator"));
            writer.write(jTextFieldChannel8.getText() +
System.getProperty("line.separator"));
            data.setDefaultIP(jTextFieldIP.getText());
            data.setIOLabels(jTextFieldChannel1.getText(),
jTextFieldChannel2.getText(), jTextFieldChannel3.getText(),
jTextFieldChannel4.getText(), jTextFieldChannel5.getText(),
jTextFieldChannel6.getText(), jTextFieldChannel7.getText(),
jTextFieldChannel8.getText());

        } catch (Exception e) {
            System.out.println(e.getMessage());
        } finally {
            try {
                writer.close();
            } catch (Exception e) {
                System.out.println(e.getMessage());
            }
        }
    }

    /**
     * Reads the option files and displays the current settings
     */
    public void getOptionsFromFile() {
        try (BufferedReader br = new BufferedReader(new FileReader("ROV
Options.txt"))) {
            defaultIP = br.readLine();
            jTextFieldIP.setText(defaultIP);
            for (int i = 0; i < 8; i++) {
```

```
                channels.add(i, br.readLine());
            }
            jTextFieldChannel1.setText((String) channels.get(0));
            jTextFieldChannel2.setText((String) channels.get(1));
            jTextFieldChannel3.setText((String) channels.get(2));
            jTextFieldChannel4.setText((String) channels.get(3));
            jTextFieldChannel5.setText((String) channels.get(4));
            jTextFieldChannel6.setText((String) channels.get(5));
            jTextFieldChannel7.setText((String) channels.get(6));
            jTextFieldChannel8.setText((String) channels.get(7));
        } catch (Exception e) {
            System.out.println(e.getMessage());
        }
    }

    private void jButtonCanelActionPerformed(java.awt.event.ActionEvent evt) {
        this.dispose();
    }

    private void jTextFieldChannel5ActionPerformed(java.awt.event.ActionEvent
evt) {
        // TODO add your handling code here:
    }

    private void jTextFieldChannel6ActionPerformed(java.awt.event.ActionEvent
evt) {
        // TODO add your handling code here:
    }


    // Variables declaration - do not modify
    private javax.swing.JButton jButtonApply;
    private javax.swing.JButton jButtonCanel;
    private javax.swing.JButton jButtonOK;
    private javax.swing.JLabel jLabel1;
    private javax.swing.JLabel jLabel2;
    private javax.swing.JLabel jLabel3;
    private javax.swing.JLabel jLabel4;
    private javax.swing.JLabel jLabel5;
    private javax.swing.JLabel jLabel6;
    private javax.swing.JLabel jLabel7;
```

```
    private javax.swing.JLabel jLabel8;
    private javax.swing.JLabel jLabelChannelsHeader;
    private javax.swing.JLabel jLabelIPHeader;
    private javax.swing.JSeparator jSeparator1;
    private javax.swing.JTextField jTextFieldChannel1;
    private javax.swing.JTextField jTextFieldChannel2;
    private javax.swing.JTextField jTextFieldChannel3;
    private javax.swing.JTextField jTextFieldChannel4;
    private javax.swing.JTextField jTextFieldChannel5;
    private javax.swing.JTextField jTextFieldChannel6;
    private javax.swing.JTextField jTextFieldChannel7;
    private javax.swing.JTextField jTextFieldChannel8;
    private javax.swing.JTextField jTextFieldIP;
    // End of variables declaration
}
```

```
/*
 * To change this license header, choose License Headers in Project
Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package ntnusubsea.gui;

import java.awt.Color;
import java.awt.GraphicsEnvironment;
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.ParseException;
import java.util.Observable;
import java.util.Observer;
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.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.
 *
 * @author Marius Nonsvik
 */
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 Double previousSetpoint = 0.00;
    private Double redoSetpoint = 0.00;
    private int mode = 0;
    private EchoSounderFrame echoSounder;
    private TCPClient client;
    private IOControlFrame io;
    private final String LIGHTSID = "<Lights>";
    private final String MODEID = "<Mode>";
    private final String SETPOINTID = "<Setpoint>";

    /**
     * Creates new form ROVFrame
     *
     * @param echoSounder Echo sounder frame to show graphs
     * @param data Data containing shared variables
     * @param client TCP client to send commands and receive data
     * @param io I/O frame to control inputs and outputs
     */
    public ROVFrame(EchoSounderFrame echoSounder, Data data, TCPClient client,
IOControlFrame io) {
        initComponents();
        this.echoSounder = echoSounder;
        this.data = data;
        this.client = client;
        this.io = io;
        videoSheet = new ImagePanel(cameraPanel.getWidth(),
cameraPanel.getHeight());
        videoSheet.setBackground(cameraPanel.getBackground());
        fullscreenVideoSheet = new ImagePanel(cameraPanel1.getWidth(),
cameraPanel1.getHeight());
        fullscreenVideoSheet.setBackground(cameraPanel1.getBackground());
        videoSheet.setOpaque(false);
        fullscreenVideoSheet.setOpaque(false);
        cameraPanel.add(videoSheet);
        cameraPanel1.add(fullscreenVideoSheet);
        setpointLabel.setText("Current setpoint: " + setpoint + "m");
        exitFullscreenButton.getInputMap().put(KeyStroke.getKeyStroke(KeyEvent
.VK_ESCAPE, 0), "exitFullscreen");
        depthInputTextField.getInputMap().put(KeyStroke.getKeyStroke(KeyEvent.
```

```
VK_ENTER, 0), "sendInput");
        exitFullscreenButton.getActionMap().put("exitFullscreen",
exitFullscreenAction);
        depthInputTextField.getActionMap().put("sendInput", sendInputAction);
    }

    /**
     * Calls the paintSheet method which updates the diplayed image.
     *
     * @param image The image to be displayed on the GUI
     */
    public void showImage(BufferedImage image) {
        if (fullscreen.isVisible()) {
            fullscreenVideoSheet.setSize(cameraPanel1.getSize());
            fullscreenVideoSheet.paintSheet(ImageUtils.resize(image,
fullscreenVideoSheet.getParent().getWidth(),
fullscreenVideoSheet.getParent().getHeight()));
        } else {
            videoSheet.paintSheet(ImageUtils.resize(image,
videoSheet.getParent().getWidth(), videoSheet.getParent().getHeight()));
            videoSheet.setSize(cameraPanel.getSize());

        }
    }

    /**
     * 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() {

        fullscreen = new javax.swing.JFrame();
        cameraPanel1 = new javax.swing.JPanel();
        exitFullscreenButton = new javax.swing.JButton();
        buttonGroup1 = new javax.swing.ButtonGroup();
        helpframe = new javax.swing.JFrame();
        jButton1 = new javax.swing.JButton();
```

```
        jLabel1 = new javax.swing.JLabel();
        background = new javax.swing.JPanel();
        cameraPanel = new javax.swing.JPanel();
        fullscreenButton = new javax.swing.JButton();
        controlPanel = new javax.swing.JPanel();
        depthPanel = new javax.swing.JPanel();
        sendButton = new javax.swing.JButton();
        depthInputTextField = new javax.swing.JFormattedTextField();
        depthHeader = new javax.swing.JLabel();
        jSeparator4 = new javax.swing.JSeparator();
        depthModeButton = new javax.swing.JRadioButton();
        seafloorModeButton = new javax.swing.JRadioButton();
        setpointLabel = new javax.swing.JLabel();
        lightPanel = new javax.swing.JPanel();
        lightHeader = new javax.swing.JLabel();
        jSeparator1 = new javax.swing.JSeparator();
        jSeparator5 = new javax.swing.JSeparator();
        lightIndicator = new javax.swing.JLabel();
        lightSwitch = new javax.swing.JToggleButton();
        emergencyPanel = new javax.swing.JPanel();
        emergencyHeader = new javax.swing.JLabel();
        emergencyStopButton = new javax.swing.JButton();
        jSeparator6 = new javax.swing.JSeparator();
        positionPanel = new javax.swing.JPanel();
        positionHeader = new javax.swing.JLabel();
        jSeparator7 = new javax.swing.JSeparator();
        headingLabel = new javax.swing.JLabel();
        latitudeLabel = new javax.swing.JLabel();
        longitudeLabel = new javax.swing.JLabel();
        jSeparator2 = new javax.swing.JSeparator();
        jSeparator3 = new javax.swing.JSeparator();
        infoPanel = new javax.swing.JPanel();
        pitchLabel = new javax.swing.JLabel();
        seafloorDepthRovLabel = new javax.swing.JLabel();
        rovDepthLabel = new javax.swing.JLabel();
        wingLabel = new javax.swing.JLabel();
        actuatorPanel1 = new javax.swing.JPanel();
        actuatorHeader1 = new javax.swing.JLabel();
        actuatorDutyCycleBar1 = new javax.swing.JProgressBar();
        warningLabel1 = new javax.swing.JLabel();
        actuatorPanel2 = new javax.swing.JPanel();
```

```
        actuatorHeader2 = new javax.swing.JLabel();
        actuatorDutyCycleBar2 = new javax.swing.JProgressBar();
        warningLabel2 = new javax.swing.JLabel();
        seafloorDepthBoatLabel = new javax.swing.JLabel();
        rollLabel = new javax.swing.JLabel();
        leakLabel = new javax.swing.JLabel();
        jMenuBar = new javax.swing.JMenuBar();
        jMenuFile = new javax.swing.JMenu();
        jMenuItemConnect = new javax.swing.JMenuItem();
        jMenuItemDisconnect = new javax.swing.JMenuItem();
        jMenuItemExit = new javax.swing.JMenuItem();
        jMenuEdit = new javax.swing.JMenu();
        jMenuUndo = new javax.swing.JMenuItem();
        jMenuRedo = new javax.swing.JMenuItem();
        jMenuTools = new javax.swing.JMenu();
        jMenuEchosounder = new javax.swing.JMenuItem();
        jMenuIOController = new javax.swing.JMenuItem();
        jMenuOptions = new javax.swing.JMenuItem();
        jMenuHelp = new javax.swing.JMenu();
        jMenuAbout = new javax.swing.JMenuItem();

        fullscreen.setDefaultCloseOperation(javax.swing.WindowConstants.DISPOS
E_ON_CLOSE);
        fullscreen.setFocusTraversalPolicyProvider(true);
        fullscreen.setLocationByPlatform(true);
        fullscreen.setUndecorated(true);
        fullscreen.addKeyListener(new java.awt.event.KeyAdapter() {
            public void keyPressed(java.awt.event.KeyEvent evt) {
                fullscreenKeyPressed(evt);
            }
        });

        cameraPanel1.setBackground(new java.awt.Color(0, 0, 0));
        cameraPanel1.setForeground(new java.awt.Color(255, 255, 255));
        cameraPanel1.setToolTipText("");
        cameraPanel1.setMinimumSize(new java.awt.Dimension(450, 320));
        cameraPanel1.setPreferredSize(new java.awt.Dimension(718, 580));

        exitFullscreenButton.setBackground(new java.awt.Color(0, 0, 0));
        exitFullscreenButton.setIcon(new
javax.swing.ImageIcon(getClass().getResource("/ntnusubsea/gui/Images/exitfulls
```

```
creenbutton.png"))); // NOI18N
        exitFullscreenButton.setBorder(null);
        exitFullscreenButton.setContentAreaFilled(false);
        exitFullscreenButton.setMinimumSize(new java.awt.Dimension(30, 30));
        exitFullscreenButton.addActionListener(new
java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                exitFullscreenButtonActionPerformed(evt);
            }
        });

        javax.swing.GroupLayout cameraPanel1Layout = new
javax.swing.GroupLayout(cameraPanel1);
        cameraPanel1.setLayout(cameraPanel1Layout);
        cameraPanel1Layout.setHorizontalGroup(
            cameraPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Ali
gnment.LEADING)
            .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
cameraPanel1Layout.createSequentialGroup()
                .addGap(0, 708, Short.MAX_VALUE)
                .addComponent(exitFullscreenButton,
javax.swing.GroupLayout.PREFERRED_SIZE, 30,
javax.swing.GroupLayout.PREFERRED_SIZE))
        );
        cameraPanel1Layout.setVerticalGroup(
            cameraPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Ali
gnment.LEADING)
            .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
cameraPanel1Layout.createSequentialGroup()
                .addGap(0, 572, Short.MAX_VALUE)
                .addComponent(exitFullscreenButton,
javax.swing.GroupLayout.PREFERRED_SIZE, 30,
javax.swing.GroupLayout.PREFERRED_SIZE))
        );

        javax.swing.GroupLayout fullscreenLayout = new
javax.swing.GroupLayout(fullscreen.getContentPane());
        fullscreen.getContentPane().setLayout(fullscreenLayout);
        fullscreenLayout.setHorizontalGroup(
            fullscreenLayout.createParallelGroup(javax.swing.GroupLayout.Align
ment.LEADING)
```

```
            .addComponent(cameraPanel1,
javax.swing.GroupLayout.Alignment.TRAILING,
javax.swing.GroupLayout.DEFAULT_SIZE, 738, Short.MAX_VALUE)
        );
        fullscreenLayout.setVerticalGroup(
            fullscreenLayout.createParallelGroup(javax.swing.GroupLayout.Align
ment.LEADING)
            .addComponent(cameraPanel1,
javax.swing.GroupLayout.Alignment.TRAILING,
javax.swing.GroupLayout.DEFAULT_SIZE, 602, Short.MAX_VALUE)
        );

        helpframe.setDefaultCloseOperation(javax.swing.WindowConstants.DISPOSE
_ON_CLOSE);
        helpframe.setTitle("About");
        helpframe.setType(java.awt.Window.Type.POPUP);

        jButton1.setText("Close");
        jButton1.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jButton1ActionPerformed(evt);
            }
        });

        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 vessel and act as a multi-sensor platform, were it shall be easy to
place new sesnors. There will also be a video stream from the ROV.
<br/><br/>The system consists of a Raspberry Pi on the ROV that is connected
to several Arduino microcontrollers that are connected to sensors and and
programed with the control system. The external computer which is on the
surface vessel is connected to a GPS, echo sounder and the RBPi. It will
present and save data in addition to handle user commands.<br/><br/> Created
by Marius Nonsvik, Kristian S Homdrom, Morten L Solli and Daniel
Reite.<html>");
        jLabel1.setToolTipText("");

        javax.swing.GroupLayout helpframeLayout = new
javax.swing.GroupLayout(helpframe.getContentPane());
        helpframe.getContentPane().setLayout(helpframeLayout);
        helpframeLayout.setHorizontalGroup(
```

```
            helpframeLayout.createParallelGroup(javax.swing.GroupLayout.Align
ment.LEADING)
            .addGroup(helpframeLayout.createSequentialGroup()
                .addContainerGap()
                .addGroup(helpframeLayout.createParallelGroup(javax.swing.Grou
pLayout.Alignment.LEADING)
                    .addGroup(helpframeLayout.createSequentialGroup()
                        .addGap(0, 160, Short.MAX_VALUE)
                        .addComponent(jButton1)
                        .addGap(0, 161, Short.MAX_VALUE))
                    .addComponent(jLabel1,
javax.swing.GroupLayout.PREFERRED_SIZE, 0, Short.MAX_VALUE))
                .addContainerGap())
        );
        helpframeLayout.setVerticalGroup(
            helpframeLayout.createParallelGroup(javax.swing.GroupLayout.Align
ment.LEADING)
            .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
helpframeLayout.createSequentialGroup()
                .addComponent(jLabel1, javax.swing.GroupLayout.DEFAULT_SIZE,
260, Short.MAX_VALUE)
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RE
LATED)
                .addComponent(jButton1)
                .addContainerGap())
        );

        setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);
        setTitle("Towed ROV");
        setAutoRequestFocus(false);
        setBackground(new java.awt.Color(255, 255, 255));
        setCursor(new java.awt.Cursor(java.awt.Cursor.DEFAULT_CURSOR));
        setMinimumSize(new java.awt.Dimension(600, 480));

        background.setBackground(new java.awt.Color(28, 28, 28));
        background.setForeground(new java.awt.Color(255, 255, 255));
        background.setToolTipText("");
        background.setMinimumSize(new java.awt.Dimension(600, 480));

        cameraPanel.setBackground(new java.awt.Color(0, 0, 0));
        cameraPanel.setForeground(new java.awt.Color(255, 255, 255));
```

```
        cameraPanel.setToolTipText("");
        cameraPanel.setAlignmentX(0.8F);
        cameraPanel.setAlignmentY(0.8F);
        cameraPanel.setMinimumSize(new java.awt.Dimension(450, 320));
        cameraPanel.setPreferredSize(new java.awt.Dimension(718, 580));

        fullscreenButton.setBackground(new java.awt.Color(0, 0, 0));
        fullscreenButton.setForeground(new java.awt.Color(255, 255, 255));
        fullscreenButton.setIcon(new
javax.swing.ImageIcon(getClass().getResource("/ntnusubsea/gui/Images/Fullscree
nbutton.png"))); // NOI18N
        fullscreenButton.setBorder(null);
        fullscreenButton.setContentAreaFilled(false);
        fullscreenButton.setFocusable(false);
        fullscreenButton.setHideActionText(true);
        fullscreenButton.setName(""); // NOI18N
        fullscreenButton.setPreferredSize(new java.awt.Dimension(30, 30));
        fullscreenButton.addActionListener(new java.awt.event.ActionListener()
{
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                fullscreenButtonActionPerformed(evt);
            }
        });

        javax.swing.GroupLayout cameraPanelLayout = new
javax.swing.GroupLayout(cameraPanel);
        cameraPanel.setLayout(cameraPanelLayout);
        cameraPanelLayout.setHorizontalGroup(
            cameraPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alig
nment.LEADING)
            .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
cameraPanelLayout.createSequentialGroup()
                .addGap(0, 664, Short.MAX_VALUE)
                .addComponent(fullscreenButton,
javax.swing.GroupLayout.PREFERRED_SIZE, 30,
javax.swing.GroupLayout.PREFERRED_SIZE))
        );
        cameraPanelLayout.setVerticalGroup(
            cameraPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alig
nment.LEADING)
            .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
```

```
cameraPanelLayout.createSequentialGroup()
                .addContainerGap(567, Short.MAX_VALUE)
                .addComponent(fullscreenButton,
javax.swing.GroupLayout.PREFERRED_SIZE, 30,
javax.swing.GroupLayout.PREFERRED_SIZE))
        );

        controlPanel.setBackground(new java.awt.Color(28, 28, 28));
        controlPanel.setMinimumSize(new java.awt.Dimension(150, 140));
        controlPanel.setPreferredSize(new java.awt.Dimension(768, 190));

        depthPanel.setBackground(new java.awt.Color(28, 28, 28));

        sendButton.setBackground(new java.awt.Color(255, 255, 255));
        sendButton.setFont(new java.awt.Font("Tahoma", 0, 12)); // NOI18N
        sendButton.setText("Send");
        sendButton.setEnabled(false);
        sendButton.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                sendButtonActionPerformed(evt);
            }
        });

        depthInputTextField.setFormatterFactory(new
javax.swing.text.DefaultFormatterFactory(new
javax.swing.text.NumberFormatter(new java.text.DecimalFormat("#.000"))));
        depthInputTextField.setToolTipText("Depth (Meters)");

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

        depthModeButton.setBackground(new java.awt.Color(28, 28, 28));
        buttonGroup1.add(depthModeButton);
        depthModeButton.setForeground(new java.awt.Color(255, 255, 255));
        depthModeButton.setSelected(true);
        depthModeButton.setText("Depth");
        depthModeButton.addActionListener(new java.awt.event.ActionListener()
{
            public void actionPerformed(java.awt.event.ActionEvent evt) {
```

```
                depthModeButtonActionPerformed(evt);
            }
        });

        seafloorModeButton.setBackground(new java.awt.Color(28, 28, 28));
        buttonGroup1.add(seafloorModeButton);
        seafloorModeButton.setForeground(new java.awt.Color(255, 255, 255));
        seafloorModeButton.setText("Distance from seafloor");
        seafloorModeButton.addActionListener(new
java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                seafloorModeButtonActionPerformed(evt);
            }
        });

        setpointLabel.setBackground(new java.awt.Color(28, 28, 28));
        setpointLabel.setFont(new java.awt.Font("Tahoma", 0, 14)); // NOI18N
        setpointLabel.setForeground(new java.awt.Color(255, 255, 255));
        setpointLabel.setText("Current setpoint: 100");
        setpointLabel.setToolTipText("");
        setpointLabel.setOpaque(true);

        javax.swing.GroupLayout depthPanelLayout = new
javax.swing.GroupLayout(depthPanel);
        depthPanel.setLayout(depthPanelLayout);
        depthPanelLayout.setHorizontalGroup(
            depthPanelLayout.createParallelGroup(javax.swing.GroupLayout.Align
ment.LEADING)
            .addComponent(depthHeader, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
            .addComponent(jSeparator4)
            .addGroup(depthPanelLayout.createSequentialGroup()
                .addContainerGap()
                .addGroup(depthPanelLayout.createParallelGroup(javax.swing.Gro
upLayout.Alignment.LEADING)
                    .addGroup(depthPanelLayout.createSequentialGroup()
                        .addComponent(depthInputTextField)
                        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlac
ement.UNRELATED)
                        .addComponent(sendButton))
                    .addGroup(depthPanelLayout.createSequentialGroup()
```

```
                        .addGroup(depthPanelLayout.createParallelGroup(javax.s
wing.GroupLayout.Alignment.LEADING)
                            .addGroup(depthPanelLayout.createParallelGroup(jav
ax.swing.GroupLayout.Alignment.LEADING, false)
                                .addComponent(seafloorModeButton,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)
                                .addComponent(depthModeButton,
javax.swing.GroupLayout.PREFERRED_SIZE, 133,
javax.swing.GroupLayout.PREFERRED_SIZE))
                            .addComponent(setpointLabel,
javax.swing.GroupLayout.PREFERRED_SIZE, 185,
javax.swing.GroupLayout.PREFERRED_SIZE))
                        .addGap(0, 0, Short.MAX_VALUE))))
        );
        depthPanelLayout.setVerticalGroup(
            depthPanelLayout.createParallelGroup(javax.swing.GroupLayout.Align
ment.LEADING)
            .addGroup(depthPanelLayout.createSequentialGroup()
                .addComponent(depthHeader,
javax.swing.GroupLayout.PREFERRED_SIZE, 27,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RE
LATED)
                .addComponent(jSeparator4,
javax.swing.GroupLayout.PREFERRED_SIZE, 10,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RE
LATED, javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                .addComponent(setpointLabel,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RE
LATED, javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                .addGroup(depthPanelLayout.createParallelGroup(javax.swing.Gro
upLayout.Alignment.LEADING, false)
                    .addComponent(sendButton,
javax.swing.GroupLayout.DEFAULT_SIZE, 29, Short.MAX_VALUE)
                    .addComponent(depthInputTextField))
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RE
LATED, javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
```

```
                .addComponent(depthModeButton,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RE
LATED)
                .addComponent(seafloorModeButton,
javax.swing.GroupLayout.PREFERRED_SIZE, 24,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE))
        );

        lightPanel.setBackground(new java.awt.Color(28, 28, 28));

        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");

        jSeparator1.setBackground(new java.awt.Color(204, 204, 204));
        jSeparator1.setOrientation(javax.swing.SwingConstants.VERTICAL);

        lightIndicator.setBackground(new java.awt.Color(28, 28, 28));
        lightIndicator.setFont(new java.awt.Font("Tahoma", 0, 18)); // NOI18N
        lightIndicator.setForeground(new java.awt.Color(255, 255, 255));
        lightIndicator.setHorizontalAlignment(javax.swing.SwingConstants.CENTE
R);
        lightIndicator.setText("Off");

        lightSwitch.setBackground(new java.awt.Color(28, 28, 28));
        lightSwitch.setForeground(new java.awt.Color(28, 28, 28));
        lightSwitch.setIcon(new
javax.swing.ImageIcon(getClass().getResource("/ntnusubsea/gui/Images/Power
off.gif"))); // NOI18N
        lightSwitch.setBorder(null);
        lightSwitch.setContentAreaFilled(false);
        lightSwitch.setDisabledIcon(new
javax.swing.ImageIcon(getClass().getResource("/ntnusubsea/gui/Images/Power
off.gif"))); // NOI18N
        lightSwitch.setEnabled(false);
        lightSwitch.setFocusable(false);
```

```
        lightSwitch.setSelectedIcon(new
javax.swing.ImageIcon(getClass().getResource("/ntnusubsea/gui/Images/Power
on.gif"))); // NOI18N
        lightSwitch.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                lightSwitchActionPerformed(evt);
            }
        });

        javax.swing.GroupLayout lightPanelLayout = new
javax.swing.GroupLayout(lightPanel);
        lightPanel.setLayout(lightPanelLayout);
        lightPanelLayout.setHorizontalGroup(
            lightPanelLayout.createParallelGroup(javax.swing.GroupLayout.Align
ment.LEADING)
            .addGroup(lightPanelLayout.createSequentialGroup()
                .addComponent(jSeparator1,
javax.swing.GroupLayout.PREFERRED_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addGroup(lightPanelLayout.createParallelGroup(javax.swing.Gro
upLayout.Alignment.LEADING)
                    .addComponent(lightHeader,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)
                    .addComponent(jSeparator5)
                    .addGroup(lightPanelLayout.createSequentialGroup()
                        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlac
ement.RELATED, 53, Short.MAX_VALUE)
                        .addComponent(lightSwitch,
javax.swing.GroupLayout.PREFERRED_SIZE, 105,
javax.swing.GroupLayout.PREFERRED_SIZE)
                        .addGap(0, 0, 0)
                        .addComponent(lightIndicator,
javax.swing.GroupLayout.PREFERRED_SIZE, 36,
javax.swing.GroupLayout.PREFERRED_SIZE)
                        .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE))))
        );
        lightPanelLayout.setVerticalGroup(
            lightPanelLayout.createParallelGroup(javax.swing.GroupLayout.Align
ment.LEADING)
```

```
            .addGroup(lightPanelLayout.createSequentialGroup()
                .addComponent(lightHeader,
javax.swing.GroupLayout.PREFERRED_SIZE, 27,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RE
LATED)
                .addComponent(jSeparator5,
javax.swing.GroupLayout.PREFERRED_SIZE, 10,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RE
LATED, javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                .addGroup(lightPanelLayout.createParallelGroup(javax.swing.Gro
upLayout.Alignment.LEADING)
                    .addComponent(lightSwitch,
javax.swing.GroupLayout.PREFERRED_SIZE, 102,
javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addComponent(lightIndicator,
javax.swing.GroupLayout.PREFERRED_SIZE, 102,
javax.swing.GroupLayout.PREFERRED_SIZE))
                .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE))
            .addComponent(jSeparator1,
javax.swing.GroupLayout.Alignment.TRAILING)
        );

        emergencyPanel.setBackground(new java.awt.Color(28, 28, 28));

        emergencyHeader.setBackground(new java.awt.Color(28, 28, 28));
        emergencyHeader.setFont(new java.awt.Font("Tahoma", 0, 14)); // NOI18N
        emergencyHeader.setForeground(new java.awt.Color(255, 255, 255));
        emergencyHeader.setHorizontalAlignment(javax.swing.SwingConstants.CENT
ER);
        emergencyHeader.setText("Emergency surfacing");

        emergencyStopButton.setBackground(new java.awt.Color(28, 28, 28));
        emergencyStopButton.setForeground(new java.awt.Color(28, 28, 28));
        emergencyStopButton.setIcon(new
javax.swing.ImageIcon(getClass().getResource("/ntnusubsea/gui/Images/stopbutto
n.png"))); // NOI18N
        emergencyStopButton.setBorder(null);
        emergencyStopButton.setDisabledIcon(new
```

```
javax.swing.ImageIcon(getClass().getResource("/ntnusubsea/gui/Images/stopbutto
n.png"))); // NOI18N
        emergencyStopButton.setEnabled(false);
        emergencyStopButton.addActionListener(new
java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                emergencyStopButtonActionPerformed(evt);
            }
        });

        javax.swing.GroupLayout emergencyPanelLayout = new
javax.swing.GroupLayout(emergencyPanel);
        emergencyPanel.setLayout(emergencyPanelLayout);
        emergencyPanelLayout.setHorizontalGroup(
            emergencyPanelLayout.createParallelGroup(javax.swing.GroupLayout.
Alignment.LEADING)
            .addComponent(emergencyHeader,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)
            .addComponent(jSeparator6)
            .addGroup(emergencyPanelLayout.createSequentialGroup()
                .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)
                .addComponent(emergencyStopButton,
javax.swing.GroupLayout.PREFERRED_SIZE, 100,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE))
        );
        emergencyPanelLayout.setVerticalGroup(
            emergencyPanelLayout.createParallelGroup(javax.swing.GroupLayout.
Alignment.LEADING)
            .addGroup(emergencyPanelLayout.createSequentialGroup()
                .addComponent(emergencyHeader,
javax.swing.GroupLayout.PREFERRED_SIZE, 27,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RE
LATED)
                .addComponent(jSeparator6,
javax.swing.GroupLayout.PREFERRED_SIZE, 10,
javax.swing.GroupLayout.PREFERRED_SIZE)
```

```
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RE
LATED, javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                .addComponent(emergencyStopButton,
javax.swing.GroupLayout.PREFERRED_SIZE, 100,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE))
        );

        positionPanel.setBackground(new java.awt.Color(28, 28, 28));

        positionHeader.setBackground(new java.awt.Color(28, 28, 28));
        positionHeader.setFont(new java.awt.Font("Tahoma", 0, 14)); // NOI18N
        positionHeader.setForeground(new java.awt.Color(255, 255, 255));
        positionHeader.setHorizontalAlignment(javax.swing.SwingConstants.CENTE
R);
        positionHeader.setText("Position");

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

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

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

        javax.swing.GroupLayout positionPanelLayout = new
javax.swing.GroupLayout(positionPanel);
        positionPanel.setLayout(positionPanelLayout);
        positionPanelLayout.setHorizontalGroup(
            positionPanelLayout.createParallelGroup(javax.swing.GroupLayout.Al
ignment.LEADING)
            .addComponent(positionHeader,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
```

```
Short.MAX_VALUE)
            .addComponent(jSeparator7)
            .addGroup(positionPanelLayout.createSequentialGroup()
                .addGap(10, 10, 10)
                .addGroup(positionPanelLayout.createParallelGroup(javax.swing.
GroupLayout.Alignment.LEADING)
                    .addComponent(longitudeLabel,
javax.swing.GroupLayout.Alignment.TRAILING,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)
                    .addComponent(latitudeLabel,
javax.swing.GroupLayout.Alignment.TRAILING,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)
                    .addComponent(headingLabel,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)))
        );
        positionPanelLayout.setVerticalGroup(
            positionPanelLayout.createParallelGroup(javax.swing.GroupLayout.Al
ignment.LEADING)
            .addGroup(positionPanelLayout.createSequentialGroup()
                .addComponent(positionHeader,
javax.swing.GroupLayout.PREFERRED_SIZE, 27,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RE
LATED)
                .addComponent(jSeparator7,
javax.swing.GroupLayout.PREFERRED_SIZE, 10,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RE
LATED, javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                .addComponent(headingLabel,
javax.swing.GroupLayout.PREFERRED_SIZE, 25,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RE
LATED, javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                .addComponent(latitudeLabel,
javax.swing.GroupLayout.PREFERRED_SIZE, 28,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RE
```

```
LATED, javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                .addComponent(longitudeLabel,
javax.swing.GroupLayout.PREFERRED_SIZE, 26,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE))
        );

        jSeparator2.setBackground(new java.awt.Color(204, 204, 204));
        jSeparator2.setOrientation(javax.swing.SwingConstants.VERTICAL);

        jSeparator3.setBackground(new java.awt.Color(204, 204, 204));
        jSeparator3.setOrientation(javax.swing.SwingConstants.VERTICAL);

        javax.swing.GroupLayout controlPanelLayout = new
javax.swing.GroupLayout(controlPanel);
        controlPanel.setLayout(controlPanelLayout);
        controlPanelLayout.setHorizontalGroup(
            controlPanelLayout.createParallelGroup(javax.swing.GroupLayout.Ali
gnment.LEADING)
            .addGroup(controlPanelLayout.createSequentialGroup()
                .addComponent(depthPanel,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RE
LATED)
                .addComponent(lightPanel,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RE
LATED)
                .addComponent(jSeparator2,
javax.swing.GroupLayout.PREFERRED_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addGap(0, 0, 0)
                .addComponent(emergencyPanel,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)
                .addGap(0, 0, 0)
                .addComponent(jSeparator3,
javax.swing.GroupLayout.PREFERRED_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
```

```
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addGap(0, 0, 0)
                .addComponent(positionPanel,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE))
        );
        controlPanelLayout.setVerticalGroup(
            controlPanelLayout.createParallelGroup(javax.swing.GroupLayout.Ali
gnment.LEADING)
            .addComponent(depthPanel, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
            .addComponent(lightPanel, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
            .addComponent(emergencyPanel,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)
            .addComponent(positionPanel, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
            .addComponent(jSeparator2,
javax.swing.GroupLayout.Alignment.TRAILING)
            .addComponent(jSeparator3,
javax.swing.GroupLayout.Alignment.TRAILING)
        );

        infoPanel.setBackground(new java.awt.Color(28, 28, 28));

        pitchLabel.setFont(new java.awt.Font("Tahoma", 0, 14)); // NOI18N
        pitchLabel.setForeground(new java.awt.Color(255, 255, 255));
        pitchLabel.setText("<html>Pitch angle:<br/>10");
        pitchLabel.setBorder(new
javax.swing.border.SoftBevelBorder(javax.swing.border.BevelBorder.RAISED));

        seafloorDepthRovLabel.setFont(new java.awt.Font("Tahoma", 0, 14)); //
NOI18N
        seafloorDepthRovLabel.setForeground(new java.awt.Color(255, 255,
255));
        seafloorDepthRovLabel.setText("<html>Depth beneath ROV:<br/>10");
        seafloorDepthRovLabel.setBorder(new
javax.swing.border.SoftBevelBorder(javax.swing.border.BevelBorder.RAISED));

        rovDepthLabel.setFont(new java.awt.Font("Tahoma", 0, 14)); // NOI18N
```

```
        rovDepthLabel.setForeground(new java.awt.Color(255, 255, 255));
        rovDepthLabel.setText("<html>ROV depth:<br/>10");
        rovDepthLabel.setBorder(new
javax.swing.border.SoftBevelBorder(javax.swing.border.BevelBorder.RAISED));

        wingLabel.setFont(new java.awt.Font("Tahoma", 0, 14)); // NOI18N
        wingLabel.setForeground(new java.awt.Color(255, 255, 255));
        wingLabel.setText("<html>Wing angle:<br/>10");
        wingLabel.setBorder(new
javax.swing.border.SoftBevelBorder(javax.swing.border.BevelBorder.RAISED));

        actuatorPanel1.setBackground(new java.awt.Color(28, 28, 28));
        actuatorPanel1.setBorder(new
javax.swing.border.SoftBevelBorder(javax.swing.border.BevelBorder.RAISED,
java.awt.Color.lightGray, java.awt.Color.white, java.awt.Color.darkGray,
java.awt.Color.gray));
        actuatorPanel1.setForeground(new java.awt.Color(255, 255, 255));

        actuatorHeader1.setFont(new java.awt.Font("Tahoma", 0, 14)); // NOI18N
        actuatorHeader1.setForeground(new java.awt.Color(255, 255, 255));
        actuatorHeader1.setHorizontalAlignment(javax.swing.SwingConstants.CENT
ER);
        actuatorHeader1.setText("Actuator 1 duty cycle");

        actuatorDutyCycleBar1.setForeground(new java.awt.Color(0, 255, 0));
        actuatorDutyCycleBar1.setMaximum(511);
        actuatorDutyCycleBar1.addChangeListener(new
javax.swing.event.ChangeListener() {
            public void stateChanged(javax.swing.event.ChangeEvent evt) {
                actuatorDutyCycleBar1StateChanged(evt);
            }
        });

        warningLabel1.setBackground(new java.awt.Color(28, 28, 28));
        warningLabel1.setFont(new java.awt.Font("Tahoma", 1, 12)); // NOI18N
        warningLabel1.setForeground(new java.awt.Color(255, 255, 255));
        warningLabel1.setHorizontalAlignment(javax.swing.SwingConstants.CENTER
);
        warningLabel1.setOpaque(true);

        javax.swing.GroupLayout actuatorPanel1Layout = new
```

```
javax.swing.GroupLayout(actuatorPanel1);
        actuatorPanel1.setLayout(actuatorPanel1Layout);
        actuatorPanel1Layout.setHorizontalGroup(
            actuatorPanel1Layout.createParallelGroup(javax.swing.GroupLayout.
Alignment.LEADING)
            .addComponent(actuatorHeader1,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)
            .addComponent(warningLabel1, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
            .addComponent(actuatorDutyCycleBar1,
javax.swing.GroupLayout.Alignment.TRAILING,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)
        );
        actuatorPanel1Layout.setVerticalGroup(
            actuatorPanel1Layout.createParallelGroup(javax.swing.GroupLayout.
Alignment.LEADING)
            .addGroup(actuatorPanel1Layout.createSequentialGroup()
                .addComponent(actuatorHeader1,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RE
LATED)
                .addComponent(actuatorDutyCycleBar1,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RE
LATED, javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                .addComponent(warningLabel1,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE))
        );

        actuatorPanel2.setBackground(new java.awt.Color(28, 28, 28));
        actuatorPanel2.setBorder(new
javax.swing.border.SoftBevelBorder(javax.swing.border.BevelBorder.RAISED,
java.awt.Color.lightGray, java.awt.Color.white, java.awt.Color.darkGray,
java.awt.Color.gray));
        actuatorPanel2.setForeground(new java.awt.Color(255, 255, 255));
```

```
        actuatorHeader2.setFont(new java.awt.Font("Tahoma", 0, 14)); // NOI18N
        actuatorHeader2.setForeground(new java.awt.Color(255, 255, 255));
        actuatorHeader2.setHorizontalAlignment(javax.swing.SwingConstants.CENT
ER);
        actuatorHeader2.setText("Actuator 2 duty cycle");

        actuatorDutyCycleBar2.setForeground(new java.awt.Color(0, 255, 0));
        actuatorDutyCycleBar2.setMaximum(512);
        actuatorDutyCycleBar2.setToolTipText("");
        actuatorDutyCycleBar2.addChangeListener(new
javax.swing.event.ChangeListener() {
            public void stateChanged(javax.swing.event.ChangeEvent evt) {
                actuatorDutyCycleBar2StateChanged(evt);
            }
        });

        warningLabel2.setBackground(new java.awt.Color(28, 28, 28));
        warningLabel2.setFont(new java.awt.Font("Tahoma", 1, 12)); // NOI18N
        warningLabel2.setForeground(new java.awt.Color(255, 255, 255));
        warningLabel2.setHorizontalAlignment(javax.swing.SwingConstants.CENTER
);

        javax.swing.GroupLayout actuatorPanel2Layout = new
javax.swing.GroupLayout(actuatorPanel2);
        actuatorPanel2.setLayout(actuatorPanel2Layout);
        actuatorPanel2Layout.setHorizontalGroup(
            actuatorPanel2Layout.createParallelGroup(javax.swing.GroupLayout.
Alignment.LEADING)
            .addComponent(actuatorHeader2,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)
            .addComponent(actuatorDutyCycleBar2,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)
            .addComponent(warningLabel2, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
        );
        actuatorPanel2Layout.setVerticalGroup(
            actuatorPanel2Layout.createParallelGroup(javax.swing.GroupLayout.
Alignment.LEADING)
            .addGroup(actuatorPanel2Layout.createSequentialGroup()
```

```
                .addComponent(actuatorHeader2,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RE
LATED, javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                .addComponent(actuatorDutyCycleBar2,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RE
LATED, javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                .addComponent(warningLabel2,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE))
        );

        seafloorDepthBoatLabel.setFont(new java.awt.Font("Tahoma", 0, 14)); //
NOI18N
        seafloorDepthBoatLabel.setForeground(new java.awt.Color(255, 255,
255));
        seafloorDepthBoatLabel.setText("<html>Depth beneath boat:<br/>10");
        seafloorDepthBoatLabel.setBorder(new
javax.swing.border.SoftBevelBorder(javax.swing.border.BevelBorder.RAISED));

        rollLabel.setFont(new java.awt.Font("Tahoma", 0, 14)); // NOI18N
        rollLabel.setForeground(new java.awt.Color(255, 255, 255));
        rollLabel.setText("<html>Roll angle:<br/>10");
        rollLabel.setBorder(new
javax.swing.border.SoftBevelBorder(javax.swing.border.BevelBorder.RAISED));

        leakLabel.setBackground(new java.awt.Color(28, 28, 28));
        leakLabel.setFont(new java.awt.Font("Tahoma", 0, 14)); // NOI18N
        leakLabel.setForeground(new java.awt.Color(255, 255, 255));
        leakLabel.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
        leakLabel.setIcon(new
javax.swing.ImageIcon(getClass().getResource("/ntnusubsea/gui/Images/IO_on.png
")())); // NOI18N
        leakLabel.setText("No leak detected");
        leakLabel.setBorder(new
javax.swing.border.SoftBevelBorder(javax.swing.border.BevelBorder.RAISED,
java.awt.Color.lightGray, java.awt.Color.white, java.awt.Color.darkGray,
java.awt.Color.gray));
```

```
        javax.swing.GroupLayout infoPanelLayout = new
javax.swing.GroupLayout(infoPanel);
        infoPanel.setLayout(infoPanelLayout);
        infoPanelLayout.setHorizontalGroup(
            infoPanelLayout.createParallelGroup(javax.swing.GroupLayout.Align
ment.LEADING)
            .addComponent(actuatorPanel1,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)
            .addComponent(actuatorPanel2,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)
            .addGroup(infoPanelLayout.createSequentialGroup()
                .addContainerGap(15, Short.MAX_VALUE)
                .addGroup(infoPanelLayout.createParallelGroup(javax.swing.Grou
pLayout.Alignment.TRAILING)
                    .addComponent(leakLabel,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)
                    .addGroup(infoPanelLayout.createSequentialGroup()
                        .addGroup(infoPanelLayout.createParallelGroup(javax.sw
ing.GroupLayout.Alignment.LEADING)
                            .addComponent(wingLabel,
javax.swing.GroupLayout.PREFERRED_SIZE, 140,
javax.swing.GroupLayout.PREFERRED_SIZE)
                            .addComponent(pitchLabel,
javax.swing.GroupLayout.PREFERRED_SIZE, 140,
javax.swing.GroupLayout.PREFERRED_SIZE)
                            .addComponent(rollLabel,
javax.swing.GroupLayout.PREFERRED_SIZE, 140,
javax.swing.GroupLayout.PREFERRED_SIZE))
                        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlac
ement.RELATED, javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                        .addGroup(infoPanelLayout.createParallelGroup(javax.sw
ing.GroupLayout.Alignment.LEADING)
                            .addComponent(seafloorDepthBoatLabel,
javax.swing.GroupLayout.PREFERRED_SIZE, 140,
javax.swing.GroupLayout.PREFERRED_SIZE)
                            .addComponent(rovDepthLabel,
javax.swing.GroupLayout.PREFERRED_SIZE, 140,
```

```
javax.swing.GroupLayout.PREFERRED_SIZE)
                                .addComponent(seafloorDepthRovLabel,
javax.swing.GroupLayout.PREFERRED_SIZE, 140,
javax.swing.GroupLayout.PREFERRED_SIZE))))
                .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE))
        );
        infoPanelLayout.setVerticalGroup(
            infoPanelLayout.createParallelGroup(javax.swing.GroupLayout.Align
ment.LEADING)
            .addGroup(infoPanelLayout.createSequentialGroup()
                .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)
                .addGroup(infoPanelLayout.createParallelGroup(javax.swing.Grou
pLayout.Alignment.BASELINE)
                    .addComponent(seafloorDepthBoatLabel,
javax.swing.GroupLayout.PREFERRED_SIZE, 55,
javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addComponent(rollLabel,
javax.swing.GroupLayout.PREFERRED_SIZE, 55,
javax.swing.GroupLayout.PREFERRED_SIZE))
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RE
LATED, javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                .addGroup(infoPanelLayout.createParallelGroup(javax.swing.Grou
pLayout.Alignment.LEADING, false)
                    .addComponent(pitchLabel,
javax.swing.GroupLayout.PREFERRED_SIZE, 55,
javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addComponent(seafloorDepthRovLabel,
javax.swing.GroupLayout.PREFERRED_SIZE, 55,
javax.swing.GroupLayout.PREFERRED_SIZE))
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RE
LATED, javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                .addGroup(infoPanelLayout.createParallelGroup(javax.swing.Grou
pLayout.Alignment.LEADING)
                    .addComponent(wingLabel,
javax.swing.GroupLayout.PREFERRED_SIZE, 55,
javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addComponent(rovDepthLabel,
javax.swing.GroupLayout.PREFERRED_SIZE, 55,
javax.swing.GroupLayout.PREFERRED_SIZE))
```

```
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RE
LATED, javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                .addComponent(leakLabel,
javax.swing.GroupLayout.PREFERRED_SIZE, 90,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addGap(57, 57, 57)
                .addComponent(actuatorPanel1,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RE
LATED)
                .addComponent(actuatorPanel2,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)
                .addContainerGap())
        );

        javax.swing.GroupLayout backgroundLayout = new
javax.swing.GroupLayout(background);
        background.setLayout(backgroundLayout);
        backgroundLayout.setHorizontalGroup(
            backgroundLayout.createParallelGroup(javax.swing.GroupLayout.Align
ment.LEADING)
            .addGroup(backgroundLayout.createSequentialGroup()
                .addGroup(backgroundLayout.createParallelGroup(javax.swing.Gro
upLayout.Alignment.TRAILING)
                    .addComponent(controlPanel,
javax.swing.GroupLayout.Alignment.LEADING,
javax.swing.GroupLayout.DEFAULT_SIZE, 694, Short.MAX_VALUE)
                    .addComponent(cameraPanel,
javax.swing.GroupLayout.DEFAULT_SIZE, 694, Short.MAX_VALUE))
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RE
LATED)
                .addComponent(infoPanel, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                .addContainerGap())
        );
        backgroundLayout.setVerticalGroup(
            backgroundLayout.createParallelGroup(javax.swing.GroupLayout.Align
ment.LEADING)
            .addGroup(backgroundLayout.createSequentialGroup()
```

```
                .addComponent(cameraPanel,
javax.swing.GroupLayout.DEFAULT_SIZE, 597, Short.MAX_VALUE)
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RE
LATED)
                .addComponent(controlPanel,
javax.swing.GroupLayout.DEFAULT_SIZE, 162, Short.MAX_VALUE))
            .addComponent(infoPanel,
javax.swing.GroupLayout.Alignment.TRAILING,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)
        );

        jMenuFile.setText("File");

        jMenuItemConnect.setText("Connect");
        jMenuItemConnect.addActionListener(new java.awt.event.ActionListener()
{
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jMenuItemConnectActionPerformed(evt);
            }
        });
        jMenuFile.add(jMenuItemConnect);

        jMenuItemDisconnect.setText("Disconnect");
        jMenuItemDisconnect.addActionListener(new
java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jMenuItemDisconnectActionPerformed(evt);
            }
        });
        jMenuFile.add(jMenuItemDisconnect);

        jMenuItemExit.setText("Exit");
        jMenuItemExit.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jMenuItemExitActionPerformed(evt);
            }
        });
        jMenuFile.add(jMenuItemExit);

        jMenuBar.add(jMenuFile);
```

```
        jMenuEdit.setText("Edit");

        jMenuUndo.setAccelerator(javax.swing.KeyStroke.getKeyStroke(java.awt.e
vent.KeyEvent.VK_Z, java.awt.event.InputEvent.CTRL_MASK));
        jMenuUndo.setText("Undo");
        jMenuUndo.setToolTipText("");
        jMenuUndo.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jMenuUndoActionPerformed(evt);
            }
        });
        jMenuEdit.add(jMenuUndo);

        jMenuRedo.setAccelerator(javax.swing.KeyStroke.getKeyStroke(java.awt.e
vent.KeyEvent.VK_Y, java.awt.event.InputEvent.CTRL_MASK));
        jMenuRedo.setText("Redo");
        jMenuRedo.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jMenuRedoActionPerformed(evt);
            }
        });
        jMenuEdit.add(jMenuRedo);

        jMenuBar.add(jMenuEdit);

        jMenuTools.setText("Tools");

        jMenuEchosounder.setText("Echo sounder");
        jMenuEchosounder.addActionListener(new java.awt.event.ActionListener()
{
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jMenuEchosounderActionPerformed(evt);
            }
        });
        jMenuTools.add(jMenuEchosounder);

        jMenuIOController.setText("I/O Controller");
        jMenuIOController.addActionListener(new
java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
```

```
                jMenuIOControllerActionPerformed(evt);
            }
        });
        jMenuTools.add(jMenuIOController);

        jMenuOptions.setText("Options");
        jMenuOptions.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jMenuOptionsActionPerformed(evt);
            }
        });
        jMenuTools.add(jMenuOptions);

        jMenuBar.add(jMenuTools);

        jMenuHelp.setText("Help");
        jMenuHelp.setToolTipText("");

        jMenuAbout.setText("About");
        jMenuAbout.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jMenuAboutActionPerformed(evt);
            }
        });
        jMenuHelp.add(jMenuAbout);

        jMenuBar.add(jMenuHelp);

        setJMenuBar(jMenuBar);

        javax.swing.GroupLayout layout = new
javax.swing.GroupLayout(getContentPane());
        getContentPane().setLayout(layout);
        layout.setHorizontalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADI
NG)
            .addComponent(background, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
        );
        layout.setVerticalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADI
```

```
NG)
            .addComponent(background,
javax.swing.GroupLayout.Alignment.TRAILING,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)
        );

        pack();
    }// </editor-fold>

    private void jMenuUndoActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        redoSetpoint = setpoint;
        setpoint = previousSetpoint;
        depthInputTextField.setText(setpoint.toString());
        sendButton.doClick();
    }

    private void fullscreenButtonActionPerformed(java.awt.event.ActionEvent
evt) {
        Rectangle maximumWindowBounds =
GraphicsEnvironment.getLocalGraphicsEnvironment().getMaximumWindowBounds();
        int width = (int) maximumWindowBounds.getWidth();
        int height = (int) maximumWindowBounds.getHeight();
        fullscreen.setLocationRelativeTo(this);
        this.setVisible(false);
        fullscreen.setExtendedState(MAXIMIZED_BOTH);
        fullscreen.setUndecorated(true);
        fullscreen.setVisible(true);
        exitFullscreenButton.setBounds(width -
exitFullscreenButton.getWidth(), height - exitFullscreenButton.getHeight() -
25, 30, 30);
    }

    private void
exitFullscreenButtonActionPerformed(java.awt.event.ActionEvent evt) {
        fullscreen.setVisible(false);
        fullscreen.dispose();
        this.setVisible(true);
    }
```

```
    private void fullscreenKeyPressed(java.awt.event.KeyEvent evt) {
        int key = evt.getKeyCode();
        if (key == KeyEvent.VK_ESCAPE) {
            fullscreen.dispose();
        }
        System.out.println(key);
    }

    private void
actuatorDutyCycleBar1StateChanged(javax.swing.event.ChangeEvent evt) {
        int actuatorTime1 = actuatorDutyCycleBar1.getValue();
        if (actuatorTime1 <= 255) {
            actuatorDutyCycleBar1.setForeground(new Color(actuatorTime1, 255,
0));
            warningLabel1.setText("");
            warningLabel1.setBackground(new Color(28, 28, 28));
        } else {
            actuatorDutyCycleBar1.setForeground(new Color(255, 255 + (256 -
actuatorTime1), 0));
            warningLabel1.setText("Warning!");
            warningLabel1.setBackground(Color.red);
        }
    }

    private void sendButtonActionPerformed(java.awt.event.ActionEvent evt) {

        try {
            depthInputTextField.commitEdit();
            double newSetpoint;
            try {
                newSetpoint = (double) depthInputTextField.getValue();
            } catch (ClassCastException ex) {
                Long newSetpointLong = (long) depthInputTextField.getValue();
                newSetpoint = newSetpointLong.doubleValue();
            }
            //newSetpoint = Double.parseDouble(new
DecimalFormat("#.000").format(newSetpoint));
            if (newSetpoint <= 50 && newSetpoint >= 0) {
                try {
                    if (true) {
                        client.sendCommand(SETPOINTID + newSetpoint);
```

```
                        client.sendCommand(MODEID + mode);
                        //       System.out.println("<Setpoint>" +
newSetpoint);
                        //       System.out.println("<Mode>" + mode);
                        setpointLabel.setBackground(new Color(28, 28, 28));
                        previousSetpoint = setpoint;
                        setpoint = newSetpoint;
                        depthInputTextField.setValue(null);
                        setpointLabel.setText("Current setpoint: " + setpoint
+ "m");
                    } else {
                        depthInputTextField.setValue(null);
                        JOptionPane.showMessageDialog(this,
                                "No connection established.",
                                "Connection error",
                                JOptionPane.ERROR_MESSAGE);
                    }
                } catch (Exception ex) {
                    System.out.println(ex.getMessage());
                }
            } else {
                depthInputTextField.setValue(null);
                JOptionPane.showMessageDialog(this,
                        "Input is invalid. (Max depth 50m)",
                        "Input error",
                        JOptionPane.ERROR_MESSAGE);
            }
        } catch (ParseException ex) {
            depthInputTextField.setValue(null);
            System.out.println(ex.getMessage());

        }

    }

    private void seafloorModeButtonActionPerformed(java.awt.event.ActionEvent
evt) {
        mode = 1;
        System.out.println("Mode 1");
    }
```

```
    private void lightSwitchActionPerformed(java.awt.event.ActionEvent evt) {
        if (lightSwitch.isSelected()) {
            try {
                client.sendCommand(LIGHTSID + 1);
                lightIndicator.setText("On");
            } catch (IOException ex) {
                Logger.getLogger(ROVFrame.class.getName()).log(Level.SEVERE,
null, ex);
            }
        } else {
            try {
                client.sendCommand(LIGHTSID + 0);
                lightIndicator.setText("Off");
            } catch (IOException ex) {
                Logger.getLogger(ROVFrame.class.getName()).log(Level.SEVERE,
null, ex);
            }
        }
    }

    private void emergencyStopButtonActionPerformed(java.awt.event.ActionEvent
evt) {
//        previousSetpoint = setpoint;
//        setpoint = 0.000;
        setpointLabel.setText("Emergency! Setpoint: " + setpoint + "m");
        setpointLabel.setBackground(new Color(255, 0, 0));
        depthModeButton.doClick();
        depthInputTextField.setText("0");
        sendButton.doClick();
    }

    private void jMenuItemConnectActionPerformed(java.awt.event.ActionEvent
evt) {
        String ip = (String) JOptionPane.showInputDialog(this, "Enter IP",
"Connection", JOptionPane.PLAIN_MESSAGE, null, null, data.getDefaultIP());
        try {
            client.connect(ip);
            sendButton.setEnabled(true);
            lightSwitch.setEnabled(true);
            emergencyStopButton.setEnabled(true);
            io.enableIO();
```

```
            client.sendCommand("<KP>" + data.getKp());
            Thread.sleep(100);
            client.sendCommand("<KI>" + data.getKi());
            Thread.sleep(100);
            client.sendCommand("<KD>" + data.getKd());
        } catch (Exception ex) {
            JOptionPane.showMessageDialog(this,
                    "Connection failed.",
                    "Conncetion error",
                    JOptionPane.ERROR_MESSAGE);
        }
    }

    private void jMenuItemDisconnectActionPerformed(java.awt.event.ActionEvent
evt) {
        try {
            client.sendCommand("Quit");
            client.disconnect();
            sendButton.setEnabled(false);
            lightSwitch.setEnabled(false);
            emergencyStopButton.setEnabled(false);
            io.disableIO();
            JOptionPane.showMessageDialog(this,
                    "Successfully disconnected.",
                    "Disconnected",
                    JOptionPane.PLAIN_MESSAGE);
        } catch (IOException ex) {
            JOptionPane.showMessageDialog(this,
                    "Failed to disconnect.",
                    "Disconnect error",
                    JOptionPane.ERROR_MESSAGE);
        }
    }

    private void jMenuItemExitActionPerformed(java.awt.event.ActionEvent evt)
{
        System.exit(0);
    }

    private void jMenuRedoActionPerformed(java.awt.event.ActionEvent evt) {
        Double tempsetpoint = setpoint;
```

```
        setpoint = redoSetpoint;
        depthInputTextField.setText(setpoint.toString());
        sendButton.doClick();
        previousSetpoint = tempsetpoint;
    }


    private void jMenuEchosounderActionPerformed(java.awt.event.ActionEvent
evt) {
        if (echoSounder.isVisible()) {
            echoSounder.setVisible(false);
        } else {
            echoSounder.setVisible(true);
        }
    }

    private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
        helpframe.dispose();
    }

    private void jMenuAboutActionPerformed(java.awt.event.ActionEvent evt) {
        helpframe.setVisible(true);
        helpframe.setSize(helpframe.getPreferredSize());
        helpframe.setLocation(this.getLocation().x, this.getLocation().y);
// TODO add your handling code here:
    }

    private void
actuatorDutyCycleBar2StateChanged(javax.swing.event.ChangeEvent evt) {
        int actuatorTime2 = actuatorDutyCycleBar2.getValue();
        if (actuatorTime2 <= 255) {
            actuatorDutyCycleBar2.setForeground(new Color(actuatorTime2, 255,
0));
            warningLabel2.setText("");
            warningLabel2.setBackground(new Color(28, 28, 28));
        } else {
            actuatorDutyCycleBar2.setForeground(new Color(255, 255 + (256 -
actuatorTime2), 0));
            warningLabel2.setText("Warning!");
            warningLabel2.setBackground(Color.red);
        }
```

```
    }

    private void jMenuOptionsActionPerformed(java.awt.event.ActionEvent evt) {
        OptionsFrame options = new OptionsFrame(data);
        options.setVisible(true);
        options.setLocation(this.getLocation().x, this.getLocation().y);
    }

    private void depthModeButtonActionPerformed(java.awt.event.ActionEvent
evt) {
        mode = 0;
        System.out.println("Mode 0");
    }

    private void jMenuIOControllerActionPerformed(java.awt.event.ActionEvent
evt) {
        if (io.isVisible()) {
            io.setVisible(false);
        } else {
            io.setVisible(true);
        }
    }

    Action exitFullscreenAction = new AbstractAction() {
        public void actionPerformed(ActionEvent e) {
            exitFullscreenButton.doClick();
        }
    };

    Action sendInputAction = new AbstractAction() {
        public void actionPerformed(ActionEvent e) {
            if (depthInputTextField.isFocusOwner()) {
                sendButton.doClick();
            }

        }
    };

    // Variables declaration - do not modify
    private javax.swing.JProgressBar actuatorDutyCycleBar1;
    private javax.swing.JProgressBar actuatorDutyCycleBar2;
```

```
    private javax.swing.JLabel actuatorHeader1;
    private javax.swing.JLabel actuatorHeader2;
    private javax.swing.JPanel actuatorPanel1;
    private javax.swing.JPanel actuatorPanel2;
    private javax.swing.JPanel background;
    private javax.swing.ButtonGroup buttonGroup1;
    private javax.swing.JPanel cameraPanel;
    private javax.swing.JPanel cameraPanel1;
    private javax.swing.JPanel controlPanel;
    private javax.swing.JLabel depthHeader;
    private javax.swing.JFormattedTextField depthInputTextField;
    private javax.swing.JRadioButton depthModeButton;
    private javax.swing.JPanel depthPanel;
    private javax.swing.JLabel emergencyHeader;
    private javax.swing.JPanel emergencyPanel;
    private javax.swing.JButton emergencyStopButton;
    private javax.swing.JButton exitFullscreenButton;
    private javax.swing.JFrame fullscreen;
    private javax.swing.JButton fullscreenButton;
    private javax.swing.JLabel headingLabel;
    private javax.swing.JFrame helpframe;
    private javax.swing.JPanel infoPanel;
    private javax.swing.JButton jButton1;
    private javax.swing.JLabel jLabel1;
    private javax.swing.JMenuItem jMenuAbout;
    private javax.swing.JMenuBar jMenuBar;
    private javax.swing.JMenuItem jMenuEchosounder;
    private javax.swing.JMenu jMenuEdit;
    private javax.swing.JMenu jMenuFile;
    private javax.swing.JMenu jMenuHelp;
    private javax.swing.JMenuItem jMenuIOController;
    private javax.swing.JMenuItem jMenuItemConnect;
    private javax.swing.JMenuItem jMenuItemDisconnect;
    private javax.swing.JMenuItem jMenuItemExit;
    private javax.swing.JMenuItem jMenuOptions;
    private javax.swing.JMenuItem jMenuRedo;
    private javax.swing.JMenu jMenuTools;
    private javax.swing.JMenuItem jMenuUndo;
    private javax.swing.JSeparator jSeparator1;
    private javax.swing.JSeparator jSeparator2;
    private javax.swing.JSeparator jSeparator3;
```

```
    private javax.swing.JSeparator jSeparator4;
    private javax.swing.JSeparator jSeparator5;
    private javax.swing.JSeparator jSeparator6;
    private javax.swing.JSeparator jSeparator7;
    private javax.swing.JLabel latitudeLabel;
    private javax.swing.JLabel leakLabel;
    private javax.swing.JLabel lightHeader;
    private javax.swing.JLabel lightIndicator;
    private javax.swing.JPanel lightPanel;
    private javax.swing.JToggleButton lightSwitch;
    private javax.swing.JLabel longitudeLabel;
    private javax.swing.JLabel pitchLabel;
    private javax.swing.JLabel positionHeader;
    private javax.swing.JPanel positionPanel;
    private javax.swing.JLabel rollLabel;
    private javax.swing.JLabel rovDepthLabel;
    private javax.swing.JLabel seafloorDepthBoatLabel;
    private javax.swing.JLabel seafloorDepthRovLabel;
    private javax.swing.JRadioButton seafloorModeButton;
    private javax.swing.JButton sendButton;
    private javax.swing.JLabel setpointLabel;
    private javax.swing.JLabel warningLabel1;
    private javax.swing.JLabel warningLabel2;
    private javax.swing.JLabel wingLabel;
    // End of variables declaration

    @Override
    public void run() {
        try {
            videoImage = ImageIO.read(new
File("C:\\Users\\marno\\OneDrive\\Documents\\NetBeansProjects\\NTNUSubsea
GUI\\src\\ntnusubsea\\gui\\Images\\banan.jpg"));
        } catch (IOException ex) {
            Logger.getLogger(NTNUSubseaGUI.class.getName()).log(Level.SEVERE,
null, ex);
        }
        this.showImage(videoImage);
        this.setVisible(true); //To change body of generated methods, choose
Tools | Templates.

        //
```

```
        try {
            //  wingImage = ImageIO.read(new File("Images/rovwing.png"));
        } catch (Exception ex) {
            System.out.println(ex.getMessage());
        }
        this.showImage(videoImage);

//        try {
//            Thread.sleep(10);
//        } catch (InterruptedException ex) {
//
Logger.getLogger(NTNUSubseaGUI.class.getName()).log(Level.SEVERE, null, ex);
//        }
    }

    @Override
    public void update(Observable o, Object arg) {
//        actuatorDutyCycleBar1.setValue(data.getBarValue());
        System.out.println(data.getPitchAngle());
        this.showImage(data.getVideoImage());
        headingLabel.setText("Heading: " + data.getHeading());
        longitudeLabel.setText("Longitude: " + data.getLongitude());
        latitudeLabel.setText("Latitude: " + data.getLatitude());
        rovDepthLabel.setText("<html>ROV depth:<br/>" + data.getDepth());
        rollLabel.setText("<html>Roll angle:<br/>" + data.getRollAngle());
        pitchLabel.setText("<html>Pitch angle:<br/>" + data.getPitchAngle());
        wingLabel.setText("<html>Wing angle:<br/>" + data.getWingAngle());
        seafloorDepthBoatLabel.setText("<html>Depth beneath boat:<br/>" +
data.getSeafloorBoat());
        seafloorDepthRovLabel.setText("<html>Depth beneath ROV:<br/>" +
data.getSeafloorRov());
        if (data.getLeakStatus()) {
            leakLabel.setText("Leak detected!");
            try {
                leakLabel.setIcon(new ImageIcon(ImageIO.read(new
File("src/ntnusubsea/gui/Images/IO_off.png"))));
            } catch (IOException ex) {
                Logger.getLogger(ROVFrame.class.getName()).log(Level.SEVERE,
null, ex);
            }
        } else {
```

```
            leakLabel.setText("No leak detected");
            try {
                leakLabel.setIcon(new ImageIcon(ImageIO.read(new
File("src/ntnusubsea/gui/Images/IO_on.png"))));
            } catch (IOException ex) {
                Logger.getLogger(ROVFrame.class.getName()).log(Level.SEVERE,
null, ex);
            }
        }
    }
}
```

```
/*
 * To change this license header, choose License Headers in Project
Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package ntnusubsea.gui;

import java.io.*;
import java.net.*;
import java.nio.ByteBuffer;
import java.nio.ByteOrder;
import java.nio.channels.SocketChannel;

/**
 * Client class that handles the connection to the server, retrieves the video
 * stream and sends commands to the server
 *
 * @author Marius Nonsvik
 */
public class TCPClient implements Runnable {
    private boolean connected = false;
    private Socket soc;
    private InputStream inputStream;
    private OutputStream outputStream;
    private String IP;
    private final int port = 5001;
    private ObjectInputStream ois;
    private PrintStream ps;
    private Data data;
    private SocketChannel sc;

    public TCPClient(Data data) {
        this.data = data;
    }

    /**
     * Connects the client to the server through a socket and saves the IP and
     * port to the global variables IP and port
     *
     * @param IP IP of the server to connect to
```

```
     * @throws IOException Throws an IOException when the connection is
     * unsuccessful
     */
    public void connect(String IP) throws IOException {
        soc = new Socket(IP, port);
        this.inputStream = soc.getInputStream();
        this.outputStream = soc.getOutputStream();
        this.ps = new PrintStream(outputStream);
        this.IP = IP;
        connected = true;
    }

    /**
     * Retrieves the byte array containing sensor data from the ROV and
updates
     * these values in the data storage.
     *
     * @throws Exception Throws exception if header is invalid or data is null
     */
    public void receiveData() throws Exception {

        byte[] byteArray = new byte[42];
        if (inputStream.available() >= 42) {
            inputStream.read(byteArray, 0, 42);
            data.setActuatorStatus(ByteBuffer.wrap(byteArray).order(ByteOrder.
BIG_ENDIAN).get(0));
            data.setDepth(ByteBuffer.wrap(byteArray).order(ByteOrder.BIG_ENDI
AN).getFloat(1));
            data.setSeafloorRov(ByteBuffer.wrap(byteArray).order(ByteOrder.BIG
_ENDIAN).getFloat(5)); // not tested
            data.setTemperature(ByteBuffer.wrap(byteArray).order(ByteOrder.BIG
_ENDIAN).getFloat(9));
            data.setPressure(ByteBuffer.wrap(byteArray).order(ByteOrder.BIG_EN
DIAN).getFloat(13));
            data.setRollAngle(ByteBuffer.wrap(byteArray).order(ByteOrder.BIG_E
NDIAN).getFloat(17));
            data.setPitchAngle(ByteBuffer.wrap(byteArray).order(ByteOrder.BIG
_ENDIAN).getFloat(21));
            data.setLeakStatus(ByteBuffer.wrap(byteArray).order(ByteOrder.BIG
_ENDIAN).get(25));
            data.setChannel(ByteBuffer.wrap(byteArray).order(ByteOrder.LITTLE
```

```
_ENDIAN).getFloat(26), 1);
            data.setChannel(ByteBuffer.wrap(byteArray).order(ByteOrder.LITTLE
_ENDIAN).getFloat(30), 2);
            data.setChannel(ByteBuffer.wrap(byteArray).order(ByteOrder.LITTLE
_ENDIAN).getFloat(34), 3);
            data.setChannel(ByteBuffer.wrap(byteArray).order(ByteOrder.LITTLE
_ENDIAN).getFloat(38), 4);
        }
    }

    /**
     * Sends a command to the server.
     *
     * @param s String containing the command to send
     * @throws IOException Throws IOException if client is disconnected
     */
    public synchronized void sendCommand(String s) throws IOException {
        ps.print(s + System.getProperty("line.separator"));
        outputStream.flush();
        System.out.println(s);
    }


    /**
     * Returns the connection status of the socket
     *
     * @return The connection status of the socket
     */
    public boolean isConnected() {
        return connected;
    }

    /**
     * Closes the socket if the client is currently connected
     *
     * @throws IOException Throws IOException if there is a problem with the
     * connection
     */
    public void disconnect() throws IOException {
        if (soc != null) {
            soc.close();
```

```
        }
        connected = false;
    }

    @Override
    public void run() {
        if (connected) {
            try {
                receiveData();
            } catch (Exception ex) {

            }
        }
    }
}
```

```
/*
 * 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
 * sesnors. There will also be a video stream from the ROV.
 *
 * The system consists of a Raspberry Pi on the ROV that is connected to
several
 * Arduino microcontrollers that are connected to sensors and and programed
with
 * the control system.
 * The external computer which is on the surface vessel is connected to a GPS,
 * echo sounder and the RBPi. It will present and save data in addition to
 * handle user commands.
 */
package ntnusubsea.gui;

import java.awt.image.BufferedImage;
import java.io.ByteArrayInputStream;
import java.io.IOException;
import java.net.DatagramPacket;
import java.net.DatagramSocket;
import java.net.SocketException;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.imageio.ImageIO;

/**
 * This class handles incoming images from a DatagramPacket. It receives the
 * image on a DatagramSocket and returns it as a BufferedImage.
 *
 * @author MorSol and Marius Nonsvik
 */
public class UDPClient implements Runnable {

    static BufferedImage videoImage;
  //  static Socket videoSocket;
    private boolean debug = false;
    private Data data;
    private int test = 0;
    private  DatagramSocket videoSocket;
```

```
    private long timer = System.currentTimeMillis();

    public UDPClient(Data data) {
        try{
                    this.data = data;
        videoSocket = new DatagramSocket(5002);
        }
        catch(Exception e){

        }

    }

    @Override
    public void run() {
        try {
            if(System.currentTimeMillis() - timer > 60000){
                videoSocket.close();
                videoSocket = new DatagramSocket(5002);
                timer = System.currentTimeMillis();
                System.out.println("Reconnected");
            }
            //Createss new DatagramSocket for reciving DatagramPackets


            //Creating new DatagramPacket form the packet recived on the
videoSocket
            byte[] receivedData = new byte[60000];
            DatagramPacket receivePacket = new DatagramPacket(receivedData,
                    receivedData.length);
            if (receivePacket.getLength() > 0) {
                long startTime = System.currentTimeMillis();
                //Updates the videoImage from the received DatagramPacket
                videoSocket.receive(receivePacket);
                long endTime = System.currentTimeMillis();
                if (debug) {
                    System.out.println("Videopackage received");
                }
                //Reads incomming byte array into a BufferedImage
                ByteArrayInputStream bais = new
ByteArrayInputStream(receivedData);
```

```
                videoImage = ImageIO.read(bais);
                data.setVideoImage(videoImage);
                receivedData = null;
                bais = null;
                test++;
                System.out.println(endTime - startTime);
            }

        } catch (SocketException ex) {
            Logger.getLogger(UDPClient.class.getName()).log(Level.SEVERE,
null, ex);

        } catch (IOException ex) {
            Logger.getLogger(UDPClient.class.getName()).log(Level.SEVERE,
null, ex);
        }

    }
}
```

```
/*
 * To change this license header, choose License Headers in Project
Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.

 */package ntnusubsea.gui;

import java.awt.image.BufferedImage;
import java.io.File;
import java.io.IOException;
import java.time.LocalDateTime;
import java.util.Observable;
import java.util.Observer;
import java.util.logging.Level;
import java.util.logging.Logger;
import org.jcodec.api.awt.AWTSequenceEncoder;


/**
 * The class video encoder uses buffered image and encodes there images to a
 * video file.
 *
 * @author Marius Nonsvik
 *
 */
public class VideoEncoder implements Runnable, Observer {

   // private ArrayList<BufferedImage> list = new ArrayList();
    private BufferedImage videoImage;
    private Data data;
    private AWTSequenceEncoder enc;
    private int frame = 0;
    private LocalDateTime startTime;

    /**
     * Creates an instance of video encoder and create the MP4 file and gives
it
     * a unique name
     * @param data Data containing the images
     */
```

```
    public VideoEncoder(Data data) {
        try {
            startTime = LocalDateTime.now();
            String minute, hour, day, month, year;
            if(startTime.getMinute() < 10){
                minute = "0" + Integer.toString(startTime.getMinute());
            }
            else{
                minute = Integer.toString(startTime.getMinute());
            }
            if(startTime.getHour() < 10){
                hour = "0" + Integer.toString(startTime.getHour());
            }
            else{
                hour = Integer.toString(startTime.getHour());
            }
            if (startTime.getDayOfMonth() < 10) {
                day = "0" + Integer.toString(startTime.getDayOfMonth());
            } else {
                day = Integer.toString(startTime.getDayOfMonth());
            }
            if (startTime.getMonthValue() < 10) {
                month = "0" + Integer.toString(startTime.getMonthValue());
            } else {
                month = Integer.toString(startTime.getMonthValue());
            }
            year = Integer.toString(startTime.getYear());
            String fileName = "ROV Video " + hour + minute + " " + day + "." +
month + "." + year + ".mp4";
            enc = AWTSequenceEncoder.create24Fps(new File(fileName));

            this.data = data;
        } catch (IOException ex) {
            Logger.getLogger(VideoEncoder.class.getName()).log(Level.SEVERE,
null, ex);
        }
    }

    @Override
    public void run() {
       // if (frame < list.size()) {
```

```
            // if (frame < list.size()) {
            try {
         //       BufferedImage image = list.get(frame);
                enc.encodeImage(videoImage);
              //  frame++;
            } catch (Exception ex) {
                Logger.getLogger(VideoEncoder.class.getName()).log(Level.SEVER
E, null, ex);
            }
        }


    @Override
    public void update(Observable o, Object arg) {
        videoImage = data.getVideoImage();
//        if(!list.contains(image)){
//        list.add(data.getVideoImage());
//        }
    }

    /**
     * Encodes the remaining images and finishes the video
     */
    public void finishVideo() {
        try {
//            for (int i = frame; i < list.size(); i++) {
//                enc.encodeImage(list.get(i));
//            }
            enc.finish();
        } catch (IOException ex) {
            Logger.getLogger(VideoEncoder.class.getName()).log(Level.SEVERE,
null, ex);
        }
    }
}
```