NTNU
Norwegian University of
Science and Technology

Integration of Aquaculture Inspection Platform

BACHELOR THESIS

Ole Grytdal Morken (IIR: 10060)
Lars Even Sætre (IIR: 10064)
Sigurd Olav Liavaag (IIR: 10053)

Total number of pages: 95/471

Delivered: 20.05.2019, Ålesund

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

Supervisors: Ottar Laurits Osen, Houxiang Zhang

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 Osen, Houxiang Zhang

Fullmakt til elektronisk publisering av oppgaven

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

Jeg/vi gir herved NTNU i Ålesund en vederlagsfri rett til å gjøre oppgaven tilgjengelig for elektronisk publisering: ☒**ja** ☐**nei**

Er oppgaven båndlagt (konfidensiell)? ☐**ja** ☒**nei**
(Båndleggingsavtale må fylles ut)
- Hvis ja:
Kan oppgaven publiseres når båndleggingsperioden er over? ☐**ja** ☐**nei**

Er oppgaven unntatt offentlighet? ☐**ja** ☒**nei**
(inneholder taushetsbelagt informasjon. Jfr. Offl. §13/Fvl. §13)

Dato: 20.05.19

Preface

Figure 1: New Platform

Acknowledgement

We would like to thank:

- Our mentors Ottar L. Osen and Houxiang Zhang for continuous support and guidance throughout this project
- Family and friends who have supported us through this time
- Anders Sætermoen and Øyvind Andre Hanken for supplying us with parts
- DIMO AS for their generosity with sponsoring us with parts
- Saysound for lending us their trailer for transport of the platform
- Ålesunds sailclub for use of their crane and facilities while testing
- Andre Tranvåg and apprentices for help making mechanical parts
- Fellow students for good discussions about the project

Summary

This thesis concerns the integration of three previously developed systems, a platform, a winch and a ROV. The goal of this project is to show the functionality of the system together. The platform can transport the ROV to a desired location and the winch can deploy and retrieve the ROV. The platform will have an autopilot and dynamic positioning for ease of use. During development of the new system it's been emphasized that it should be easy to continue development in the future. A user-friendly application to control the platform was developed using Java. The control system consist of a PLC and raspberry pi to achieve dynamic positioning and autopilot for the platform while the ROV is in operation. The results from testing shows that the prototypes are integrated to a single system and that the functionality is achieved within the requirements.

Contents

Terminology

PID Proportional integral derivative controller

GUI Graphical User Interface, makes it possible to interact with a computer

HMI Human Machine Interface, a general description for interacting with a machine

API Application Programming Interface, activates functions from a remote software

TCP Transmission Control Protocol, connection oriented transmission protocol of information.

UDP User Datagram Protocol, non connection based transmission protocol of information.

IP Internet Protocol is a "best effort" delivery protocol

USV Unmanned Surface Vehicle, a vehicle that operates on the surface of the water without a crew

ROV Remotely Operated Vehicle, a remotely operated underwater vehicle

ACK Acknowledge message

SBC Single-Board Computer

SMC Simple Motor Controller

GPSD Global Positioning System Daemond

GPS Global Positioning System

CAD Computer-Aided Design

Notation

K_p Proportional term of a PID controller

K_i Integral term of a PID controller

K_d Derivative term of a PID controller

Kg System International unit for Kilogram

Abbreviations

IEEE Institute of Electronical and Electronic Engineers

I2C Inter Integrated Circuit

Gnd Ground in electronical circuits

DOF Degrees of Freedom, number of configurations for an object

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background

The fish farm industry is one of the largest export businesses in Norway, and it is a big industry in the northwestern part of Norway. The fish farms are placed along the coast and in the fjords where they farm mainly salmon and trout. In 2017 the fish farm industry exported fish for over 60 billion NOK [1]. With rising demand for quality, the margin for error decreases, by using automation and robotics, the industry can increase productivity and quality without increasing the cost of production.

1.2 Introduction

Students from NTNU have previously developed a prototype for an aquaculture inspection platform. This has been done over several projects due to the amount of work needed to complete the project. The project is divided into three parts; the platform, the winch, and the ROV. The purpose of the inspection platform is to do underwater survey in, and around fish farms using the ROV mounted on the platform. The platform has an electronic propulsion system with autopilot for transport between geographical positions and dynamic positioning to stay in position when the ROV is deployed to do underwater survey. At the start of this project, it consisted of three stand-alone systems with separate control systems. In this report a solution to integrate three systems to one will be presented. It will also show problems encountered regarding buoyancy, stability, leakage, mechanical weakness, electrical interference, power consumption, and how these issues where solved.

1.3 Problem Formulation

How can the three systems be integrated together so they work as one system?
What changes needs to be made so the final product is more robust and user-friendly?

Problems to be addressed

- Re-engineering of systems to be compatible as a single system.
- Develop a complete control system.

1.4 Literature Survey

This report is a continuation of previous projects done at NTNU Ålesund from 2016 to 2017, the papers are listed below.

- AQUAFARM INSPECTION - ROV, 2016 [12]
- Sea farm platform, 2017 [13]
- A Novel Low Cost ROV for Aquaculture Application, 2017 [15]
- SIMPLE WINCH FOR SEAFARM, 2017 [14]

1.5 Objectives

The Objectives for this thesis are:

1. The integration of the subsystems will show the functionality of the complete system.
2. The USV can be controlled manually, move on autopilot between geographical positions and have a functionally DP system with the use of GPS.

3. The USV can deploy and recover the ROV using the winch.

4. The control system should be simple to extend in further development.

1.6 Limitations

The project is partly limited to how much mechanical changes that can be made due to the nature of the design.

1.7 Approach

At the beginning of this project the group read the project reports from the previous groups to get an understanding of what had been done, and why. After this the group tested the systems to get an understanding of the functionality and the programming. This laid the path for how the group proceeded and what the results are in this report.

1.8 Structure of the Report

The rest of the report is structured as follows.

Chapter 2 - Theoretical basis: Chapter two gives an introduction to the theoretical background used to obtain the knowledge to complete this project.

Chapter 3 - Method: Contains a description of the methodology and materials that were considered throughout the project in order to make the best product possible.

Chapter 4 - Result: Contains a description of the finished product and results from testing.

Chapter 5 - Discussion: A summary of the results and the group's opinions on them.

Chapter 6 - Conclusions: This chapter present an overall conclusion on the project and the result.

Chapter 2

Theoretical basis

2.1 Previous work

When this project was started, it consisted of three systems mechanically mounted together. In order to operate the systems, the control systems had to be operated separately. From a software point of view, the control systems were not built in a way that they could be integrated without major changes. After the ROV and winch were mounted on the platform, it was heavier than calculated, so bouyancy could be an issue.

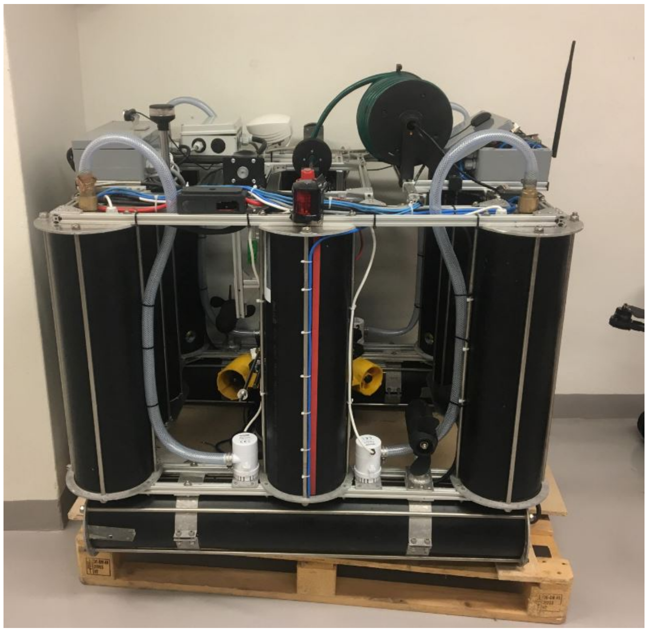


Figure 2.1: Old platform

2.1.1 Sea farm platform

The sea farm platform was built as a part of a bachelor thesis in 2017 by four students from automation. The platform was fitted with a propulsion system, a stabilization system, and a nonfunctional autopilot. In other projects, an ROV and a launch and recovery system was fitted to the sea farm platform by other students.

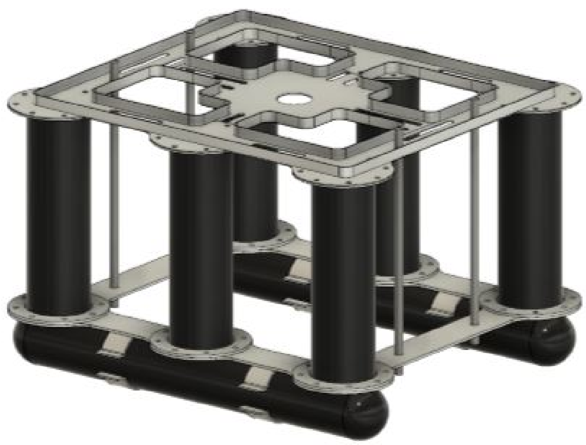


Figure 2.2: Original Platform design

Figure 2.3: Stability test

The control system for the sea farm platform was an Odroid(SBC) that controlled two arduinos that controlled the propulsion and stabilization system. An android tablet was used as a GUI and communicated with the Odroid 2.4.

Platform Design

The platform is designed in a way that it will be less affected by waves. The vertical pipes which have built-in pumps to pump in or out water give the platform the ability to raise and lower itself to gain stability while the ROV is deployed. By using these pumps, the platform can stabilize itself in waves by adjusting the amount of water in each pipe.

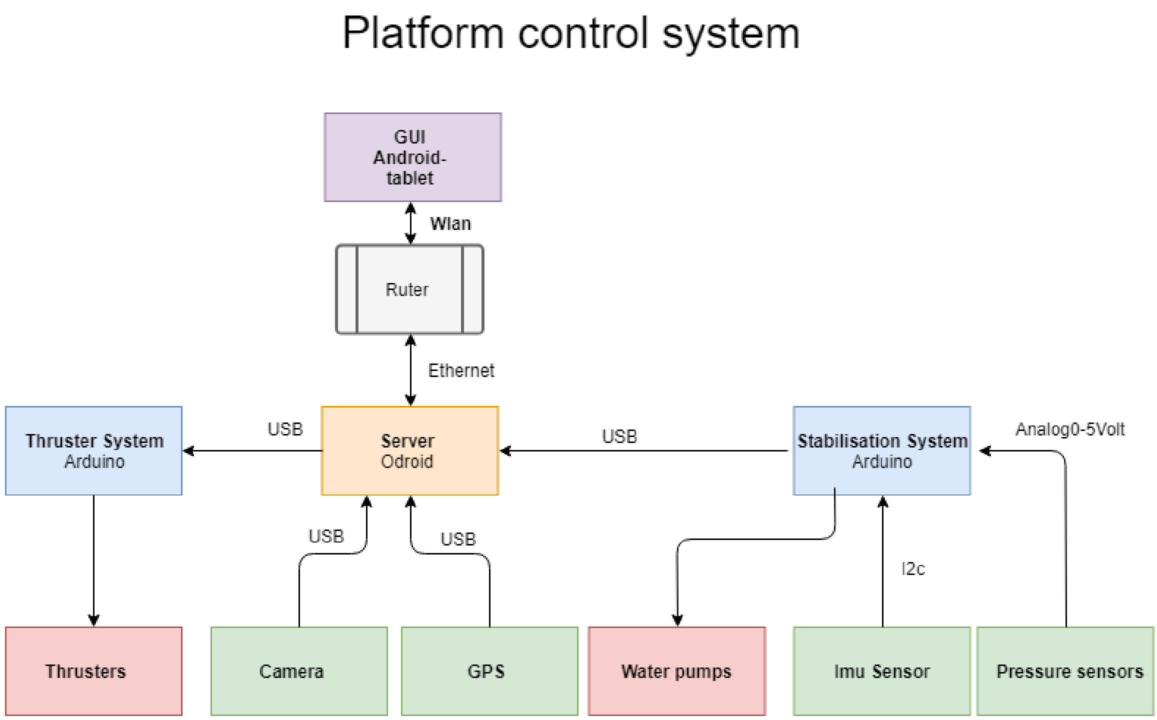


Figure 2.4: Old platform system layout

2.1.2 ROV

The ROV was designed and built autumn 2016 by the Mechatronics Lab at NTNU Ålesund [12]. The goal of the project was to build a low-cost ROV for Aquaculture application. Figure 2.5 shows the final prototype, and figure 2.6 shows the system layout. Inside of the ROV, there is a Raspberry pi and an Arduino. The Raspberry Pi is used for video streaming and communication with the GUI. The Arduino is used to read sensors, and controlling the thrusters. The ROV has in total 13 internal and external sensors. With a combination of camera, lights, cooling fan, oxygen, pressure, depth, moisture, temperature, gyroscope, compass, accelerometer and voltage sensors make it possible to monitor the environments both internally and externally. For remote control of the ROV, a GUI was made in Java.

Figure 2.5: ROV Prototype [15]

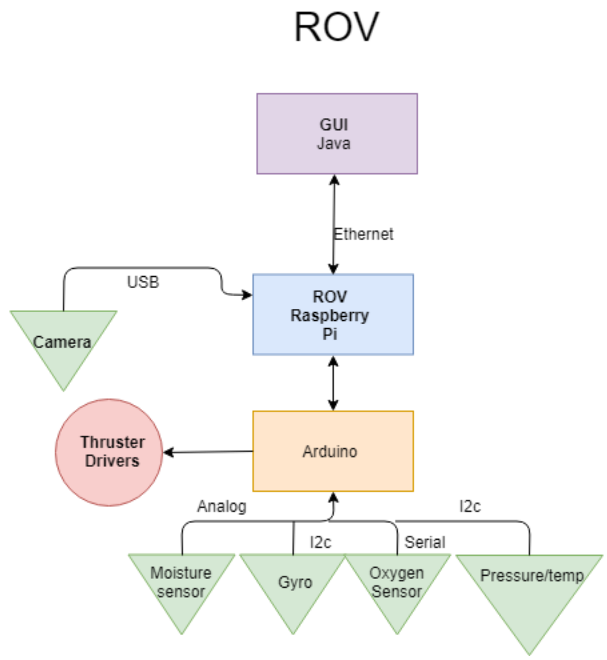


Figure 2.6: Old ROV System layout

ROV Design

The ROV has three thrusters mounted 120 degrees from each other. The ROV can move in the two-dimensional plane, by using vector calculation the force and direction of each thruster the ROV gets omnidirectional forces as seen in figure 2.7. The ROV has negative buoyancy and therefore, needs the winch to control the working depth while it is deployed [15].

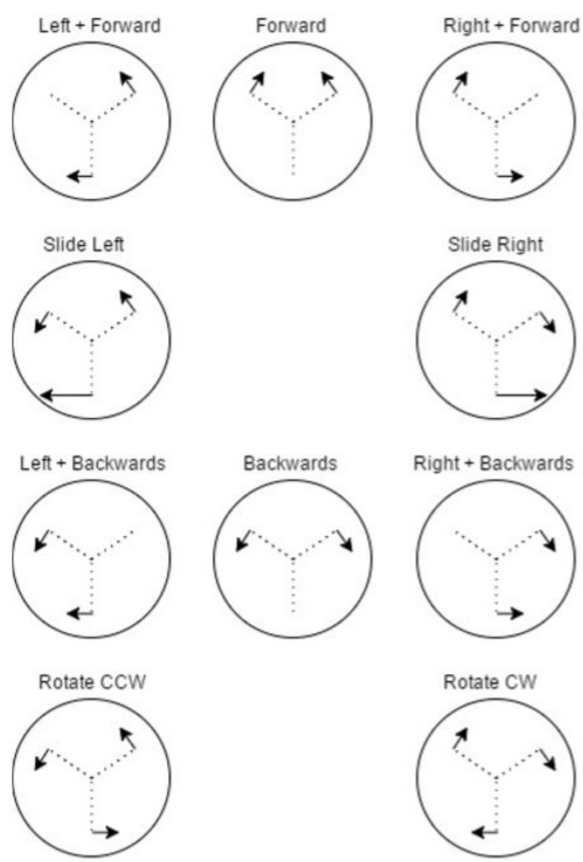


Figure 2.7: ROV Directions [15]

2.1.3 Winch

The winch for launch and recovery of the ROV also controls the depth that the ROV will operate on. The launch and recovery system is put together by the winch, spooling device and docking system, in figure 2.9 the winch and spooling device is shown. The winch is made by 3D printed parts and is controlled by a stepper motor. The spooling device is a sheave that can move freely on a stainless rod so the cable will be spooled correctly on the winch. The docking system for the ROV is there so when the ROV is not used it will be safely locked and not hanging freely under the USV. The docking system consists of two stainless steel rods for the mechanism to move along, and a lead screw to move the mechanism into the desired position. The lead screw is connected to a belt that is rotated by a stepper motor.

The design of the launch and recovery system is functional, but there is an issue with material quality. Several of the 3D printed parts were cracked or broken, as seen in figure 2.8. The structural integrity of the launch and recovery system is one of the problems that had to be addressed in this project.

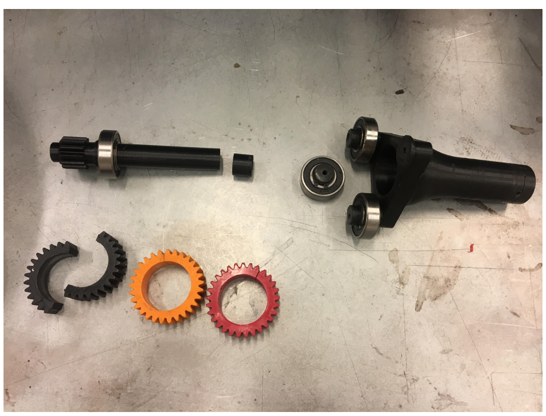


Figure 2.8: Broken parts from the winch

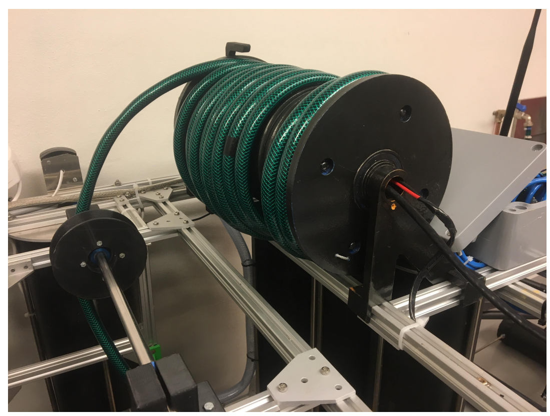


Figure 2.9: Winch on the USV

To control the launch and recovery system an Odroid and Arduino was used. By using server-client communication over TCP, the user would send a numbered command to the Arduino with the use of a text-based user interface as seen in figure 2.11. The Arduino then communicated over USB serial to the stepper motor controller as seen in figure 2.10.

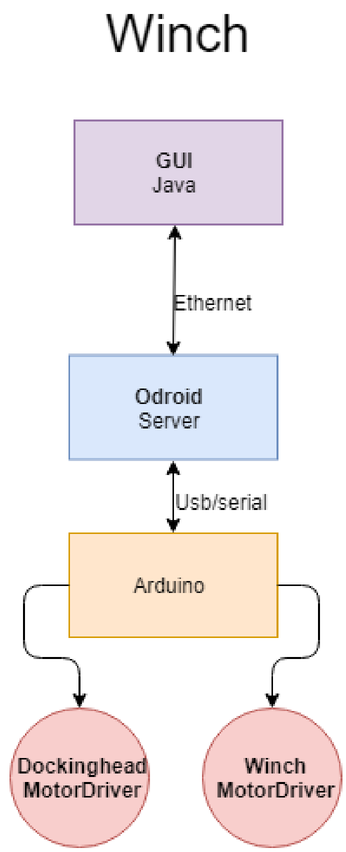


Figure 2.10: Old winch system layout

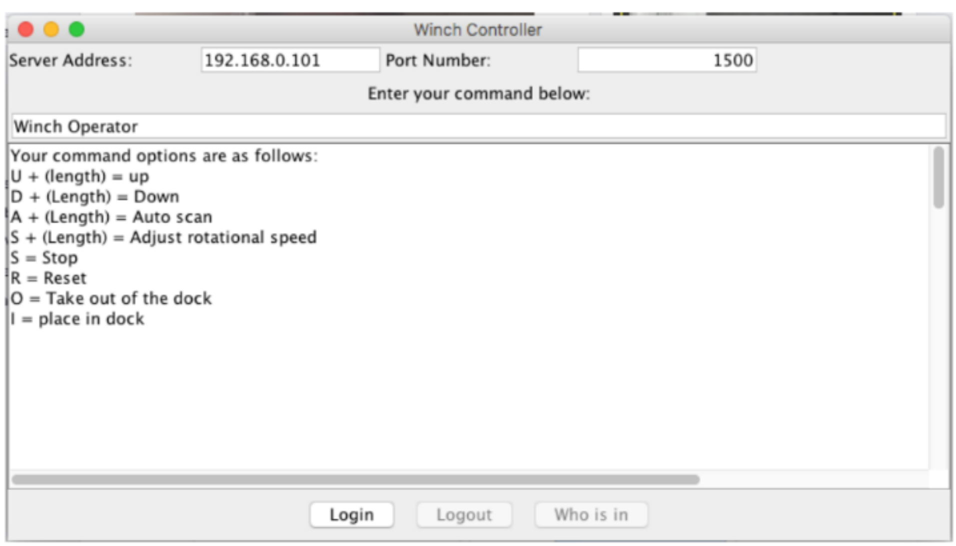


Figure 2.11: User Interface for the winch

2.2 Buoyancy and stability

2.2.1 Buoyancy

Buoyancy is the upward force that effects a submerged object. The force is determent by the volume of displaced fluid and the fluid density. In other terms, buoyancy force equal to the weight of the displaced fluid [3].

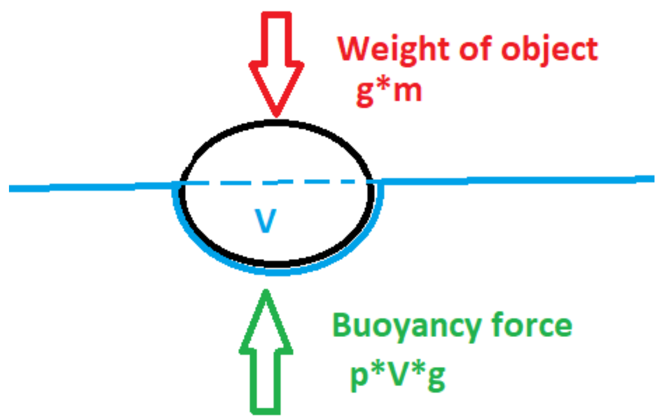


Figure 2.12: Buoyancy force

The buoyancy force is written like this:

$$F_b = \rho * V * g \tag{2.1}$$

where "V" is the submerged volume of the object.
The downward force or the weight of the object is written like this:

$$F_w = m * g \tag{2.2}$$

if $F_b > F_w$ then the object will rise if $F_b < F_w$ then the object will sink, and if $F_b = F_w$ the object is neutrally buoyant, and will neither go up or down.
When an object is neutrally buoyant we can write the sum of the forces like this :

$$\rho * V - m = 0 \tag{2.3}$$

2.2.2 Stability

Stability of a vessel is the ability to correct itself when outside forces are acting upon it in different directions.

To find the stability of a vessel, one must find four points that are used to calculate stability[3].

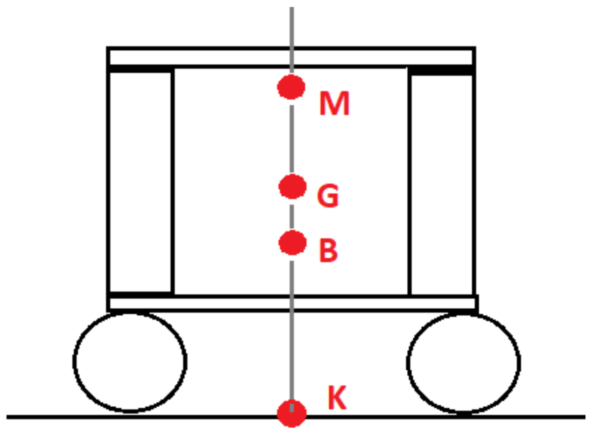


Figure 2.13: Pressure centers

Centre of gravity

Centre of gravity is the average location of the weight of an object. The point that gravity appears to act. The center of gravity will not move when the orientation of the object is rotated. The only way to move the center of gravity is to move around the weight on the object.

Centre of buoyancy

Centre of buoyancy is the average location of where the buoyancy force is acting on an object in liquid. The center of buoyancy will move when the orientation of the object is changed in the liquid.

In figure 2.14 it is shown that the center of gravity is always pushing downwards perpendicular to the surface of the earth, and the center of buoyancy is going upwards perpendicular to the surface of the waterline.

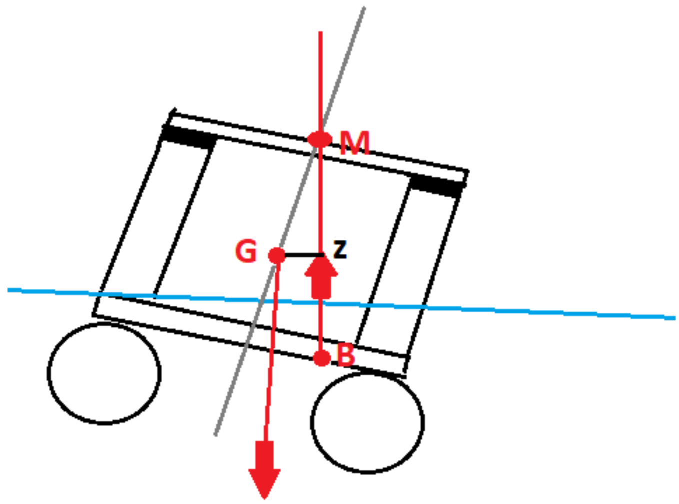


Figure 2.14: Change in center of buoyancy

GZ

GZ is the distance between the BM line, and the G point at the vertical height of G. If the point of gravity enters the BM line the vessel will not be able to turn back to the desired position since there will be no momentum. The turning momentum is dependent on the buoyancy force and the length of GZ. The length of GZ indicates the ability a vessel has to correct itself, with a negative GZ the vessel will become unstable and can topple itself. One can see that by lowering the center of gravity, the length of GZ would increase. Therefore it's desirable to have a low center of gravity.

Metacentre

In figure 2.14, one can see that where the line of the buoyancy intersects the line of the center of gravity is called the Metacenter and is indicated by the letter M.

GM

The distance GM is an indicator of initial stability. The larger the distance, the better the stability is. A vessel with a large GM is referred to as a rigid vessel.

• GM<0 vessel is unstable.

• GM=0 vessel is marginally stable.

• GM>0 vessel is stable.

2.3 PID Controller

PID is an automated method to control the input to a dynamic system. The P stands for the proportional gain and is proportional to the deviation of the desired output. The I stands for integral and produces an output proportional to the deviation over time. The D stands for derivative and produces an output proportional to the change in deviation over time. The sum of the P, I and D outputs will be the final output to the dynamic system. In figure 2.15 a typical PID-controller is shown.

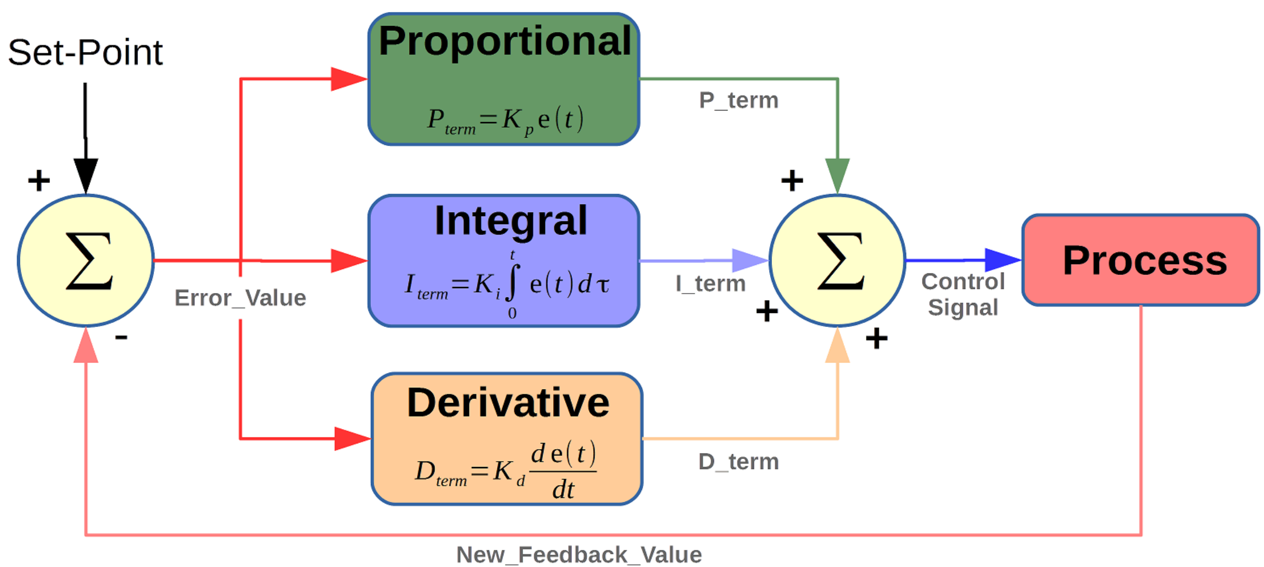


Figure 2.15: PID-controller [11]

2.4 Dynamic Positioning

Dynamic positioning (DP) is an automated system for a ship so it can use its thrusters and propellers to maintain a specific position and heading automatically. This is done by combining sensor data like wind, waves and current. A computer can use this to calculate how much power each thruster needs to output and at which angle in order to counteract the forces that are acting upon the ship away from its desired position.

There are three types of DP classifications [8]:

DP 1 means that a single fault in the system can make it shut down or not work properly.
DP 2 means that the system should not fail if a there is a single fault on an active component, like a thruster or generator, but it can fail if there is a fault on a static component like cables or a manual valve.
DP 3 means that every component have redundancy, and a single fault will not affect the system. This means that if a room is filled with water or damaged by fire, the system will still work as it should.

2.5 Autopilot

An autopilot will steer and maneuver the vessel to its desired position without human interaction. It still needs an human intervention if any unexpected events occur or when the operation completes. To have an autopilot, the vessel needs some functions [23]:

- Sensor to read yaw, roll, and pitch.
- Current position.
- Propulsion system.
- Controller performing the necessary maneuvering to reach the destination.

With these different functions combined, a functional autopilot is achieved.

2.6 Calculating Heading

Calculation of the heading would be used to maneuver the platform in the right direction. The formula for heading is[20]:

$$y = sin(\lambda_2 - \lambda_1) * cos(\varphi_2)$$

$$x = cos(\varphi_1) * sin(\varphi_2) - sin(\varphi_1) * cos(\varphi_2) * cos(\lambda_2 - \lambda_1);$$

Heading is then:

$$Heading = atan2(y, x)$$

Where start point is in radians:

$\varphi_1 = Latitude_1$

$\lambda_1 = Longitude_1$

and end point is in radians:

$\varphi_2 = Latitude_2$

$\lambda_2 = Longitude_2$

The heading have to be converted from radians to degrees.

2.7 Haversine

Haversine is used to calculate the shortest distance between two points on the earth's surface. This formula can be written like this[20]:

$$Radius = 6381e3$$

In radians:

$\varphi_1 = Latitude_1$

$\lambda_1 = Longitude_1$

$\varphi_2 = Latitude_2$

$\lambda_2 = Longitude_2$

and need the difference between the points in radians:

$\Delta\varphi = (Latitude_2 - Latitude_1)$

$\Delta\lambda = (Longitude_2 - Longitude_1)$

$$a = sin(\frac{\Delta\varphi}{2}) * sin(\frac{\Delta\varphi}{2}) + cos(\varphi_1) * cos(\varphi_2) * sin(\frac{\Delta\lambda}{2}) * sin(\frac{\Delta\lambda}{2})$$

$c = 2 * atan2(\sqrt{a}, \sqrt{1-a})$

$d = R * c$

d is the distance in meters.

2.8 Commuinication Protocols

2.8.1 TCP

TCP is placed in the transport layer in the OSI model. TCP ensures an end to end delivery of data packets and is using the routing method from the Internet Protocol(IP). TCP or Transmission Control Protocol is a connection-oriented protocol, which means that the connection is maintained until the nodes are finished communicating or the connection is lost.

A TCP connection starts with a three-way handshake.

1. Clients sends a request called "SYN"(Synchronize Sequence Number), to the server requesting the server to open a connection with the client.

2. If the server accepts, it will send back to the client "syn+ack" message telling the client that the message was received, and the server is ready. The SYN is the synchronize Sequence Number that the server will use, and the ack number is the Synchronize Sequence Number that the server received from the client plus one. ack=syn-client+1.

3. The client sends "ack"(syn-server+1) to the server to confirm that the connection is established.

TCP ensures the data integrity of the data packets by using a method called checksum.

2.8.2 UDP

User Datagram Protocol(UDP) is a communication protocol based on the IP network. UDP is often used as a communication protocol when transmission speed is a factor. This is because UDP does not have a handshake protocol between sender and receiver. UDP provides no guarantee of delivery of packets, duplication or loss of packets. When using UDP, the sender must specify the receiving port and IP address. UDP is often used for real-time applications like streaming, where the loss of a packet is not fatal for the receiver.

2.8.3 Modbus TCP

Modbus TCP is using the Modbus RTU protocol running over TCP interface using IP networking. Modbus is placed in the application layer in the OSI model. The Modbus protocol does not use checksum itself, but the underlying TCP/IP protocol does, which secures data integrity. The Modbus protocol uses master/slave structure were only the master can initiate data transaction. In Modbus TCP, the master is referred to as a client, and the slave is referred to as a server. Modbus is open source, meaning that developers can use the protocol without paying or needing a license. Modbus TCP is simple to implement in existing network equipment since it uses the TCP/IP protocol.

2.9 IMU

An Inertial Measurement Unit (IMU) is an electronic device used to measure acceleration, rotation, pitch, yaw, and roll. This is achieved by using accelerometers, gyroscope, and magnetometers. IMUs are used onboard aircraft's and unmanned vehicles so the software gets information about the movement of the vehicle.

Chapter 3

Materials and method

3.1 Project Organisation

The project group consists of a project leader, a secretary, and members. The project leader is in charge of writing the progress report and updating the Gantt diagram as progress is made. The secretary is in charge of booking meetings with the supervisors and writing a report from each meeting. Although there is a project leader, all decisions have been made by the group as one.

Throughout the project, meetings with the supervisors has been held every two weeks. During the meetings, the group has presented ideas to solve problems and discussed possible solutions with the supervisors. A project plan was made at the beginning of the project using a Gantt diagram. The project plan has been updated as tasks where completed. In order for all group members to have access to necessary documents and files throughout the project, google drive has been utilized.

3.2 System integration

System integration is the process of bringing different subsystem into one single system. When bringing the subsystems together, challenges will occur, and it is important to plan this process carefully [2]. Some of the questions that had to be addressed in this project were:

- Is it verified that the subsystem meets the requirements that are set?

- How will the subsystems affect each other when combined?
- What restrictions do the different subsystems have?
- Are the subsystem well documented?
- Are the systems using a common standard?
- Is it possible to simplify each subsystem?
- Is it less time consuming to redo the whole subsystem than to make changes to the existing one?

3.2.1 Testing the sub systems

At the beginning of the project, it was focused on researching each subsystem. The systems that were possible to test was tested with its old software and hardware. The test was done to check if the systems were meeting the requirement that was set, and so limitations of the systems could be found. The results of these tests are documented below.

Platform

The functionalities of the platform were not possible to recreate since the software was missing. Another limitation was that the extra weight that had been added to the platform by the winch and ROV would cause the platform to sink, more about this in section 3.5.1. The pipes on the bottom of the platform where the batteries were placed were not watertight, so the batteries inside were damaged.
The thrusters were tested by manually by connecting the motor drivers to a computer. All of the thrusters were working. The pumps that was used for pumping water in and out of the vertical pipes were also tested and confirmed working.

ROV

The software that was required for testing the ROV was available. The overall functionality of the ROV was working, but the response of the system was slow. Delays measured up to 2 seconds from a pressed button on the GUI until the actual movement of the ROV was recorded. As mentioned in the report of the ROV [12]. It was not possible to activate the camera stream at the same time as the rest of the system was running, the software on the ROV was programmed in Java, and these issues were most likely due to bad implementation of threading in the programming. The GUI was informational and working as intended. There was no sign of leakage inside the ROV; the thrusters and sensors were also working.

Winch

It was concluded early that the software made for the winch was too simplistic and could not be used in the final system. As shown in section 2.1.3 parts inside of the winch was broken and needed to be replaced with stronger parts. CAD drawings for every part of the winch were available.

Conclusion from testing

- Platform

The platform software was missing and cannot be used in the complete system. The thrusters and pumps were working. The buoyancy of the platform is not enough after the ROV, and winch was installed.

- ROV

The ROV is working, but changes in the software need to be done. The GUI was well made, but not designed for more than controlling the ROV.

- Winch

The code that was made to control the winch is not usable for the complete system. Parts of the winch needs to be upgraded with stronger parts that can handle the forces that are acting upon them.

3.2.2 Electronic installation

The electronic enclosures on the platform need to be watertight since the electronic components will get damaged if exposed to water, in figure 3.2 the water can easily enter the main switch and fuse box. The enclosures should also be organized inside since this will make the job of adding functionality and finding faults in the system easier. During the inspection of the old system, it was hard to find what the different components were connected to since all the wires and components were stacked upon each other in a couple of small junction boxes as seen in figure 3.1. The cables that connected the boxes was not suited for the application. The cables had too few wires inside, that resulted in lots of cables for a few signals. By removing the cables, and changing them to cables with the correct amount of wires inside, weight and space of the cables could be reduced.

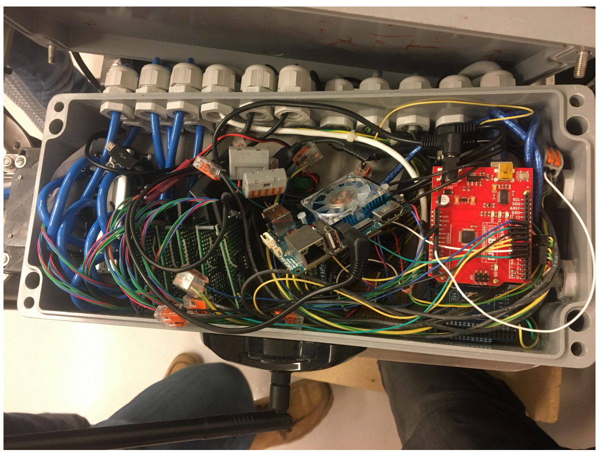


Figure 3.1: Old enclosure

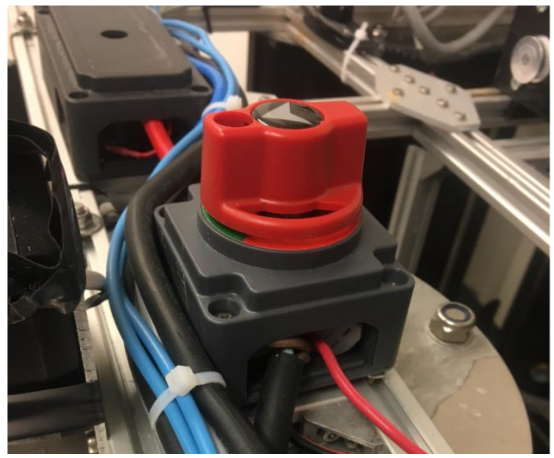


Figure 3.2: Old mainswitch

Due to the lousy cable management and the mess inside the boxes for the components, the group disconnected every component, and removed all the cables and boxes. Figure 3.3 shows the platform after it was stripped.

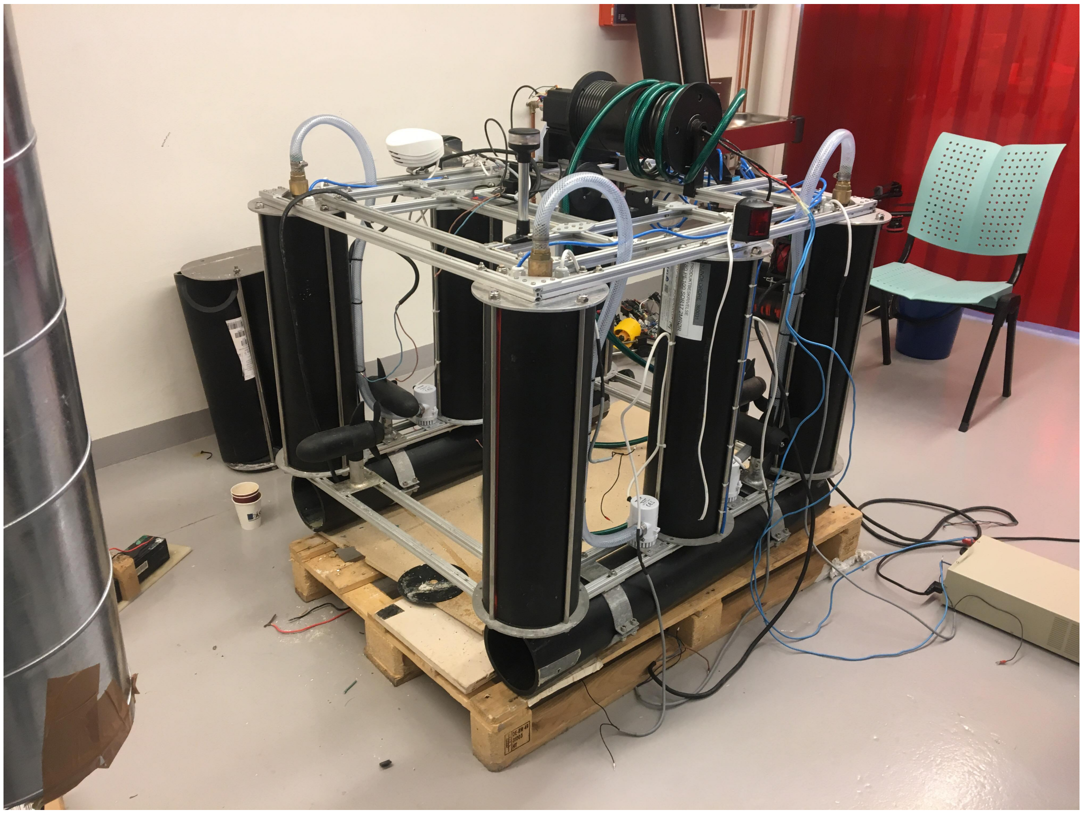


Figure 3.3: Stripped platform

3.2.3 Selection of Human-Machine Interface

The Human-Machine Interface (HMI) to the USV had to be remote as well as control and monitor the USV and the subsystems. This device should be able to send and receive data to the platform. The device must handle wireless communication, and the design must be user-friendly. With these requirements, some different solutions can be used.

Computer

By using a computer to control the USV and its subsystems, there are few limitations to what can be done. There are several types of software available to make a good GUI for the user to interact with. A computer can give the user visual information through the screen, and the user can interact with the system through the keyboard. In general, a computer does not have any I/O to have any analog inputs or outputs, but there are several ways for a computer to communicate using different communication protocols like WiFi or USB serial to the control system.

Computers come in all price ranges where the more you spend, the more powerful it is.

Tablet

Tablets are small computers with a touch screen for the user to interact with. To keep the size of the tablet as small as possible, they usually have a lightweight operating system, so the hardware does not have to be so powerful. This means that the battery capacity is good. Like a computer, a tablet cannot control I/O, but can communicate through WIFI. Tablets also have several software's where a GUI can be developed for the user to interact with, but unlike the computer, the screen is for both visual representation and for the user to interact with the control system. Since the user does commands directly on the screen, it opens up the possibility for more complex motion control through gestures on the screen.

Dedicated control station

With the use of hardware like a joystick, buttons, screen, and switches, the user gets good feedback on what they are doing. Since this system has to be wireless, there needs to be some casing to hold all the electronics as seen in. Each action the user performs must be registered by the control system using some I/O device. It is costly to build an HMI like this because there are components that are expensive as well as it takes time to make it. In figure 3.4 a possible solution for a dedicated control station is shown.

Figure 3.4: Dedicated control station

3.2.4 Selection of Graphical User Interface Programming Language

In general, there are two ways of programming a GUI. The first is a drag and drop platform where the user selects different widgets and drags them into a canvas, as seen in figure 3.5. The other method is to build up the GUI from scratch using code in an IDE like Netbeans or Eclipse. Some platforms have several types of software available so the GUI can be made by using the preferred method, others only have one of the methods available.

The advantage of using the drag and drop method is the ease of use while programming. This means that the time it takes to make the GUI often is shortened then when building it from scratch. One of the things that make it so easy to use a drag and drop method is that all widgets are premade, all the user have to do is select the desired widget and place it. One drawback by making a GUI with the drag and drop method is if the software doesn't have the right widget for the GUI. How many widgets a software has vary, but those who have a good selection of widgets are often expensive.

With the use of an IDE, the programmer can build a GUI from scratch. This often takes longer time than drag and drop method because the user must make all buttons, text, and visualization. There are functions for adding several GUI functions like buttons and text, but they are highly customizable to what the user needs. It is also necessary to place all the objects to the layout in the code, this is done by specifying at a pixel level. One of the advantages by building the whole GUI from scratch is that the final product can be just the way you want due to its high customizability.

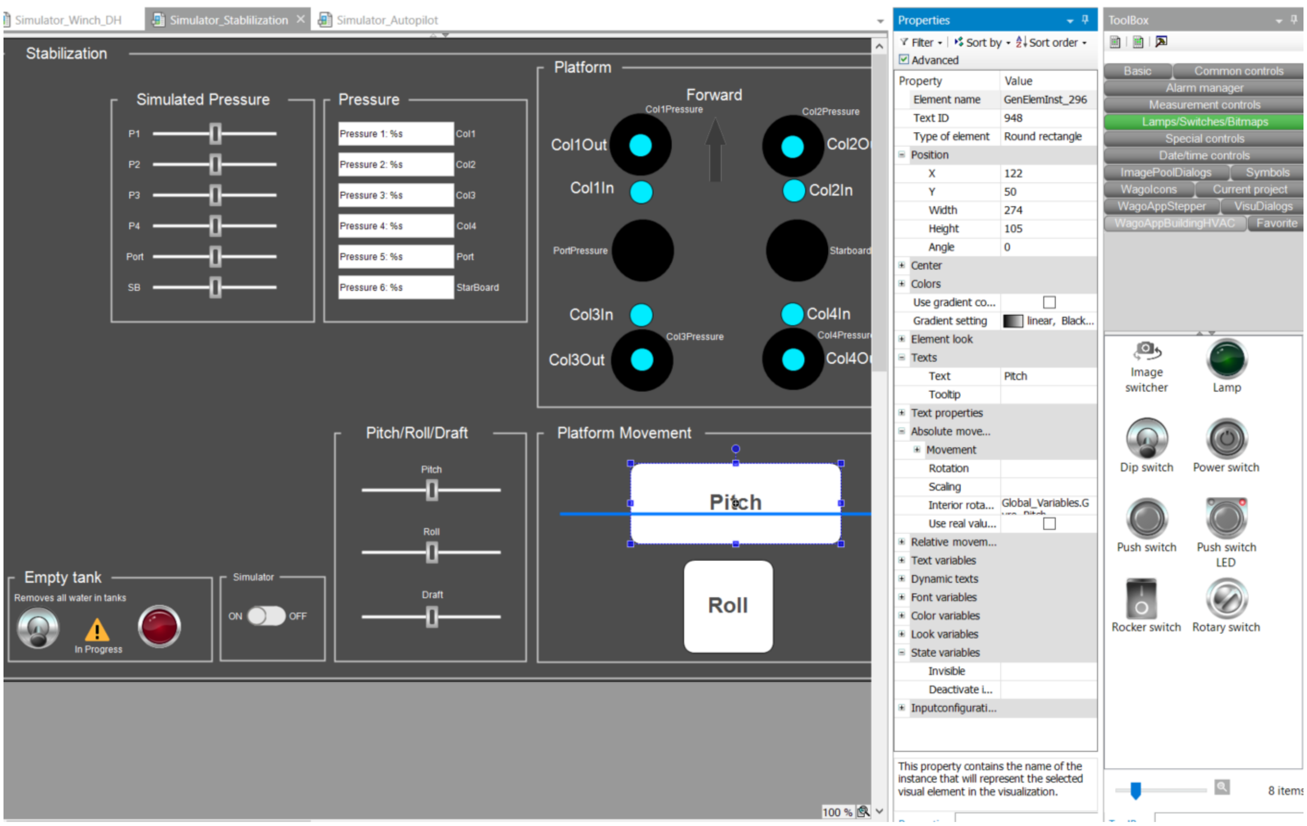


Figure 3.5: GUI for simulation in e!cockpit

In this project, the GUI was made using JavaFX because it's flexibility and because the group had some experience using Java from previous work. Java also supports the implementation of the video stream, which is an essential factor of the GUI. It also has support for a wrapper called JXmaps [10] which implements Google maps into the GUI. When programming the PLC, an interface was made using e!cockpit to simulate the platform. This was done using the drag and drop method. The final result of the GUI will be presented in chapter 4.6.1.

3.2.5 Selection of Control system platform

A control system operates, manages, and monitors a series of devices ranging from a simple heater to advanced industrial systems. In this case, the control system must handle all I/O on the USV and communications with the subsystems as well as the HMI. The control system should be robust, flexible, and have options for extension of the functionality. Below are different solutions that can be used.

Micro Controller

A microcontroller supports a range of functionality by the use of library extensions and additional hardware. The microcontroller can read and write I/O, both analog and digital values. With the use of additional hardware, the microcontroller can support several of the requirements for this project like GPS, IMU, and stepper driver. Microcontrollers and the additional hardware are relatively accessible and cheap to buy.
A drawback with the use of microcontrollers is the additional hardware that's needed to achieve the desired functionality. This means the cabinet would be filled with components which increases the chance of interference from electromagnetic noise. Most of the hardware is not designed for plug and play connection so they must be connected by wires or communicate through a bus. This increases the chance of fault on devices.

PLC

A PLC is an industrial computer designed for use in the manufacturing industry, ease of programming and diagnostic. The PLC itself is a computer with some added connection point for communication and power. In order to have more functionality, I/O modules must be added. These modules are attached to the side of the PLC, and the number of modules is limited to how much the power supply can deliver. The modules are plug and play and communicate through a communication bus integrated into each module. A PLC and modules are expensive to buy, but most of the modules needed in this project can be located at the university. Some existing hardware on the USV like the GPS and IMU can't be used with the PLC. This is because they work on different communication protocols. These components could be replaced by some that are compatible with the PLC, but it is costly.

Single Board Computer

A single board computer(SBC) is a small computer with integrated I/O on the board. The SBC has most of the same functionality as a regular computer like memory, WIFI, Ethernet, USB, and Bluetooth, but on a single board. An SBC can be used to stream a video feed from a camera with it's integrated USB port as well as it supports the GPS and IMU already mounted on the USV. As the microcontroller, not all functionality can be achieved by using only an SBC. Therefore additional hardware may be required.

3.3 Software

During this project, the following software listed in table 3.1 have been used:

NR	Software	Version	Supported-Operatingsystems	Usage
1	MatLab	R2017B	*Windows,linux,Mac*	*Calculations, and plotting graphs*
2	Netbeans IDE	8.2	*Windows,linux,Mac*	*Java programming*
3	Arduino IDE	1.8.9	*Windows,linux,Mac*	*Programming Arduino and gyro sensor*
4	E!cockpit	1.5.0.3	*Windows*	*Wago PLC programming IDE*
5	PyCharm IDE	2018.2.4	*Windows*	*Python programming IDE*
6	Siemens NX	11.0	*Windows*	*Creating CAD drawings*
7	Autodesk Fusion 360	2.0.5519	*Windows,Mac*	*Calculations on CAD drawings*
8	Putty	Release 0.70	*Windows*	*Connecting to devices over SSH*

Table 3.1: Software used in this project

3.3.1 Libraries

In this project, some libraries were used to achieve some functionality, these libraries are listed in table 3.2.

Libraries used				
Nr	Library	Platform	Usage	Version
1	JxMaps	Java	Implement Google maps in the GUI for visualization of position and to plot route for autopilot and DP.	1.3.1
2	Medusa gauge	Java	Used to make gauges in the GUI to visualize sensor data from the platform [16]	8.0
3	OpenCV	Java	Used to do image processing on the video stream	3.4.5
4	EasyModbus.java	Java	Implements modbus TCP in Java to communicate with PLC	5.5
5	OpenCV-python	Python	Reading from USB-camera on Raspberry Pi, and encode images before sending them.	4.00.21
6	Pyserial	Python	Used for reading from the serial port on Raspberry Pi.	3.4
7	PyModbus	Python	Used to communicate with other devices using Modbus in python.	2.1.0

Table 3.2: Libraries used in this project

3.4 Data

When combining three different systems into one single system, one of the challenges is to choose a communication protocol, and cable standard that is supported by all of the systems. Dependent on the type of data transferred, there are different priorities.

3.4.1 Video Streaming

Depending on the application, different transport protocols are suited for video streaming. Often when streaming stored video, and it's not important that the video is live, TCP can be used. The problem with TCP is that if there is no buffer on the client side, the video can stutter, or stop for a few seconds for the viewer since TCP demands acknowledge on each package to secure that the client has received the correct data. If more than one clients are connecting to a live video stream, it will become demanding on the Server workload, since the server has to buffer unacknowledged segments for every client.

UDP is more commonly used in live video streaming since the UDP does not care about package drops, the video will be with less delay, the downside is that with bad bandwidth on the network, the picture can become of low quality, and tearing in the image can occur [9]. UDP also support IP multicast, which makes it possible to stream to multiple devices without extra workload on the server, TCP does not support IP multicast.

There are some other protocols also that is commonly used for video streaming, for example, RTSP(Real-time streaming protocol) RTSP is commonly used in IP cameras, and is simple to set up on the Raspberry pi. With experience from an earlier project, the delay with RTSP is still a bit too long(3-4S).

3.4.2 Sensors and commands

When transferring sensor data, it's important that the data is correct and dependent on the use of the data, is speed also important. The TCP protocol is suited well for sensors and commands since TCP guarantees that the receiver receives all of the data and that the data is correct. UDP can also be used, but the receiver should check that the data is correct.

3.4.3 GPS data

To be able to know the position of the USV, GPS is needed. To get high accuracy by using GPS, it is important to choose a GPS that supports EGNOS, which is the use of antennas on land to get significantly better precision. Most GPS devices transmit their data in a protocol called NMEA sentence, the protocol was developed by an organization called National Marine Electronics Association. The NMEA sentence is not a strict protocol meaning that there is allot of different versions of the NMEA sentence, but they all have some aspect of similarity to each other.

3.4.4 Compass and orientation

The GPS is not able to deliver a stable heading of the platform alone. A compass for the system is needed to be able to know which direction the USV is facing relative to north. There are two methods for finding heading of the platform. A gyroscope can measure the changes in the heading, but the gyroscope will drift over time, resulting in wrong heading values. As well as measuring heading(Yaw), the gyroscope can measure the roll and the pitch of the platform. A magnetometer is using the magnetic field to know its heading but needs calibration since it is sensitive to magnetic noise. A combination of both of these sensors will give an accurate measurement if the magnetometer is calibrated to its surroundings. Figure 3.6 shows the platform orientation.

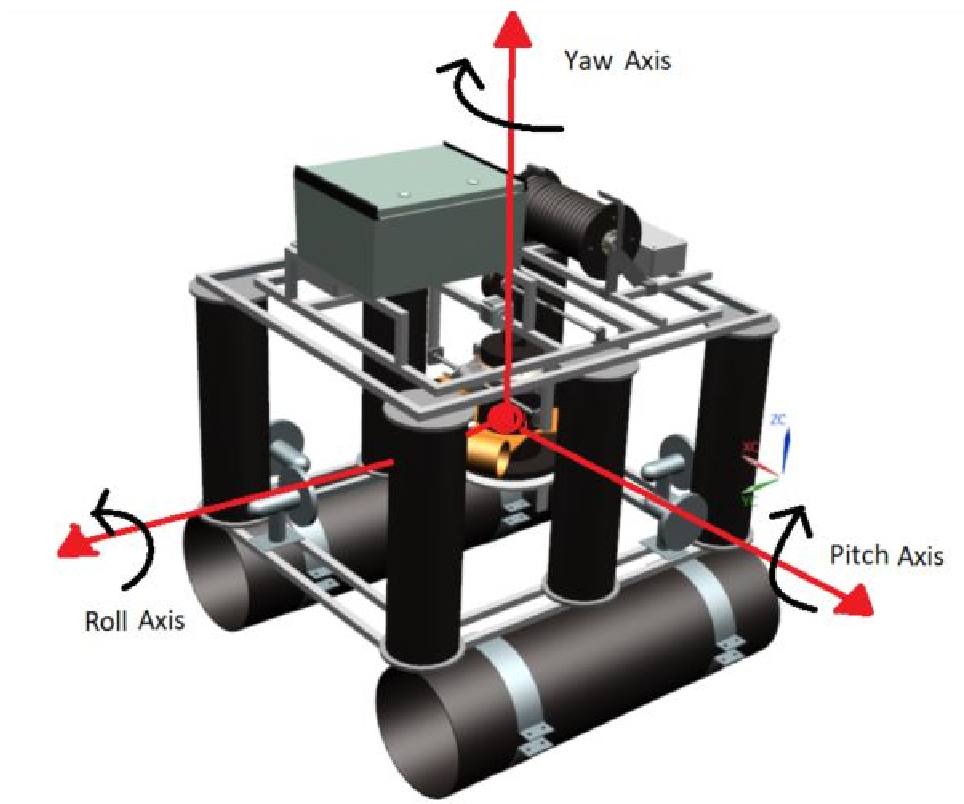


Figure 3.6: Platform orientation

3.5 Design and modeling

3.5.1 Buoyancy

With the added weight from the ROV and winch, there came an issue with the buoyancy. Video material from the previous project showed that the USV was low in the water. The calculation below shows that the platform does not have enough buoyancy as it is.

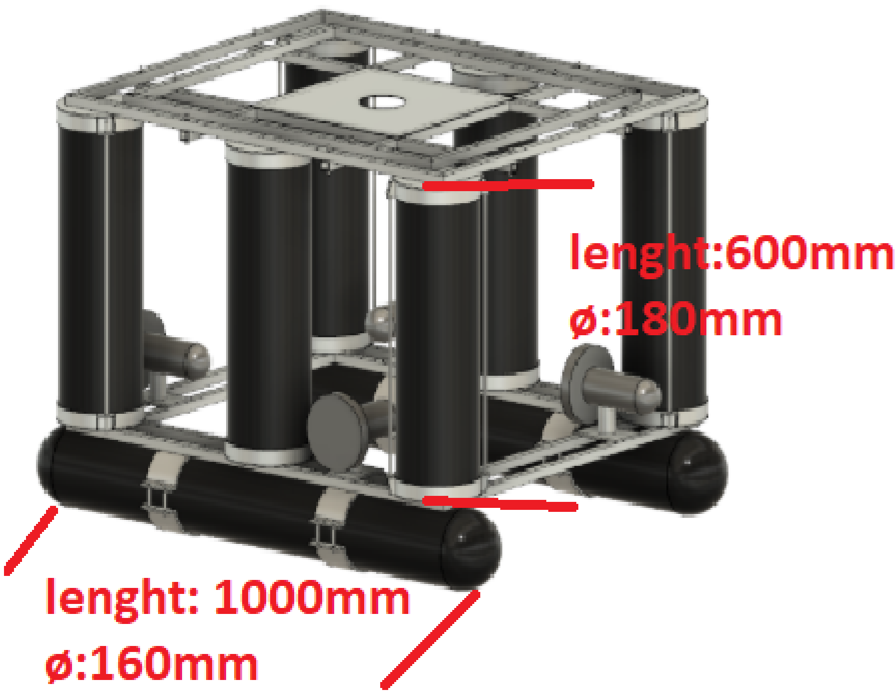


Figure 3.7: Previous pipe size

The mass of the platform before the ROV and the winch was installed was 115 kg.
which gives a downward force of $m * g = 115 * 9.81 = 1128.15N$
If we assume that the fluid density of the saltwater = $\rho = 1025Kg/m^3$ the volume of water the platform needs to displace = $1128.15/(1025 * 9.81) = 0.11m^3$

For a pipe the formula for volume: $V = \pi * r^2 * h$ the volume of the two bottom pipes is then given by : $\pi * 0.08^2 * 1 = 0.0201m^3$ the total volume of the two pipes is then $0.0201 * 2 = 0.0401m^3$

The volume of the thrusters is found by submerging them in water, and measuring how much the water rises.

Figure 3.8: Finding volume of thrusters

The result of this test was that the thruster displaced 0.6 liter of water. giving it a volume of : $V_t = 0.0006 m^3$

this means that the six vertical pipes have to cover a volume of : $0.11 - 0.0401 - 0.0006 * 4 = 0.0675 m^3$

the height of the water on the vertical pipes is then: $h = 0.01125/\pi * r^2$

442 mm up on the vertical pipes is then below water. Leaving the waterline 158mm to the top of the platform. This estimate is without counting in the volume of the frame, but this volume is negligible since the frame is made of aluminum profiles with holes. A function comparing the weight of the platform versus waterline on the vertical pipe is shown in figure 3.9.

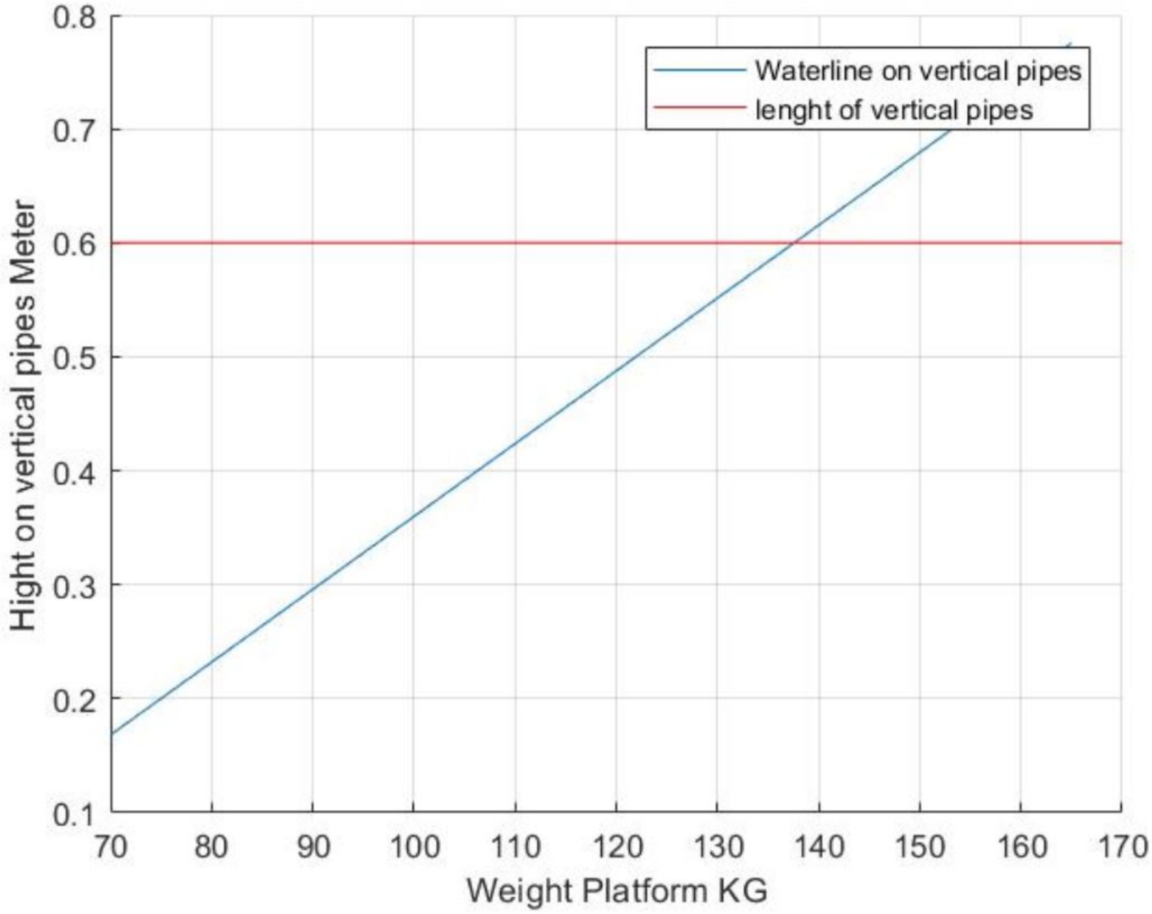


Figure 3.9: Height of the waterline based on weight

The weight of the platform after the ROV and winch were installed was measured to 156 Kg. According to figure 3.9 the platform would then sink without extra Buoyancy.

Adding Buoyancy

When adding buoyancy, there were a couple of alternatives that were considered.

- Increasing the number of vertical pipes.

  Increasing the number of vertical pipes on the sides of the platform would be hard since the thrusters are in the way. The winch group [14] had installed new pipes in the front and the back of the platform, but this is not a desirable solution since this will increase the drag force of the platform during voyage. The platform would also become more sensitive to waves since the area of wave impact would increase.

- Increasing the size of the vertical pipes.

  By increasing the size of the vertical pipes. By increasing the size of the vertical pipes, the buoyancy would increase allot with relatively small changes in diameter since:

$$V = \pi * r^2 * h \tag{3.1}$$

  The downside of this solution is the amount of work that this involves. There is little space between the thrusters and pipes. The frame of the platform would then have to be re-designed.

- Increasing the number of bottom pipes.

  A simple solution is to add another pipe on the bottom of the platform. This could have been done with a relatively low amount of work. The downside of this is that by adding more buoyancy on the bottom of the platform, the platform could become unstable.

- Increasing the size of the bottom pipes. Increasing the size of the pipes on the bottom would be a better alternative than to add new pipes on the bottom since this would open the possibility to add bigger size batteries in the pipes.

- Rebuilding the platform to a different design.

  Rebuilding the structure of the platform completely would be a good alternative, then the platform could be better suited to fit the winch and ROV. The platform could be wider and longer, making space for longer and vertical pipes witch would increase the Buoyancy. Doing this would require much work and time.

3.5.2 Calculating the new pipe size

In section 3.5.1, the different options for increasing the buoyancy was presented. Since the batteries on board the platform had low capacity, increasing the size of the batteries was desirable. To fit the new batteries, bigger pipes were needed on the bottom of the platform.

To find the estimate on how much buoyancy that was needed, a weight estimation was done. In the winch report [14] it was documented that the platform with the winch and the ROV weighed

156Kg. The current batteries weigh 16 kg, the platform would weigh another 45 kg if the batteries were replaced with four batteries weighing 15 kg each. Giving the new weight of the platform an estimated mass of 200 KG.

When calculating the required buoyancy, a requirement for the water level with no added water in the vertical pipes was set. The water level should reach up to the thrusters even if the tanks are empty. The top of the thrusters is located 20 cm up on the vertical pipes. $F_W = 200*9.81 = 1962N$ the platform would need to displace $1962/(1025*9.81) = 0.1951m^3$ water to float.

By calculating the amount of water that is displaced by the vertical pipes and the thrusters when the platform is submerged with the water level at the top of the thrusters, the extra needed volume can be calculated.

Volume vertical pipes 20 cm: $V_v p = \pi * 0.08^2 * 0.20 * 6 = 0.0241m^3$

By subtracting the volume of the thrusters and the volume covered by the vertical pipes, each new pipes have to cover a volume of $0.1951 - 0.0241 - 0.0006 * 4 = 0.1686/2$ with a length of the pipe as the same as the old ones, the new pipes need a radius of:

$r = \sqrt{\frac{0.0843}{\pi * 1}} = 0.1638m$

3.5.3 Placement of instruments

Control cabinet

There is little space for the control cabinet, but it should be placed as much in the center of the platform as possible. This is to distribute weight equally on all four corners of the platform. The cabinet should not be in danger of being submerged in the water since this would damage the electrical equipment inside. With this in mind it is desirable to have the cabinet as far away from water as possible, but at the same time try to keep the center of gravity low as it should not be too high.

Position sensors

For the IMU to be as accurate as possible, it should be placed as close to the center of mass as possible, if the IMU is placed far from the center of the platform, it can detect upward pitch as downward acceleration as an example, but this can be calibrated in software. The IMU is sensitive to EMC and should be placed alone in its own enclosure, so it's not affected by the noise from the other components inside the control cabinet. The GPS antenna should be placed in the center of the USV, and with no obstructions to get the best possible GPS signal.

Battery

The placement of batteries plays a significant role in where the center of gravity of the platform is located since the batteries are the heaviest components onboard. To increase the stability of the platform, the lower the placement of the batteries, the lower the center of gravity, as seen in section 2.2.2. The horizontal pipes under the platform are the lowest point that can be made watertight. Therefore it is where the batteries should be placed.

3.6 Simulation

Simulation is the process of imitating real life situation virtually. Simulation is used to test functions and checked that it is working as expected without the physical system connected. The advantage of this is that it is possible to find bugs and error without having a physical product to test it on. Simulation is also used to optimize functions and methods that are used and can help the programmer to understand the system better by having a visual representation of what is happening in the software. In this project, it was built a simulator in e!cockpit to simulate different situations the inspection platform will be exposed for, without having to place the platform in the water.

Simulation with e!cockpit

With the use of a PLC, this opened the possibility of creating a simulator in the PLC programming tool e!cockpit. In e!cockpit, it is integrated a visualization tool that can be used to build and simulate different input and output signals and values. By building a simulator, it gives the opportunity to test the program and functions before the system is built physically. This leads to efficient and smart programming. It was made four different simulators to simulate the different tasks.

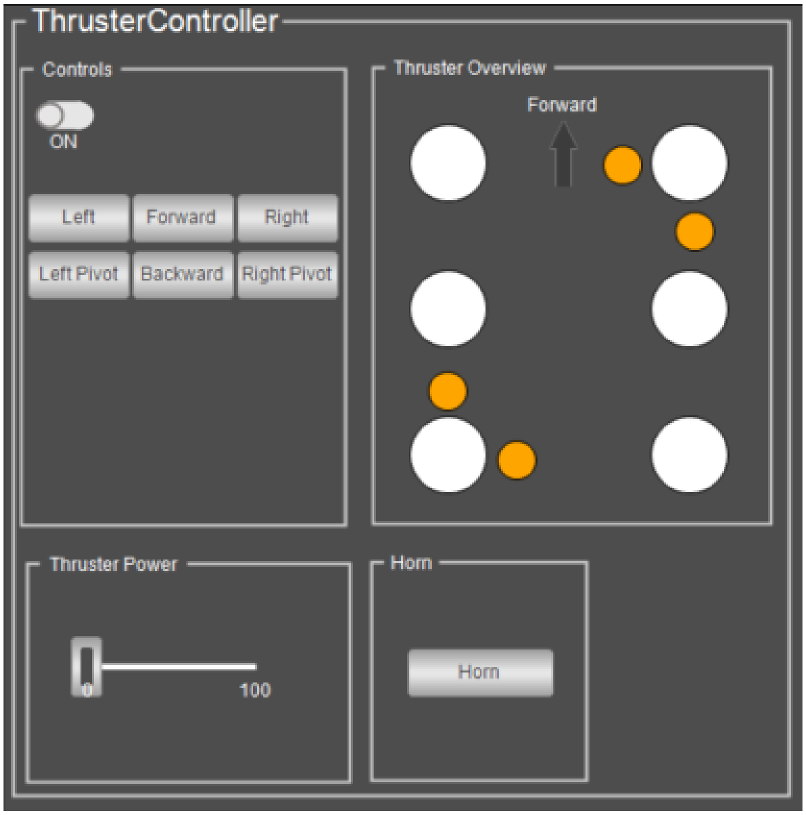


Figure 3.10: Simulator thruster controller

Figure 3.10 shows the simulator to the thruster controller. This simulator made it possible to simulate and control the thrusters before the physical system was done.

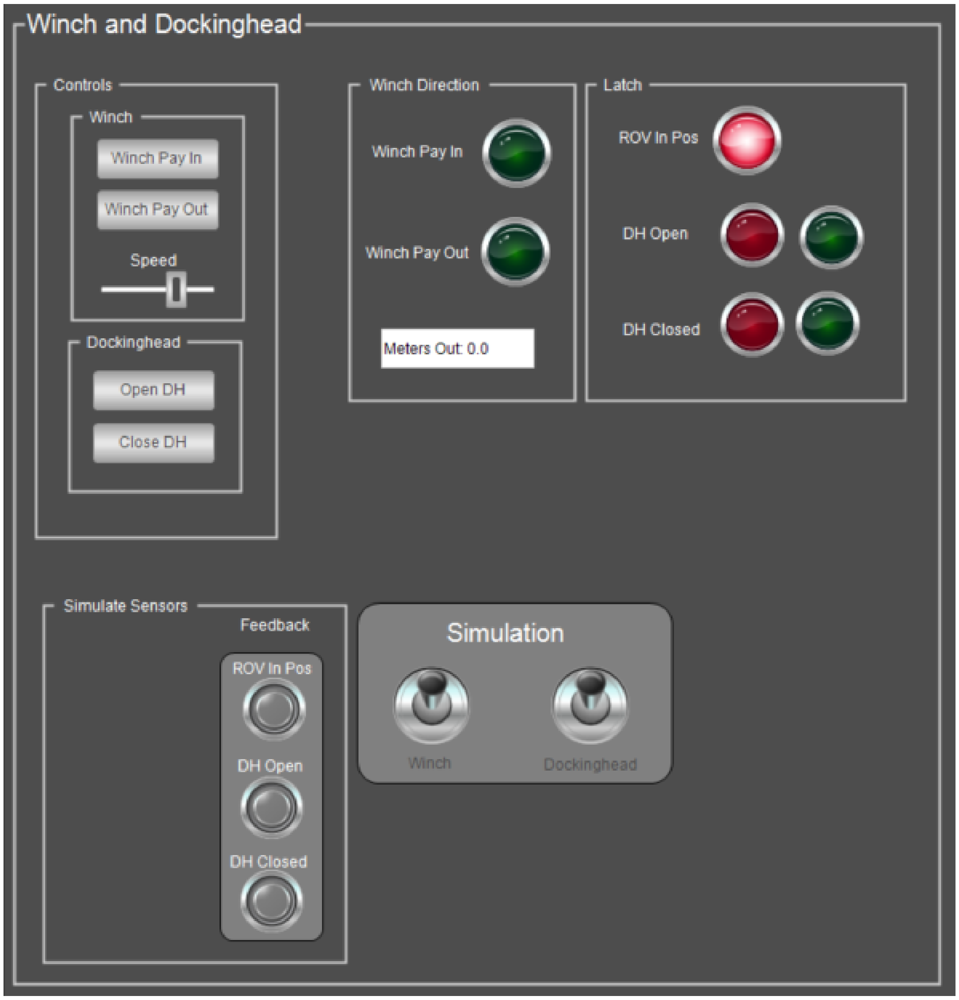


Figure 3.11: Simulator Winch and Dockinghead

Figure 3.11 shows the simulator used to simulate the winch and dockinghead. This was used to simulate when the ROV was in position and open/close the dockinghead.

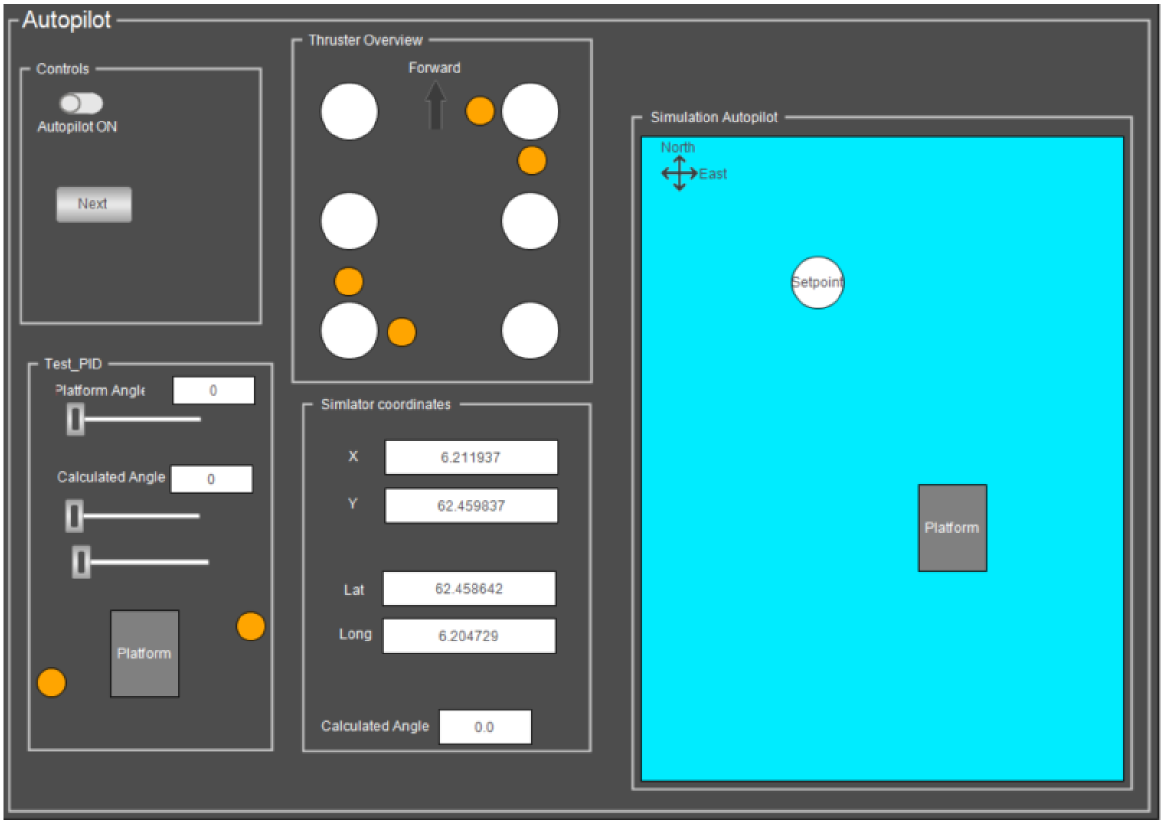


Figure 3.12: Simulator

Figure 3.12 shows the simulator used to simulate autopilot. Different coordinates were plotted to simulate the GUI. Then the heading was calculated and simulated to see if the platform would move as expected.

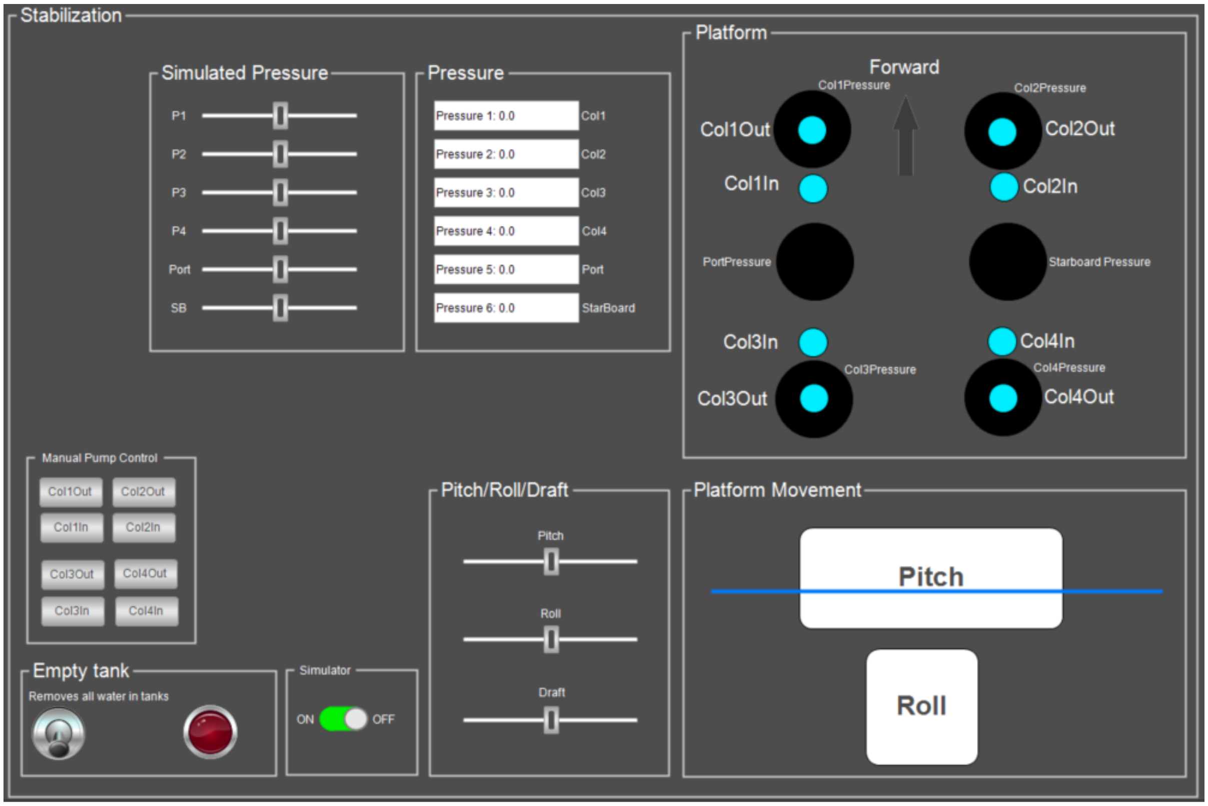


Figure 3.13: Simulator Stabilization system

Figure 3.13 shows the simulator used to simulate the stabilization system. Here the pitch, roll, draft, and pressure could be manipulated to simulate the system. The simulator will show when the pumps start/stop depending on the different outcomes from the manipulated values.

3.7 Materials

3.7.1 Additive Manufacturing

Additive Manufacturing (AM) or also called 3D printing describes the technologies that build 3D objects by adding layer-upon-layer of material [4]. 3D printing use Computer Aided Design (CAD) to build 3D models. In this project, 3D printing has been used to make parts to the winch system. Different material can be used when 3D printing parts. The choice of material depends on what the part is going to be used for.

The advantages of using 3D printing are that it is easy to use, and the models will be built with high accuracy. Disadvantages are that parts don't have high mechanical strength. That means that using 3D printed parts to build brackets to, for example, sensors will be a good solution but not a good solution to use if the parts will be exposed to high mechanical load. Table 3.3 shows two different printing material that is available at school.

PLA vs PLA Though	
PLA	Good tensile strength Good surface quality Easy to work with at high print speeds User-friendly for both home and office environments Allows the creation of high-resolution parts Ideal for models and prototypes that require aesthetic detail Great for lost casting methods to create metal parts A wide range of color options available Prints in dual extrusion with PVA or Breakaway
PLA Though	Impact strength similar to ABS, greater than regular PLA Higher stiffness compared with ABS Less brittle than regular PLA Gives a more matte surface finish quality than our normal PLA Suitable for post-processing with improved machinability compared to regular PLA More reliable than ABS for larger prints, with no delamination or warping Compatible with Ultimaker support materials (PVA and Breakaway) giving full geometric freedom when designing parts

Table 3.3: PLA [18] vs PLA Tough [19]

3.7.2 Parts/Equipment

In table 3.4 below the parts and equipment used in this project are listed.

NR	Type	Description
1	GPS	Ublox Waterproof USB 50 Channel GPS Receiver
2	IMU	The SparkFun 9DoF Razor IMU M0 features three 3-axis sensors; an accelerometer, gyroscope, and magnetometer. These sensors give the ability to send linear acceleration, angular rotation velocity, and magnetic field vectors. [17]
3	Pressure sensor	MPX2010DP pressure sensor. This is a piezoresistive pressure sensor provide an accurate and linear voltage output directly proportional to the applied pressure.
4	Microstep Driver TB6600	Microstep Driver TB6600 for two-phase stepper motors. It can output a 4A peak current. Supports PUL/DIR and CW/CCW modes 5VDC.
5	Microstep Driver DM556	Microstep Driver DM556 for two-phase and four-phase motors. It can output current from 0.5A to 5.6A. Supports PUL/DIR and CW/CCW modes.
6	Pololu Simple High-Power Motor Controller 24v12	Pololu Simple High-Power Motor Controller (SMC) is a basic control of brushed DC motors. Power input is 5.5V to 40V, and it delivers a continuous 12A. It supports four interface modes: USB, TTL Serial, analog voltage, and radio control.
7	Bilge pump	Bilge pumps from Biltema that use 12V/2.5A and delivers 32L/Min.

8	Omron Inductive Sensors	Inductive Sensors are used for non-contact detection of metallic objects. Their operating principle is based on a coil and oscillator that creates an electromagnetic field in the close surroundings of the sensing surface [6].
9	Haswing 20	The Haswing 20 is an electric outboard motor. The motor is rated to 9Kg of thrust and consume 17 Amp at max speed.
10	Raspberry Pi 3B+	The Raspberry Pi 3B+ is a single-board, low-cost, high-performance computer.
11	Stepper motor 57BYGH420	Stepper motor 2A
12	Stepper motor PD86-3-1180	Stepper motor 5.5A
13	Linksys Archer c5 v2	Router
14	Surface Go	10" Microsoft tablet
15	Camera	Webcamera

Table 3.4: Equipment

Chapter 4

Result

In section 3.2, the challenge to bring several subsystems into one complete system was introduced. In this chapter, the result and decisions of the system integration will be presented. During system integration, it has been done changes to the control system, platform design, winch system, and ROV system. During the sea trial, several tests have been performed to test the new control system. In the next section, the decisions made to integrate all three systems will be presented. After that, the buoyancy issue with the platform will be carried out, and finally, the result from the sea trial will be shown.

4.1 Platform Design

4.1.1 New pipes

In chapter 3.5.2, it was calculated that the platform needed pipes with a diameter of 326mm. The closes that could be found had a diameter of 315mm. Since the new pipes had a larger diameter than the older pipes, the new pipes had to be moved 150mm out from the platform so the ROV could fit between the pipes. New clamps for the pipes were made by bending aluminum plates and welding them to aluminum brackets. The clamps were fastened to the frame by using stainless steel 8mm bolts.

Mounting of batteries

Since the old batteries pipes leaked in water, a new concept of waterproofing the pipes had to be made, and it should be possible to inspect the insides of the pipes for leakage. The batteries were placed on a Plexiglas plate inside the pipes. The plate was placed 5 cm up from the bottom of the pipe. This was done so if water gets into the pipe, there is time to save the batteries before they are exposed to water. The batteries were glued to the plate with sealant glue.

By laser cutting Plexiglas, new end plates on the pipes were made. Two of the plates was fitted with a cable gland with an IP grading of IP68, which means it is dust tight and can be permanently submerged in water up to and deeper than one meter. The end plates were glued with sealant glue and laid overnight with 15Kg of pressure.

4.2 Re-engineering of the three systems to one common system

To integrate the three systems into one control system, it was desirable with a robust and flexible system. For the control system to the platform and the winch system, the hardware used in this project and their requirements neither control systems proposed in section 3.2.5 met all the requirements. It was chosen to use a combination of PLC and SBC for the USV and winch system. The PLC is the best candidate to control the USV and winch due to its modular design, flexible and visual programming, and the possibility to view live values while testing. The SBC gives some functionality that the PLC can't offer, like the ability to stream a video feed, read GPS signal, and communicate with the IMU. These two controllers together give good flexibility and reduce the number of controllers from the previous control system. The ROV needs a control system that can communicate with the HMI and the rest of the control system. The choice was to use an SBC that is compact and can read and write I/O as well as communicate with the HMI and PLC through Ethernet. In figure 4.1, the new complete integrated system layout is shown and figure 4.2 is the new USV design.

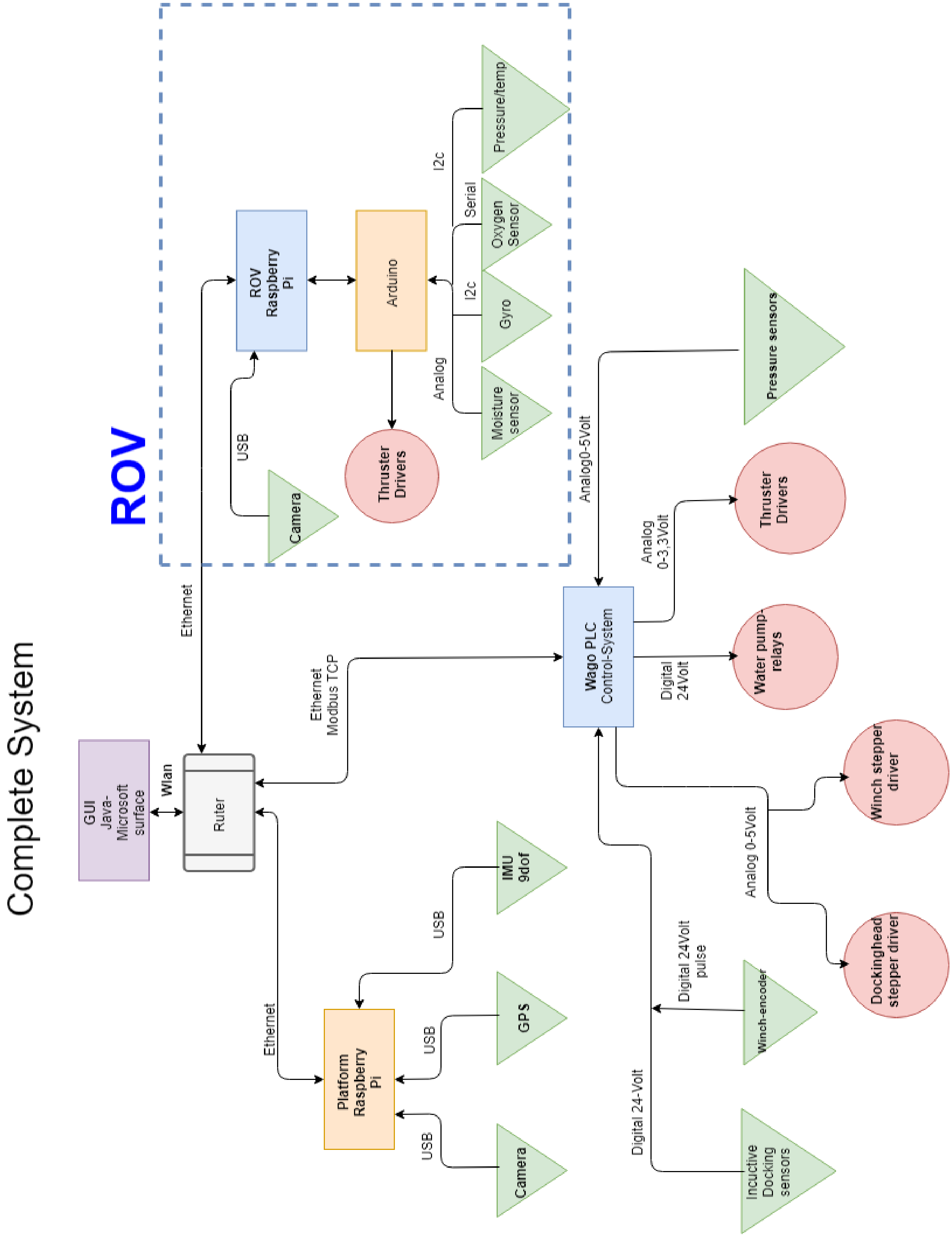


Figure 4.1: System layout

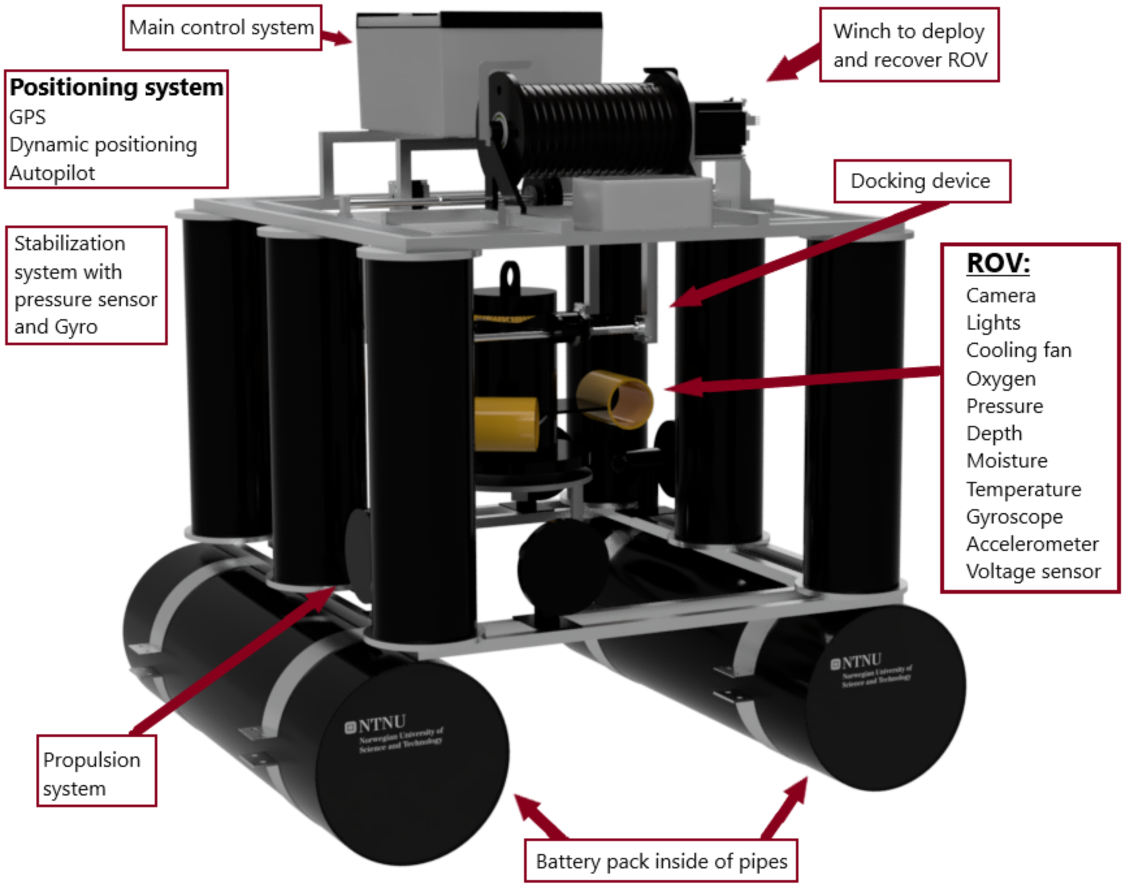


Figure 4.2: New USV design

4.2.1 Controllers

Wago PFC100 Controller

The Controller WAGO PFC100 is a programmable logic controller (PLC). This PLC will be used as the main control system. It will handle data and the I/O on the USV. The PLC supports different digital, analog, and I/O modules[21]. The modules can easily be expanded depending on what the system needs. In table 4.1, the different modules used in this project are listed. This PLC was selected because of its flexibility with its modular concept. NTNU Ålesund had the PLC in storage along with all the I/O modules that have been used in this project. All of the group members have previous experience with this PLC and the software used to program it.

Figure 4.3: Controller PFC100[21]

NR	Type	Description
1	750-602	24VDC Power supply to power the rest of the I/O modules
2	750-430	8-channel digital input
3	750-530	8-channel digital output
4	750-457	4-channel analog input; ±10 VDC with a resolution of 12 bits
5	750-559	4-channel analog output; 0...10 VDC with a resolution of 12 bits
6	750-670	Stepper controller used to control different drive power sections with pulse/direction [22]
7	750-637/000-002	Module interface for incremental encoders, 32 bits
8	750-600	End Module, it completes the internal data circuit and ensures correct data flow.

Table 4.1: Wago I/O modules used in this project

Raspberry Pi 3B+

The Raspberry Pi is an SBC with integrated WiFi, Ethernet, USB, Bluetooth, and I/O. It runs on a Linux based operating system and is powered by a 5VDC supply. It can control the integrated I/O by using several programming languages. The Raspberry pi is used for streaming of video feed and collecting sensor data. The Raspberry pi was chosen because of its low price. Both of the raspberry pi's that have been used in this project have been set up with a minimalistic operating system called "Raspbian Stretch Lite" which only include command window and does not include any desktop. This was done both to reduce RAM usage and storage usage.

4.2.2 Sensors

Ublox Waterproof USB 50 Channel GPS Receiver

This waterproof marine USB GPS was used in the previous Platform project, and there was no reason for changing it. This 50-channel GPS module supports standard NMEA-0183 messages and Supports DGPS [WAAS, EGNOS, and MSAS] signals [7].

Pressure sensors

The pressure sensors are used to measure how much water is inside each of the four vertical corner pipes. The sensors values for empty and full tanks were registered and mapped to display a value between 0 and 60 cm, which is the height of the pipe. The sensor value is used in the control system to stabilize the USV. There are also two pressure points on each side of the USV. These points are used to check the water level outside of the USV.

Inductive sensors

The inductive sensors are used to indicate when the ROV is in the correct position when it is docked or undocked from the docking head. There are three inductive sensors, one to indicate when the ROV is in the upper position and is ready to be docked, one to indicate that the locking mechanism is open and to indicate that it is closed. The sensors have a short range and only reacts when it reads metal.

4.2.3 Communication

A router with WiFi and external antenna was used to communicate between the different controllers. Measurements of the distance of the signal from this router were documented in the last platform report [13]. The router was placed inside the control-cabinet on the platform. All components on the platform used cat5 RJ45 cable to connect to the router, since a cable is quicker and more stable than WiFi.

The GUI was connected to the router through WiFi. Since this is a prototype, the HMI would always be in close range of the platform, so the range of the WiFi signal was not an important factor.

4.2.4 Control Cabinet

It was decided to mount most of the equipment inside of one cabinet. The cabinet is a plastic enclosure delivered by Rittal and has an enclosure of IP66. It is mounted cable glands that is IP68. The cabinet has the measurements: W*D*H, 400mm*300mm*200mm, originally a wider cabinet was ordered, but due delays in delivery the order had to be canceled. The figure 4.4 shows how the PLC, Raspberry Pi, SMC, Router, transformers, fuses, and microstep-drivers is fitted inside of the cabinet.

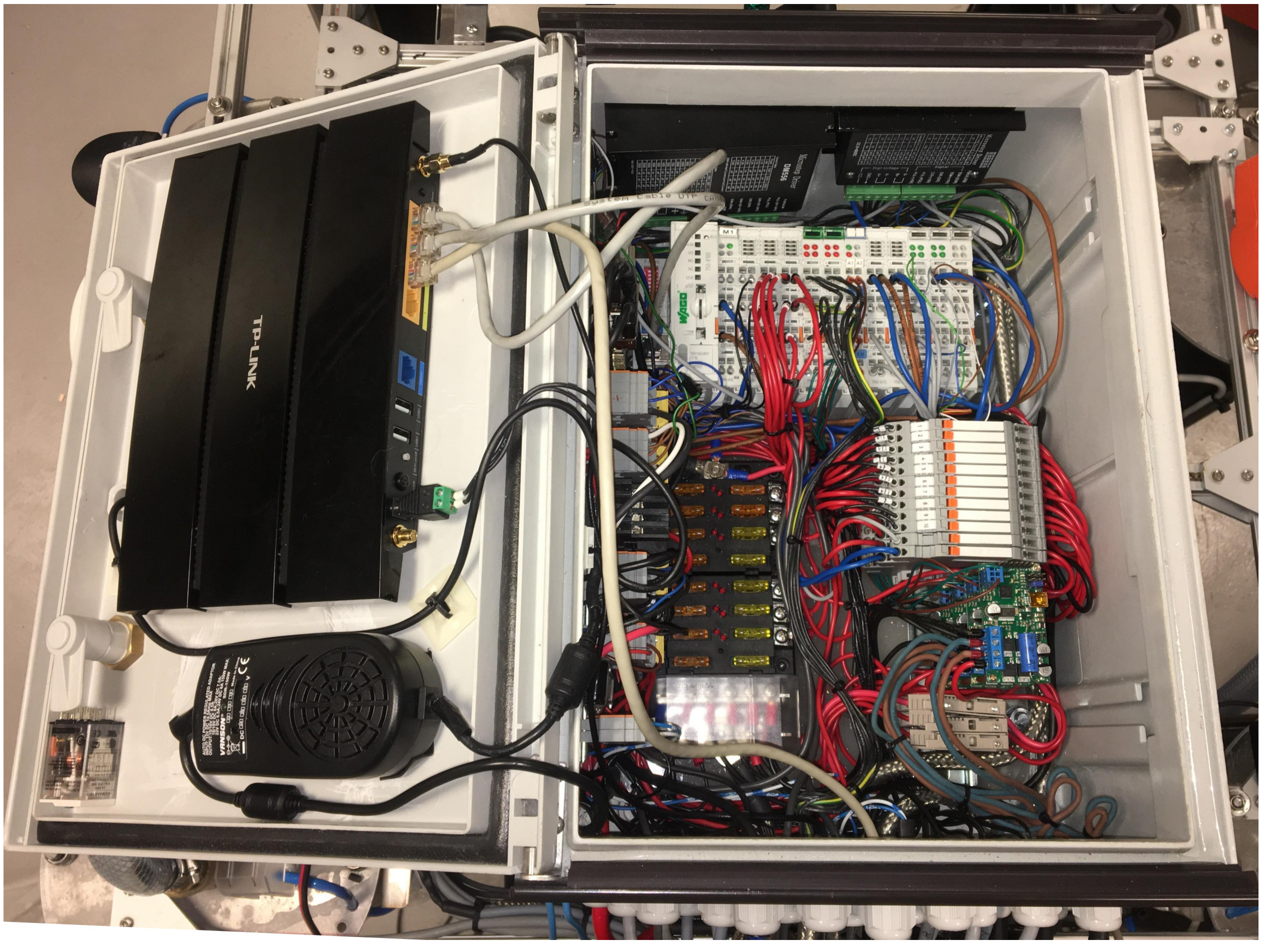


Figure 4.4: Cabinet Overview

4.2.5 Power distribution

The batteries delivers 12VDC. Onboard the platform there is need for three different voltages. The ROV, thrusters, pumps, router is using 12VDC. The Raspberry Pi and pressure sensors is using 5VDC and the PLC and microstepper drivers are using 24VDC. All equipment is protected with fuses. Figure 4.5 shows the power distribution diagram and table 4.2 is the different fuse size used. Both the 5VDC and 24VDC have their own transformers from 12VDC.

Fuse Size table	
Componet	**Fuse Size Amp**
Main fuse	70A
SMC	20A
Microstepper driver DM56	10A
Microstepper driver TB6600	10A
PLC	5A
Pumps	5A
ROV	10A
Power supply 24V	10A
Power supply 5V	10A

Table 4.2: Fuse Size Table

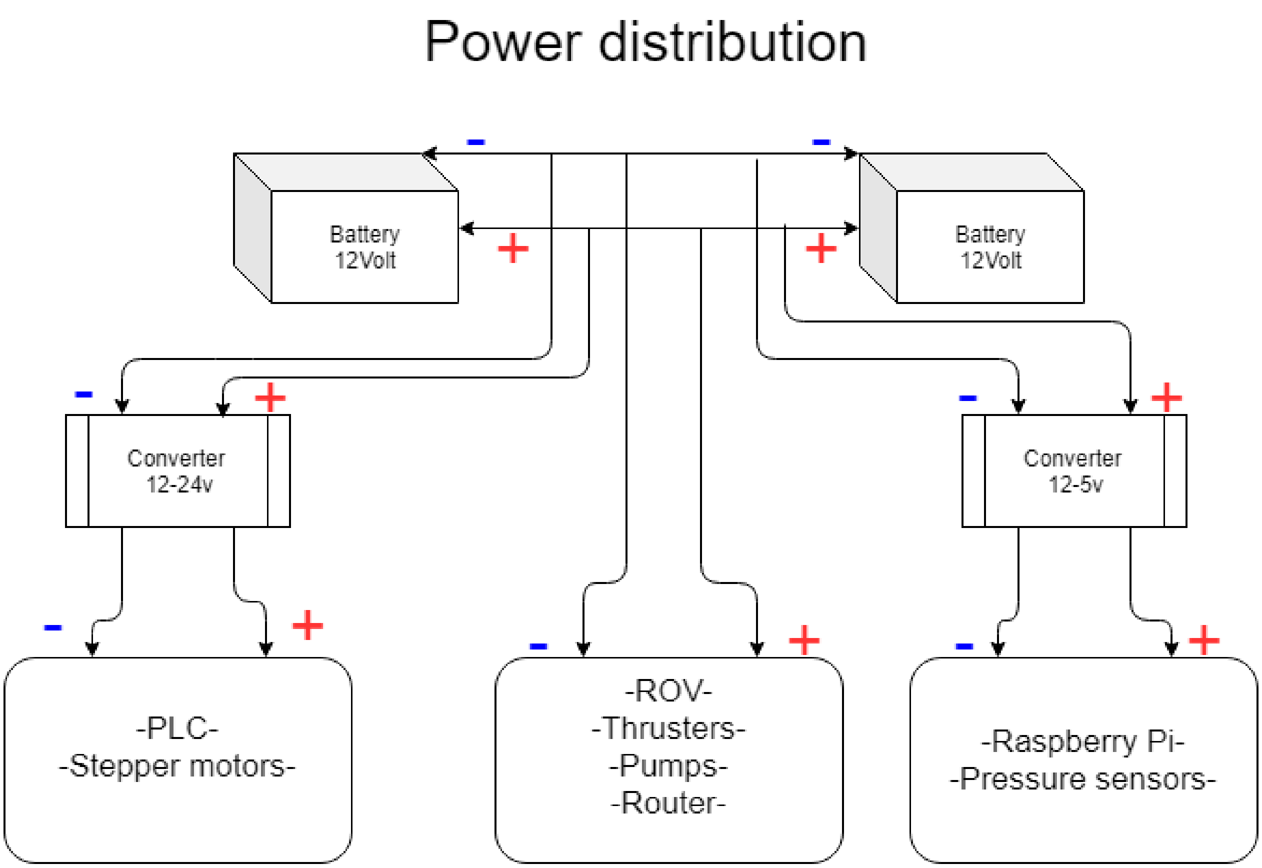


Figure 4.5: Power Distribution Diagram

Batteries

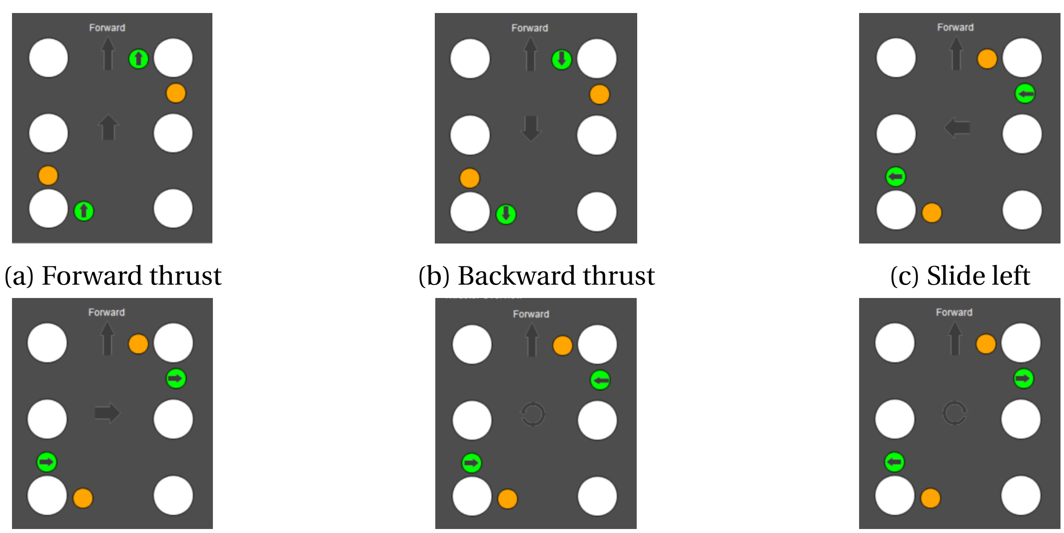
The same batteries were used again due to the price of new batteries. The water damaged batteries was replaced with new batteries that were identical to the old ones. In total, ten batteries are powering the USV. Each battery is 12Volt batteries with a capacity of 3.5Ah. When all the batteries are connected in parallel, the entire battery pack got a capacity of 35Ah 12 volt. With a current draw of 22 Ah in full thrust, the estimated run time is about 1,5 hours.

4.2.6 Controlling the Thrusters

Each thruster is powered by a Simple Motor Controller(SMC). The SMC is controlled with analog signals (0-3,3vdc) from the PLC. Depending on the analog signal, the output force of the thrusters can be regulated. To prevent unwanted events, the SMC is fail-safe. To make it fail-safe the power to the SMC goes through a relay. This relay will cut if the PLC loses power or if the PLC stops working for any reason.

4.2.7 Thruster placement

The four thrusters are placed in in two of the corners of the platform, were 2 of the thrusters are giving forward thrust, and backwards thrust. And the other two thrusters is used for rotating the platform, and sliding the platform sideways 4.6.



(a) Forward thrust (b) Backward thrust (c) Slide left

(d) Slide right (e) Rotate left (f) Rotate right

Figure 4.6: Movement of USV

4.2.8 Winch system

The new control system for the winch is with the use of a PLC. The PLC has a stepper controller module that sends signals (PWM) to a micro-step driver. The micro step driver is then powering the stepper motor. To measure how many meters of cable is out, it is mounted an incremental encoder on the winch. The encoder is connected to an incremental encoder interface extension module on the Wago PLC.A cover was 3D printed to protect the stepper motors from water. The cover was fitted with a IP68 cable gland for the cable to go trough. Figure 4.7 shows the winch.

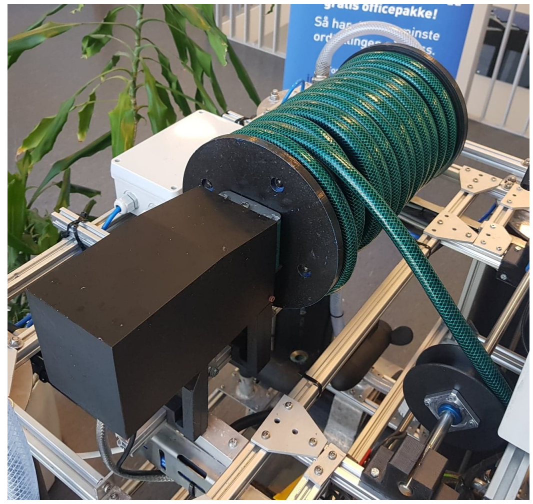


Figure 4.7: Winch Layout

Broken winch system parts

Several parts inside of the winch were damaged as seen in figure 2.8. The parts were made of regular PLA. In section 3.7.1 there is a suggestion to use PLA Though. Since the PLA Though is a stronger printing material than PLA, it was chosen to use when the new parts were 3D printed. After printing all new parts in PLA tough, the winch is working properly, and the parts seem to be strong enough.

On the docking device, the brackets that hold the two guiding rods were broken. The old brackets were 3D printed with PLA. Since the bracket is a critical part of the docking device and needs to be mechanically strong, it was decided to make new brackets in aluminum. By replacing the old plastic brackets with aluminum, the brackets are now strong and have a low weight. Figure 4.8 shows the new bracket in aluminum and figure 4.9 shows the old broken bracket made of PLA.

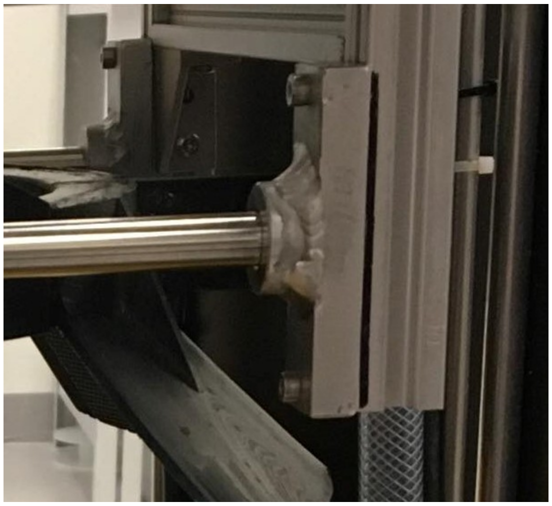


Figure 4.8: New Aluminum bracket

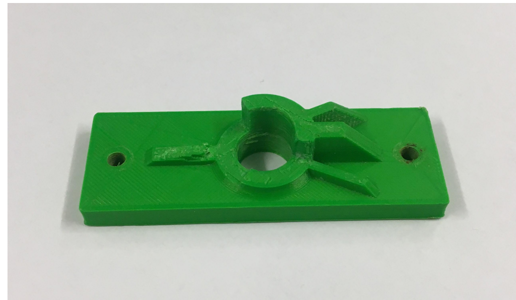


Figure 4.9: Old broken PLA bracket

Docking device

The docking device is controlled by the PLC by sending a PWM signal to the micro-step driver for the stepper motor. The stepper motor rotates a belt so the lead screw can open or close the lock. There are three inductive sensors mounted on the docking system to indicate the position of the lock and if the ROV is in upper position. Figure 4.10 shows the docking device and the different sensors location.

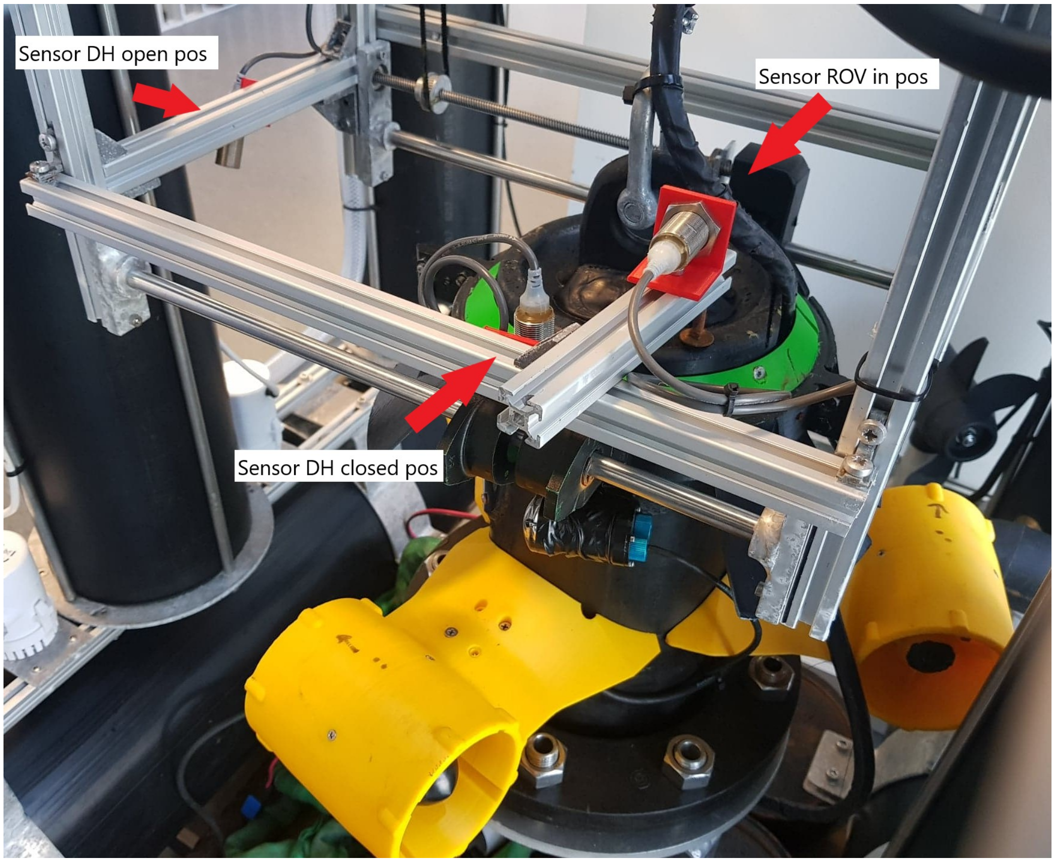


Figure 4.10: Sensors on Docking head

4.2.9 ROV

As explained in section 3.2.1 The ROV was working well during testing, but had large latency issues, and was not able to stream video at the same time as it was in operation. To solve this, it was decided to replace the software on the Raspberry. As mentioned in section 3.2.1 the old program was programmed in JAVA, but it was decided that the new program should be programmed in Python, this is due to library support on the raspberry since the raspberry is more commonly programmed in python. It is fully possible to use java on the raspberry, but it is more work to set it up than to use python.

Additional to the change in software, it was decided to replace the older Raspberry PI model 3, with a new Raspberry pi model B+ because it is more powerful.

4.3 Stabilisation system

The stabilization is done with pumping water in and out of the four corner columns. The PLC is controlling the pumps. By using the pitch and roll from the IMU the PLC will decide which pump to turn on and off. The pumps are located outside of each pipe to pump water in, and located inside the pipe to pump water out through a non-return valve. The pumps will pump water in/out depending on how much water there is inside of the columns. After the adjustment, the platform will be leveled. Figure 4.11 shows how the stabilization system is and table 4.3 described the different numbers listed in red in figure 4.11.

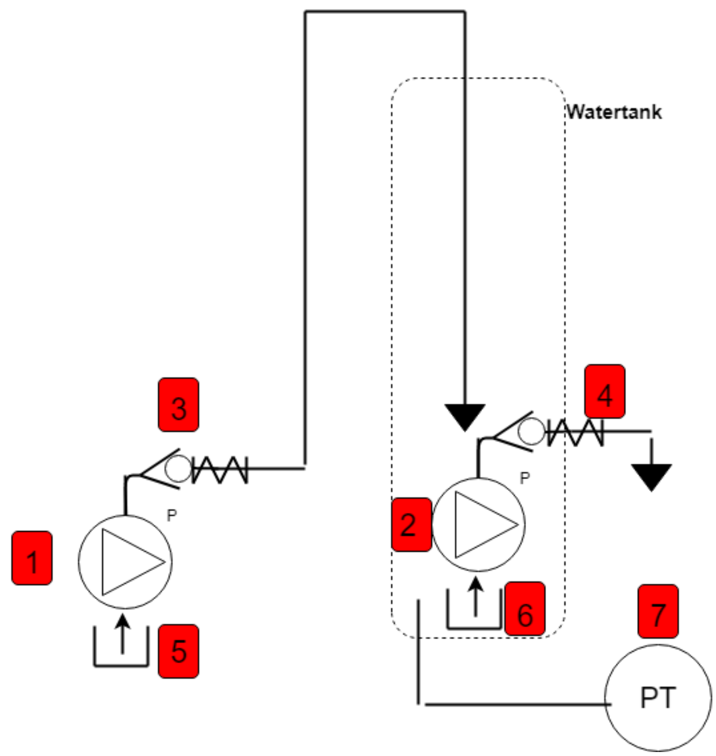


Figure 4.11: Stabilization system

NR	Type	Description
1	Filling pump	Biltema 12V/2.5A bilge pump 32L/min
2	Drain pump	Biltema 12V/2.5A bilge pump 32L/min
3	Filling-Check valve	Biltema check valve 1/2"
4	Drain-Check valve	Biltema check valve 1/2"
5	Seawater	Seawater from ocean
6	Seawater in pipe	Seawater from the inside of the pipes
7	Pressure sensor	MPX2010DP pressure sensor

Table 4.3: Stabilization system parts

4.4 Dynamic positioning

Dynamic positioning (DP) mode will keep the USV in the same position by using its thrusters. In figure 4.12, the DP-system is shown. When entering DP mode from the GUI, the GPS position to the USV is stored. This GPS position will the DP system maintain. First the USV maneuver the heading towards true north. After it will hold the position with adjusting movement in North/South and West/East direction. It is used three independent PID-controllers. One for the heading and two for the North/South and West/East movement. The distance from the setpoint is calculated with the haversine formula.

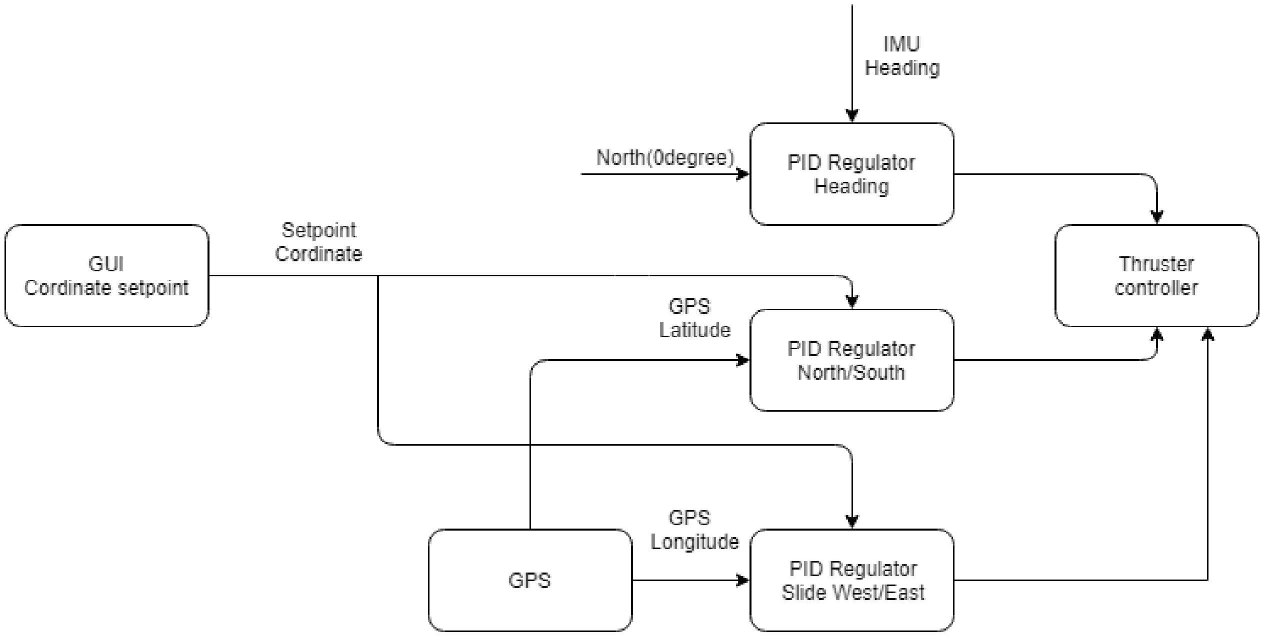


Figure 4.12: DP system

In section 4.9, the result from the sea trial of this system is shown.

4.5 Autopilot

The autopilot will steer the USV to the given destination that the user have plotted on the GUI. The autopilot system is shown in figure 4.13. The autopilot uses the GPS to know the position and the IMU to keep track on the heading. The PLC will calculate the distance with haversine formula, and the heading is calculated with an initial bearing formula. There are used two PID-controllers to maneuver the USV in autopilot. One for heading and one for forward speed. When correction heading, it will turn the shortest way towards setpoint heading. When a destination is reached and if a new destination is plotted, the USV will go to the next destination. If there is no new destination to go to the platform will go into DP-mode.

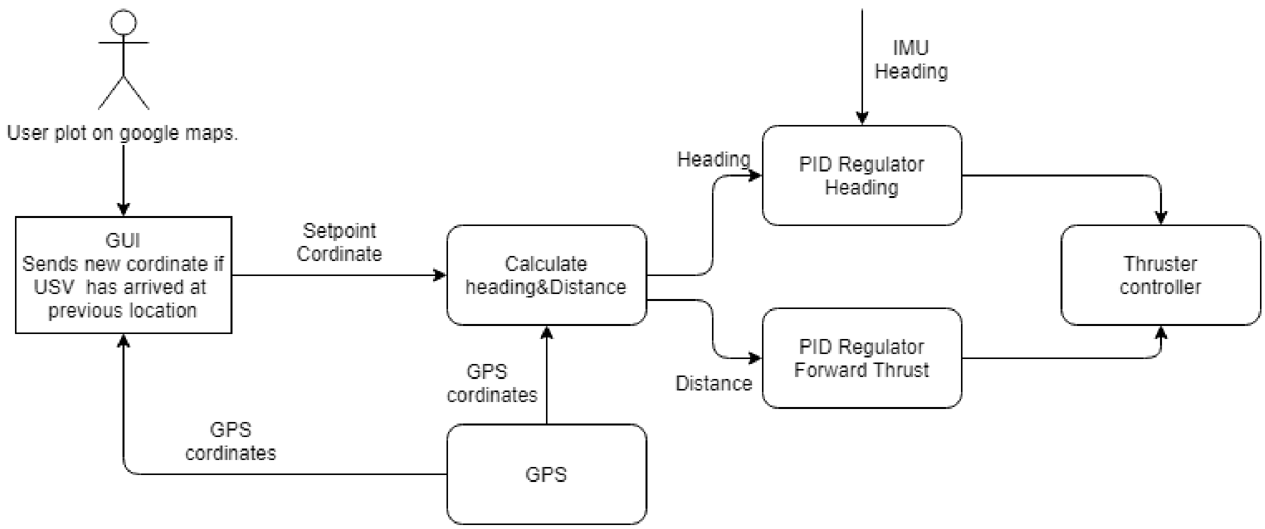


Figure 4.13: Autopilot system

In section 4.9, the result from the sea trial of this system is shown.

4.6 HMI

The Human Machine Interface(HMI) is the main method for the user to interact with the USV and ROV while in operation. Therefore it is important that the device is user-friendly and meets the necessary requirements mentioned in section 3.2.3. With this in mind, a combination of computer and tablet makes a good candidate for the HMI. A Microsoft surface is a computer in tablet form with a touch screen. It will provide powerful hardware in a small, user-friendly package where the user easily can interact with the GUI using the touch screen. It comes with a detachable keyboard that was useful during testing when changes to the GUI needed to be made. The Microsoft surface is shown in figure 4.14.

Figure 4.14: Microsoft Surface Go 10"

4.6.1 GUI

While programming the GUI, there was an overall goal that the design was user-friendly, modern, and informative.To get a modern look on the GUI and at the same time make it more pleasant to use over time, a dark colour was chosen as the background with a blue colour for the buttons with white text. The layout was separated into three tabs, so only the necessary information is shown to the user. By doing this, the user doesn't get over flooded by information while in operation.

Overview

The overview tab shown in figure 4.15 is focused on giving good feedback to the user about the system. This includes sensor data from the USV and ROV like speed, depth, and heading. It is in the overview tap the user can plot the route the USV will take during autopilot and where it should use DP mode. The map shows where the platform is located and where it is headed.

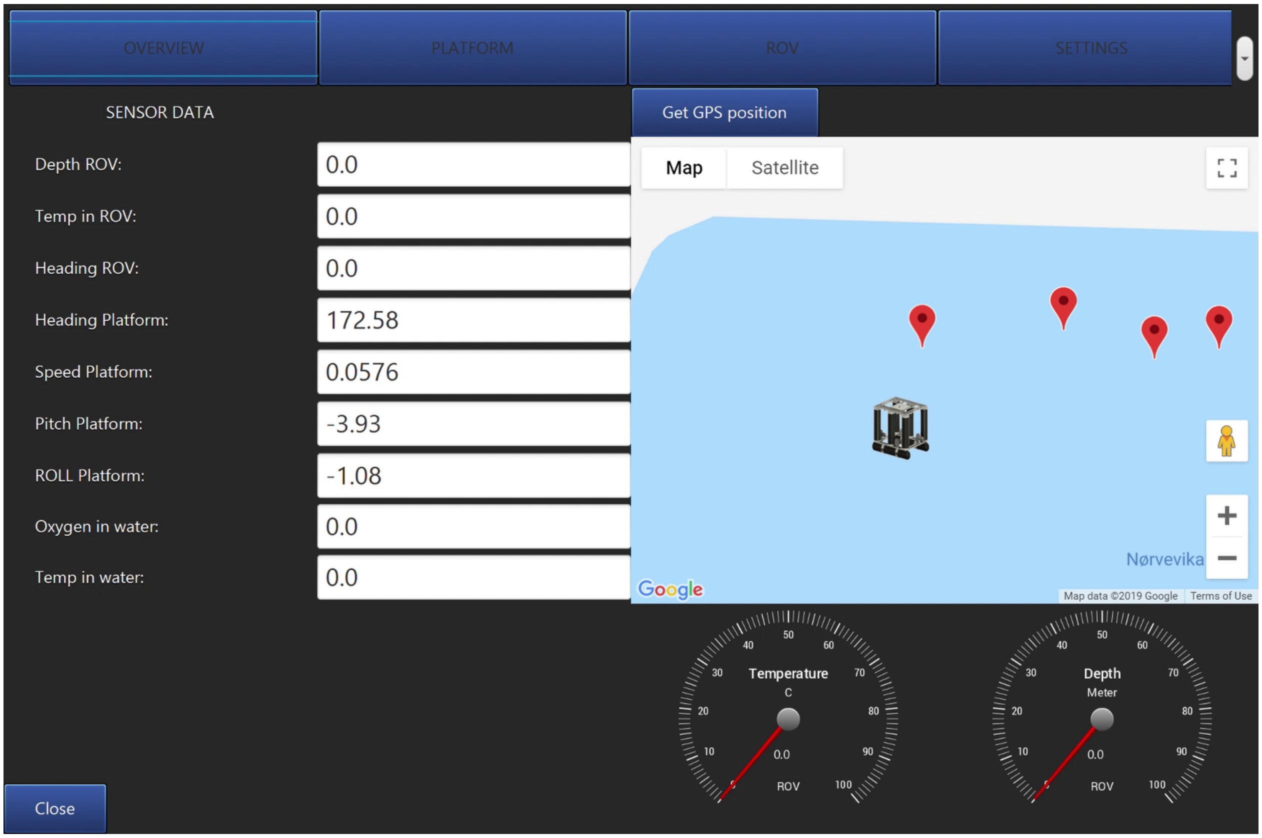


Figure 4.15: Overview tab GUI

Platform

The platform tab shown in figure 4.16 gives the user the possibility to control the USV manually by using the control buttons on the screen. It's in this tab the user decides which mode the USV will be in, manual, autopilot or DP mode. The tab also shows a video feed from the camera on the platform with the opportunity to record the video feed.

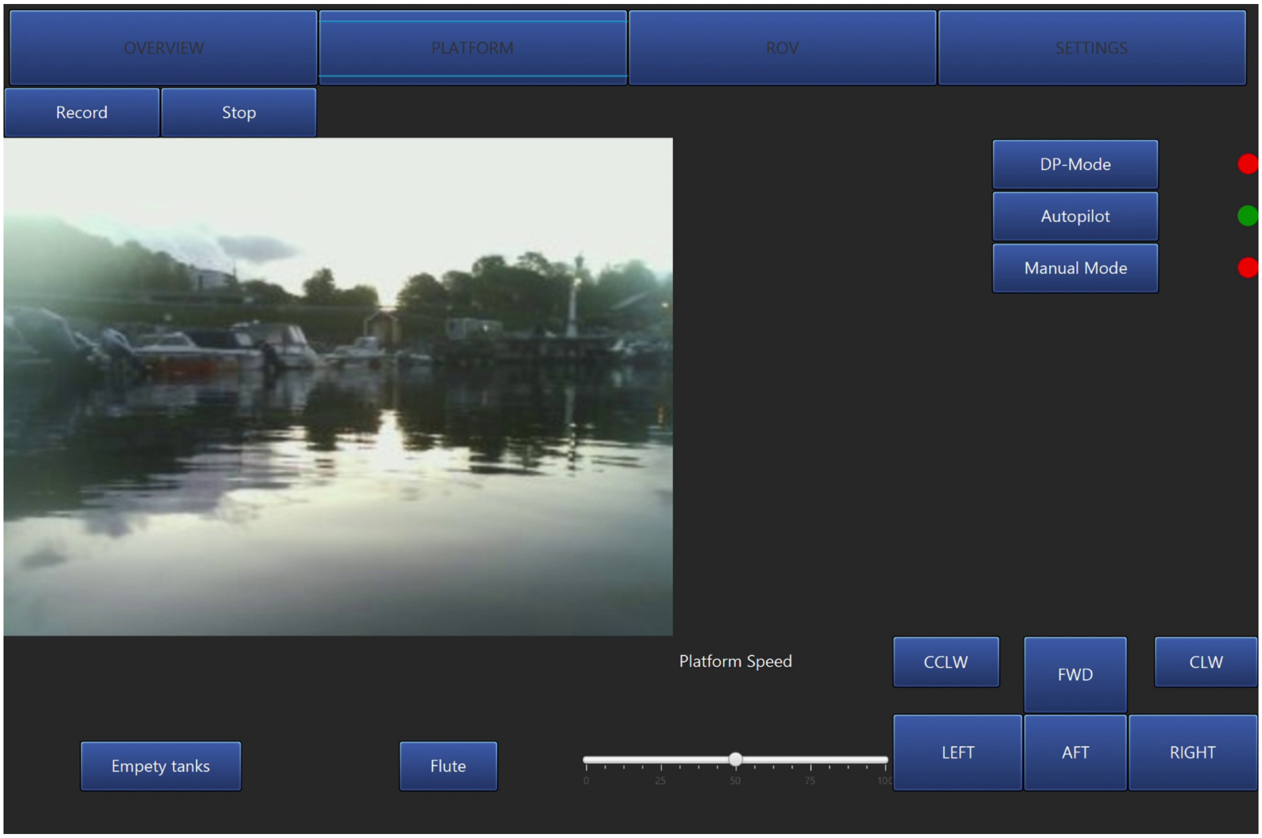


Figure 4.16: Platform tab GUI

ROV

In the ROV tab shown in figure 4.17, the user controls the winch and the ROV. The winch can deploy or retrieve the ROV by using the buttons, the speed of the winch is controlled by the slider. The omnidirectional ROV is controlled by the buttons on the right side of the screen, the speed of the ROV is controlled by the slider. The strength of the lights can be adjusted using the slider. There is also displayed information if the ROV is in its upper position and if it is locked in position. The video feed from the ROV is displayed on the left side, and the user can record the video feed.

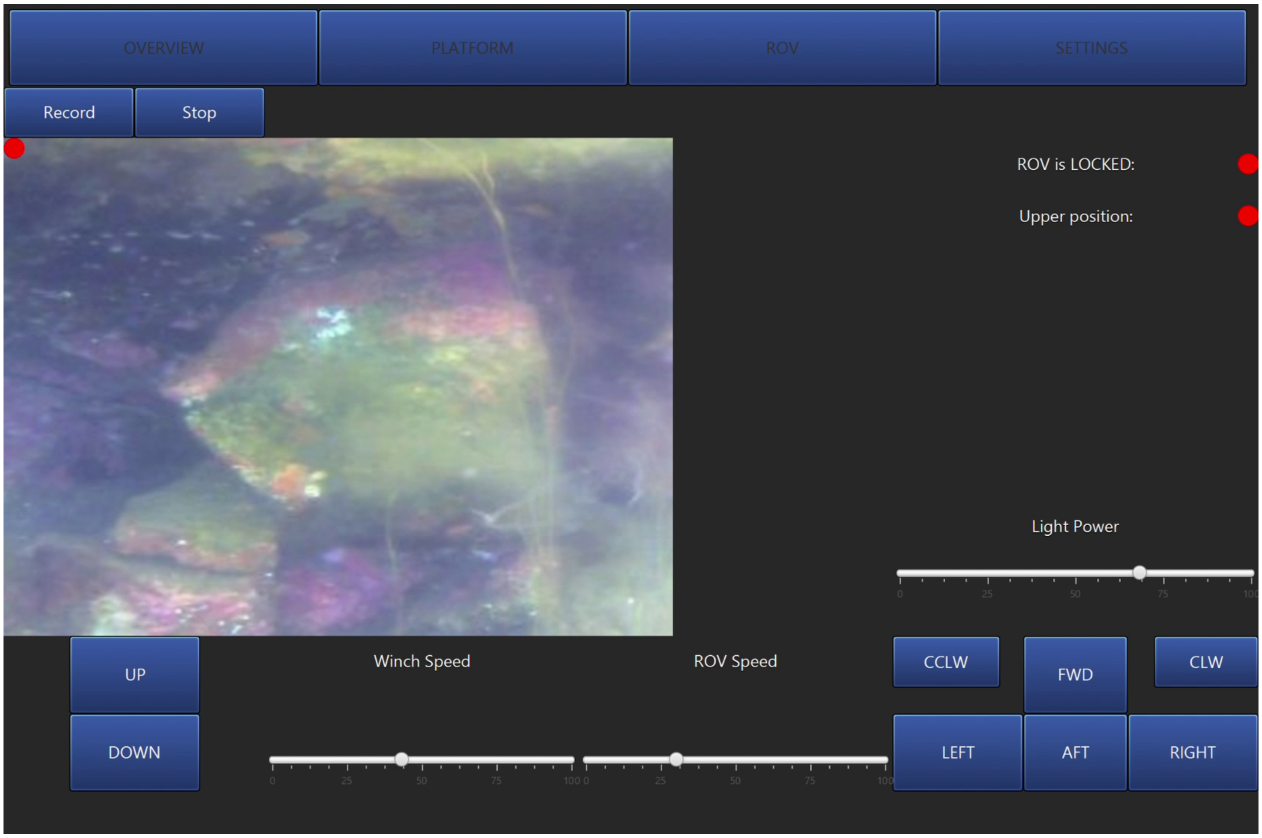


Figure 4.17: ROV tab GUI

4.7 Software development

The systems are separated into two parts, and the GUI is bringing them together. There is no data going directly between the ROV and USV controllers.

4.7.1 GUI-ROV Dataflow

The dataflow between the ROV and GUI is presented below. All communication between the ROV and the GUI is made by using UDP. The figure 4.18 is a flowchart of the GUI-ROV dataflow.

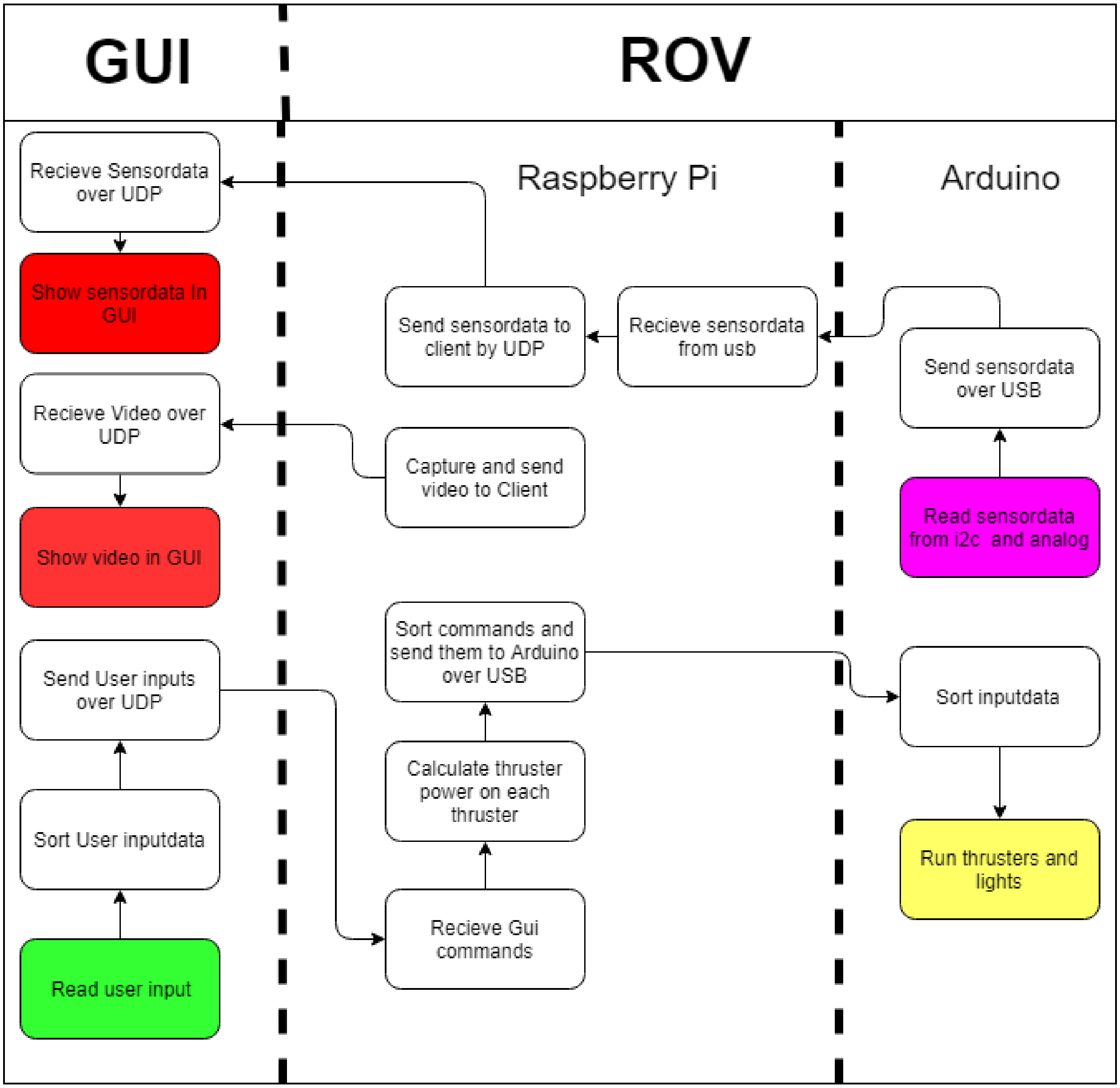


Figure 4.18: GUI-ROV Dataflow

4.7.2 GUI-USV Dataflow

The raspberry pi does not receive any data from the GUI or the USV. It only reads sensor data and sends it to the PLC, and sends the video stream to the GUI. All sensor data from the USV is pulled from the PLC using Modbus TCP. The PLC is the slave(Server) and the GUI, and the raspberry is the client(Master). The figure 4.19 is a flowchart of the GUI-USV dataflow.

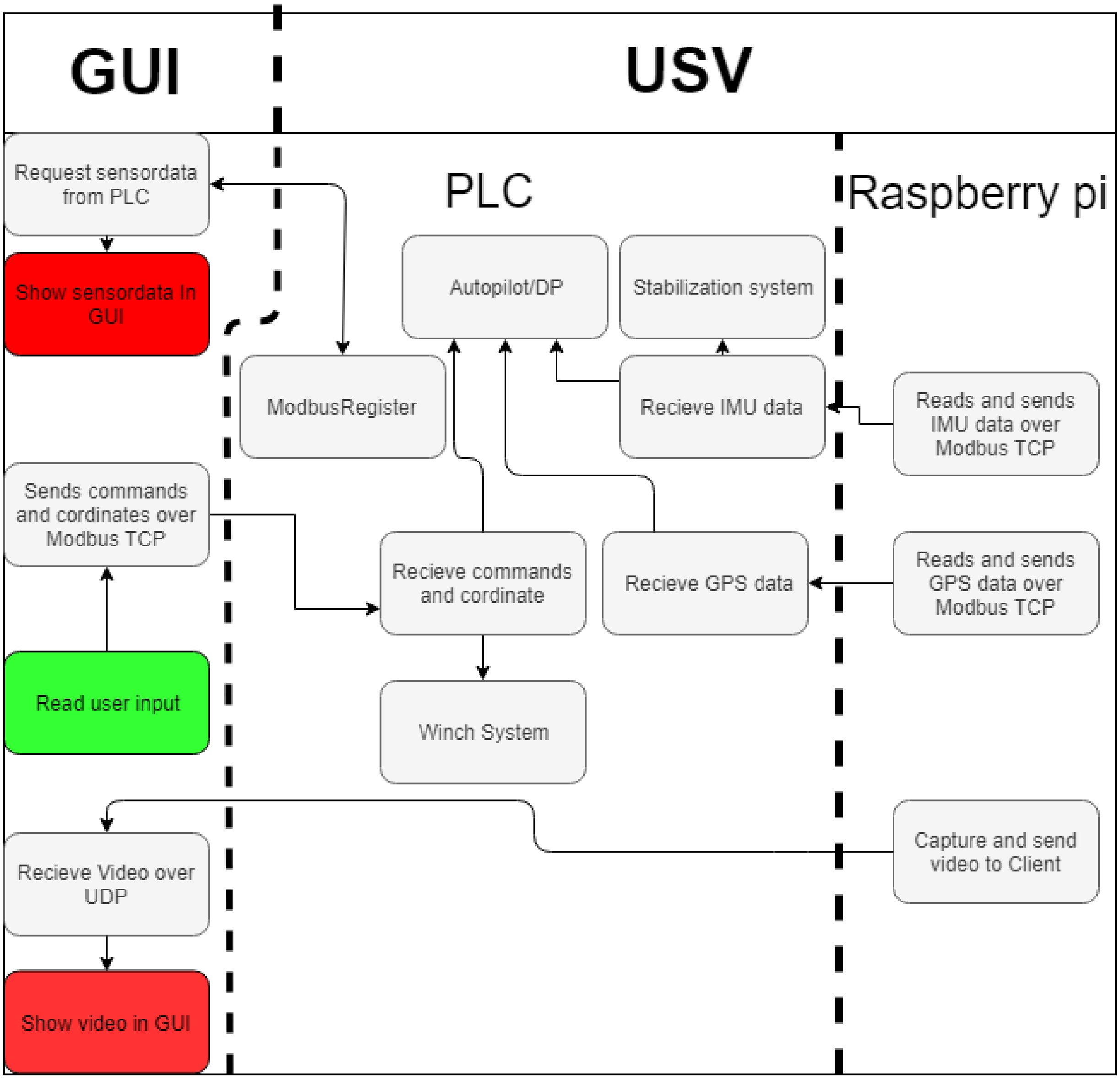


Figure 4.19: GUI-USV Dataflow

4.7.3 PLC software

The programming of the PLC is done with the associated software tool e!cockpit. With this tool, Modbus communication and all extension modules are easy to implement. The program structure is with the use of Program Organization Unit (POU).

Main program

The main program is built with the use of Sequential function chart (SFC). This have three different branches. One to Manual mode, one to Autopilot and one to DP mode. Figure 4.20 shows the SFC of the main program.

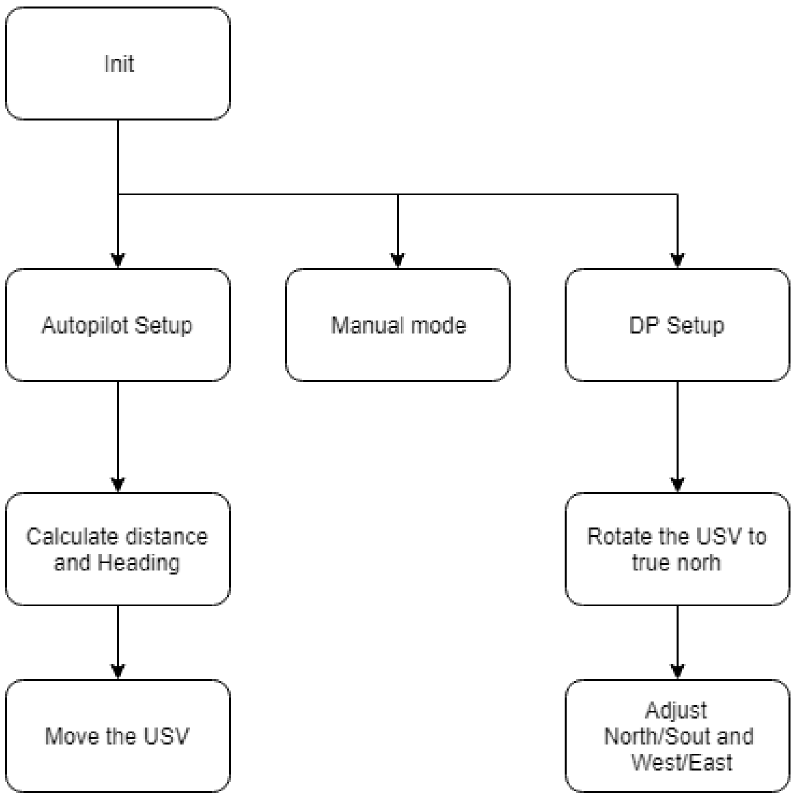


Figure 4.20: Sequential function chart of main program

Stabilization system and winch system

Stabilization uses its own task. This is done so the platform will continuously keep adjusting if necessary. Winch system has its own task to keep it possible to drive the winch and docking head.

4.7.4 USV Raspberry pi

The code on the raspberry pi is programmed in python.

Modbus Writer Class

The Modbus writer class is the class handling the sensor data and sends it to the PLC when the sensor values has been updated either from the IMU or from the GPS. Modbus writer class initiates two threads, the GPS reader thread, and the gyro reader thread. To handle data between the threads, the queue function is used. By calling "isQueueReady" in the two threads, the modbuswriter can know if the data has been updated, and it is then safe to pull data from the queue objects using the "getQueueData" the data inside the queue is an array of data.

The data is then sent to the correct Modbus register. When sending to a register, there are four things that one have to know.

- IP address of the PLC

  The PLC is set up with a static IP meaning that the IP address of the PLC is not given by the router but is set by the PLC.

- Modbus registerID and lenght

  Every register got an ID and length. The length of the register varies depending on the datatype that is sent.

- Datatype of the register

  The datatype is the datatype that the PLC expects receiving from the client.

Converting data types that is used in python to datatypes that are supported to the IEC 61131-3 PLC standard is done by using the builders that are included in the pymodbus library, as seen in figure 4.21.

```
def build64BitMessage(self, message):
    # encode 64 bit float message
    self.builder.reset()
    self.builder.add_64bit_float(message)
    encoded64 = self.builder.to_registers()
    return encoded64
```

Figure 4.21: Building Modbus data

GPS Reader class

There were options for reading GPS sensor data. To get a more flexible system, it was decided to use an open-source project called "Gpsd" instead of just reading the NMEA sentence from the serial port in python and decoding the NMEA-message to get the GPS data.

Gpsd is a daemon running in the background of the raspberry pi which is reading the messages from the GPS and makes the data available to be queried on TCP port 2947 of the host computer [5]. One problem with reading straight from the NMEA messages is that the NMEA message standard is a poorly specified standard, for instance, some GPS receivers leave the data field empty if there is no information, and other will send 0.00. Writing a program to read from one specified GPS receiver is not challenging, but the same program may not work using another receiver from another brand, since some GPS receivers even got their own specified standard. The Gpsd project has collected data from most GPS chips and will parse the NMEA message into one standard message, which is easy to decode. Gpsd is used in the android map functions in android phones, and also widely used in drones,driverless cars, and marine applications[5].

4.7.5 Raspberry pi ROV

The new program used parts of the old program that was working, but the syntax was changed to python, for example, the class for calculating power on each thruster when given an orientation and power was "copied" and changed to fit the Python programming language. The threading was changed by using a more thread-safe approach witch resulting in a more stable software that was working with no crashes during the testing period. The old video stream was completely removed, and replaced with a new code.

DataHandler class

The data handler class creates four threads, the serial reader, serial sender, UDP reader, and the UDP sender. The datahandler class creates the threads and uses the queue to pull data from each thread. To avoid data collision on the serial port. The Datahandler sends a semaphore to both the serial reader and the serial sender. This assures that the reader and sender will not open the port at the same time. When one of the threads wants to use the port, it uses the "semaphore.acquire()" function, when the job is done, the thread releases the semaphore, making it available to the other thread that is trying to acquire the semaphore.

When the Datahandler is running, it's going in a while loop. First, it checks if the UDP receiver got a new message, if there is a new message it gets the message, and calculate the thruster value for each thruster using the Thrustercontrol class, then the message is built into a byte array of seven bytes. The byte array is then sent to the queue of the serial sender class. Next, it is checked of there is new data in the queue of the serialreciever. If there is new data, the data is sent to the UDP sender class.

4.8 Data collection and calculation

4.8.1 Measurement devices

To measure the current draw of the system, a clamp meter of the type "Gmc1-prosys CP30" was clamped around the cable that was being measured. The clamp meter got a calibration certificate and got a measurement uncertainty of 29mA at 20A.The clamp meter generates a voltage of 100mV/A. The current of the system is then measured by connecting the clampmeter to an analog input on the PLC (750-457 module), with a range of -10-10Volt and a resolution of 12 bit. The rest of the measured data is from the systems that are integrated into the platform.

4.8.2 Logging

The logging is done by the PLC. The wago software got an integrated "trace" function. In the trace function, it is possible to set sample rates in seconds, and logging duration. The data is plotted live in e!cockpit, which is useful when testing and when calibrating the systems. When the data is captured, it can be saved as an excel format. This saved data can be copied into other software like Matlab for analysis.

4.9 Sea Trial

Testing the system at sea was done in Nørvevika Ålesund, the platform was transported in an enclosed trailer. And lifted to sea by a crane that was made available to us by Ålesund sail club.

The USV was tested over a period of six days. During testing, problems occurred that needed to be addressed. When the problems were fixed, a system test was done, and the results were documented. Due to battery restraints, and sometimes the crane was occupied, the testing was not always as effective. Table 4.4 is the time span of the test period. Next subsections will show the progress of the different days.

Date	Location	Time	Activity
27.04	Nørvevika	15.00-22.00	First time on water
28.04	NTNU	08.00-22.00	Finding reason of instability
29.04	Nørvevika	08.00-22.00	Testing theory
30.04	Nørvevika	08.00-22.00	Stabilisation and manual mode
11.05	Nørvevika	08.00-22.00	DP, ROV and Autopilot
12.05	Nørvevika	08.00-22.00	Final system test

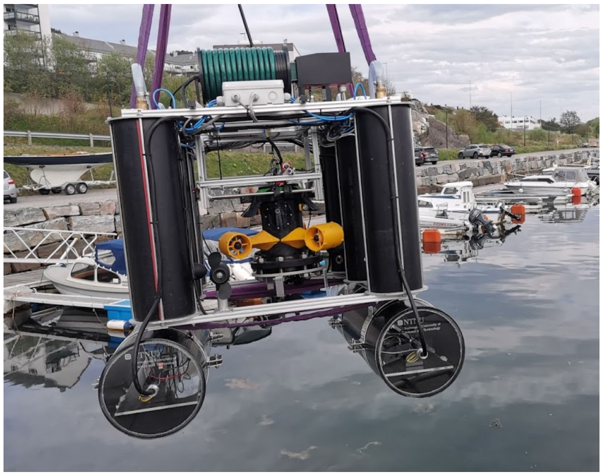
Table 4.4: Test dates

4.9.1 Sea Trial Day One - First time on water

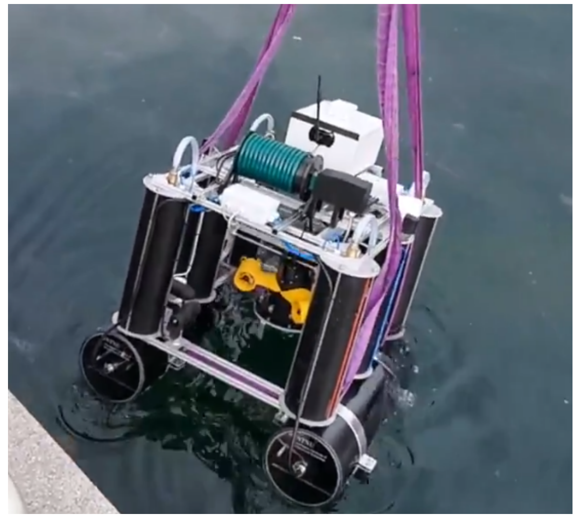
The platform was carefully placed in the water by the crane, as seen in figure 4.22. The goals for the testing that day was to check if everything was watertight, since leakage had been a problem before, and check how stable the platform was in the water. The weight of the platform was measured to 164 KG, which is 6 kg off our initial estimate off 170KG.

- Check if the pipes are watertight.

  The platform was placed in the water for 10 minutes still held by the crane. When the time was up, the platform was hoisted to land again, and the pipes for the batteries was inspected for leakage. Since it was mounted Plexiglas on the end of the pipes, it was easy to do the inspection without opening the pipes. There was no sign of leakage, and both of the battery pipes were dry. For security measure, the pipes for the batteries was inspected



(a) Platform in the crane



(b) Platform in the water

Figure 4.22: First time in the water

each time the platform was hoisted on land, and there was no sign of leakage over the whole testing period.

- Check the stability and the buoyancy of the platform in the water.

  The platform was put back into the water, and it was clear that the platform was unstable, and had too much buoyancy. Without extra mounted weight on the platform, the water level was measured to 40 cm

  As mentioned in section 3.5.2 bigger pipes were mounted that was able to hold bigger batteries that weigh more. Since new batteries were not mounted the platform weighed 45 Kg less than the weight that the pipes were dimensioned for.

- Check if pumps for stabilization system are working.

  The pumps for the stabilization system was activated using the Wago GUI that was made for commissioning, and simulation. The pumps started running, but there was little water coming through the hoses. The pumps had been sitting with saltwater for over a year, and it was a high possibility that the pumps were damaged due to this. But after further investigation, it was found that the non-return valves on each pump were stuck due to the saltwater that had been sitting there over time. After cleaning the valves, the pumps worked fine.

4.9.2 Sea Trial Day Two - Finding the reason for instability

After issues with the stability and buoyancy of the USV were discovered on the first sea trial, the group agreed that it was necessary to do some more calculations, and testing before continuing the sea trial.

Since the platform was not in balance, the platform was placed on four weight, in order to check the weight distribution. Since all of the weights that were used were different, and the ground that the weights were standing on was not completely flat, the accuracy was not optimal in this test, but by moving the weights around, it was possible to find a trend of the weight distribution 4.23.The winch motor is one of the heaviest components onboard the platform, by moving the winch 10 cm backwards, the centre of gravity got closer to the centre of the platform.

Figure 4.23: Stabilization Graph of Roll and Pitch

Finding the reason behind low stability

A theory was made that the pipes on the bottom of the vessel were too large compared to the horizontal pipes. This made the vessel unstable because the center of buoyancy would change by little when the USV was tilted in the water, due to the little volume the horizontal pipes would add in the water compare to the volume of the larger pipes that were already submerged.

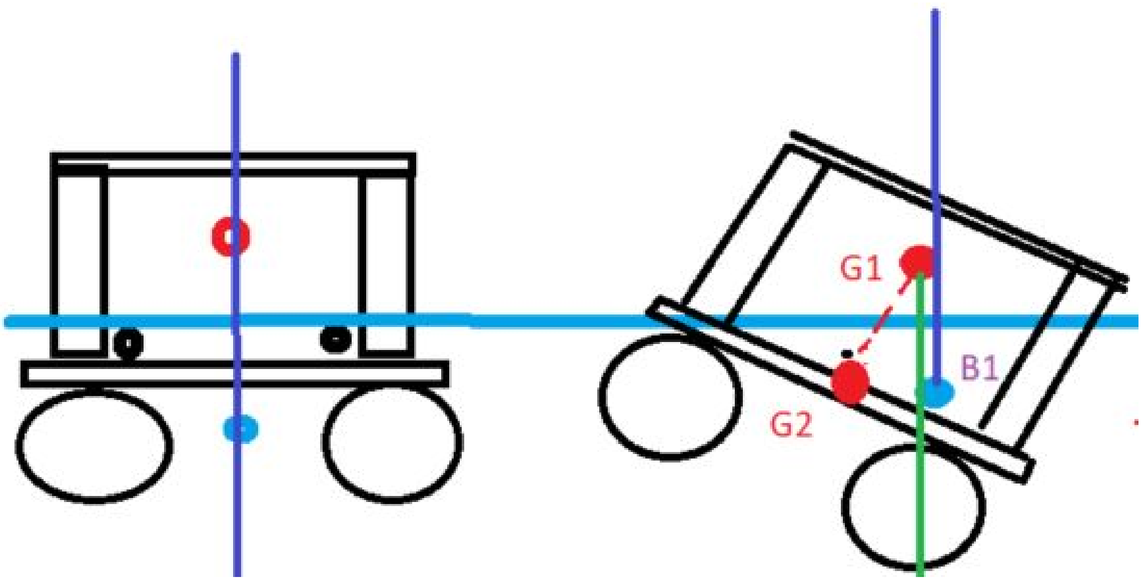


Figure 4.24: By lowering the center of gravity the stability will increase

To fix this, there were mainly three alternatives. Either to increase the size of the horizontal pipes, reduce the size of the vertical pipes, or try to lower the center of gravity enough so the buoyancy force would have a longer arm on the center of gravity, as seen in figure 4.25. Another alternative would be to increase the length and width of the platform. Since there was limited time, and it was important to check the electronic systems that were built, it was decided to add weight on the bottom of the pipes, hence lowering the center of gravity.

4.9.3 Sea Trial Day Three - Testing theory

The group acquired 100 kg of rebars from a local company. The rebars were cut into one-meter lengths so they could be fastened to the bottom of the pipes as seen in figure 4.25.

Figure 4.25: Lowering center of gravity

The stability increased gradually along with the weight added. After adding in total 80 kg, the platform was laying in the water without toppling itself, but still was not stable, the waterline was at 40 cm up on the vertical pipes, leaving 20 cm to the top of the platform.

Since adding even more weight was not a sustainable solution, it was decided to make a floating ring around the platform which would give the platform the missing buoyant force to prevent it from rolling over during testing. This ring was placed as seen in figure 4.26. The pipes are made of PVC and are 110mm in diameter. By adding this ring, the platform was now stable, and the system test could continue.

Figure 4.26: Platform Floating Ring

4.9.4 Sea Trial Day Four - Stabilization and manual mode

Sea trial day four, the system test began. First was the stability system tested, then bollard pull, after that a speed test and finally a turning test.

Stabilization system

The floating ring was moved to the top of the platform, so if the platform had zero degrees in roll and pitch, the ring would not touch the water surface. Since the platform was unstable, it was challenging to make a stabilization system that was fast enough, and at the same time, precise enough to not overfill the tanks, resulting in the platform tilting in the opposite direction since the balancing point of the platform is small. Since the pumps are not proportional, only on/off control, this would prove challenging.

An attempt to reduce the flow by the pumps was done by running the pumps in pulses, this was working, but still the platform was to unstable to be stabilized.

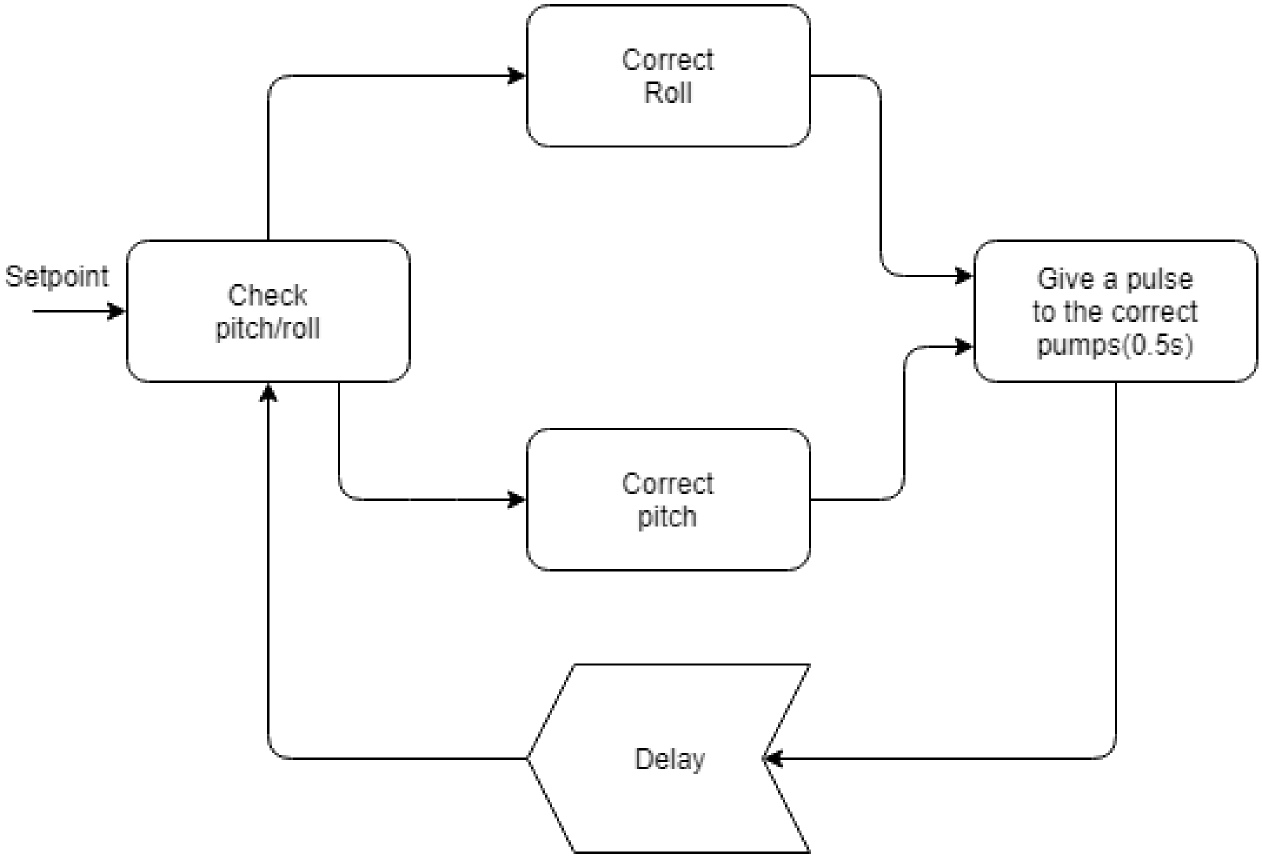


Figure 4.27: Stabilization of roll and pitch data

The stabilization system works, but due to the platform stabilization issue, the platform won't stabilize. Figure 4.28 shows the roll and pitch when the stabilization system is active. Figure 4.27 shows the flow chart of the stabilization system.

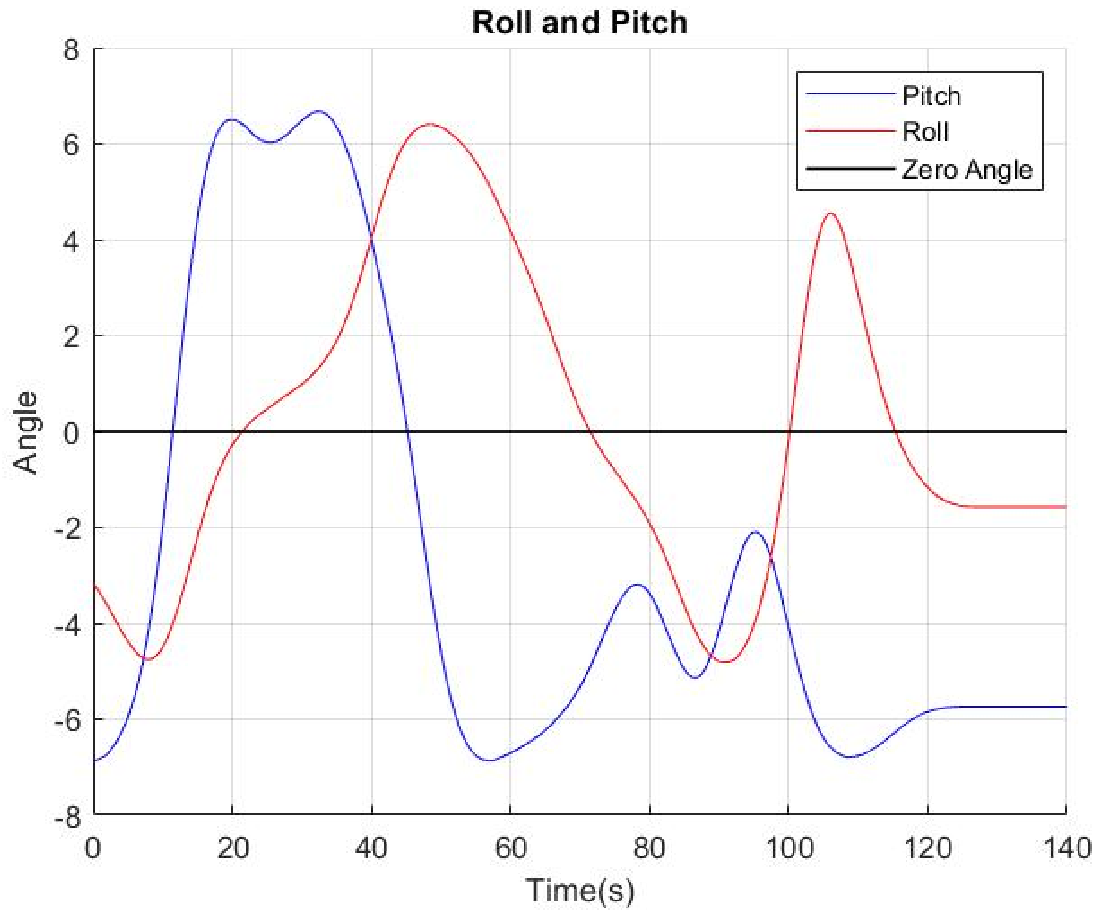


Figure 4.28: Stabilization Graph of Roll and Pitch

Bollard pull

A bollard pull test is a test which is used on to determine the pulling force of a vessel. This test was done to check the maximum pulling capacity the USV can exert. During the test, there were no waves. A load cell was connected between the USV and a fixed floating dock. The USV was controlled manually, and the speed of the thrusters was sat to maximum forward speed. The result was 0.9 kg pull force. Calculated to newton meter gives: $0.9 * 9.8 = 8.82N$.

Speed test

A test for determining the most effective power output of the thrusters compared to current draw and speed was done. In a speed test the thrusters on the USV is set to maximum power. The speed is continuously recorded with the use of GPS. Table 4.5 shows the average result from this test. Reducing power will lower the current consumption and it will not go much down in speed.

Speed Test		
Thruster Power	**Speed km/h**	**Current**
100%	0.9	22A
50%	0.6	7A
25%	0.4	5A

Table 4.5: Result from the speed test. Thruster power, speed and current consumption

Pivot Turning time

It was measured the time it took to turn the platform with pivot turning. Figure 4.29 the time the USV use is 8 seconds for a full 360 degrees turn.

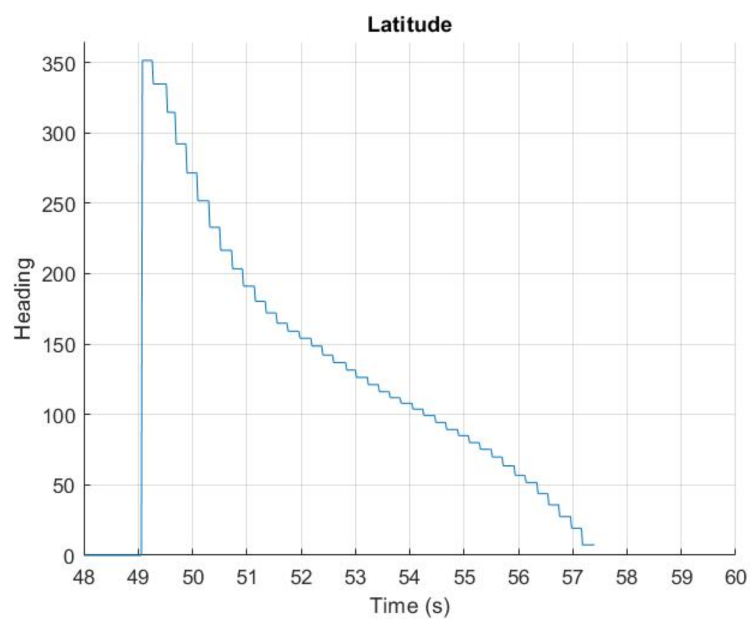


Figure 4.29: Pivot turning test

4.9.5 Sea Trial Day Five - DP and autopilot

Sea trial day five the autopilot and DP was tested. There was a fault in both the autopilot code and DP code.

Autopilot

The autopilot did not work as expected. The USV found the right heading but drove the wrong way. This was because the IMU was not calibrated right. After calibration, the heading was right.

DP

The DP test was not successful. There was a fault in the code, so the setpoint was updated every 2 seconds. Figure 4.30 and 4.31 shows clearly the fault. The DP system will update the same position as the USV as the USV is drafting away from the initial setpoint. This resulted in a nonfunctional DP. The fault was later that day fixed, and the result will be shown in section 4.9.6.

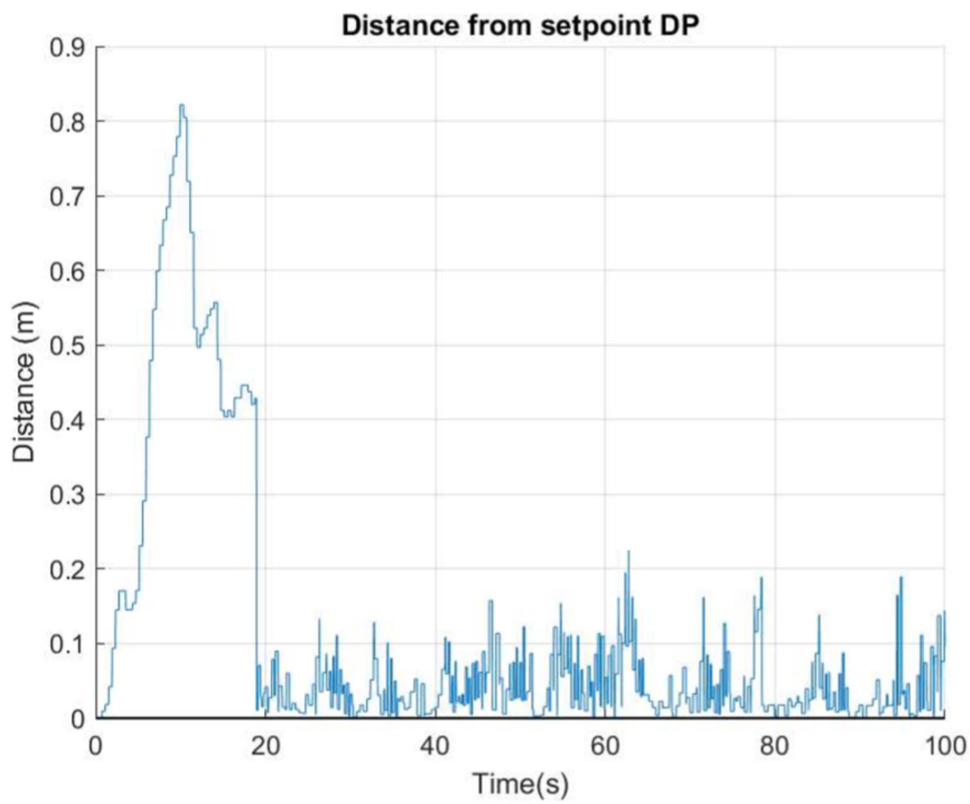


Figure 4.30: In this graph the fault in the DP system is shown

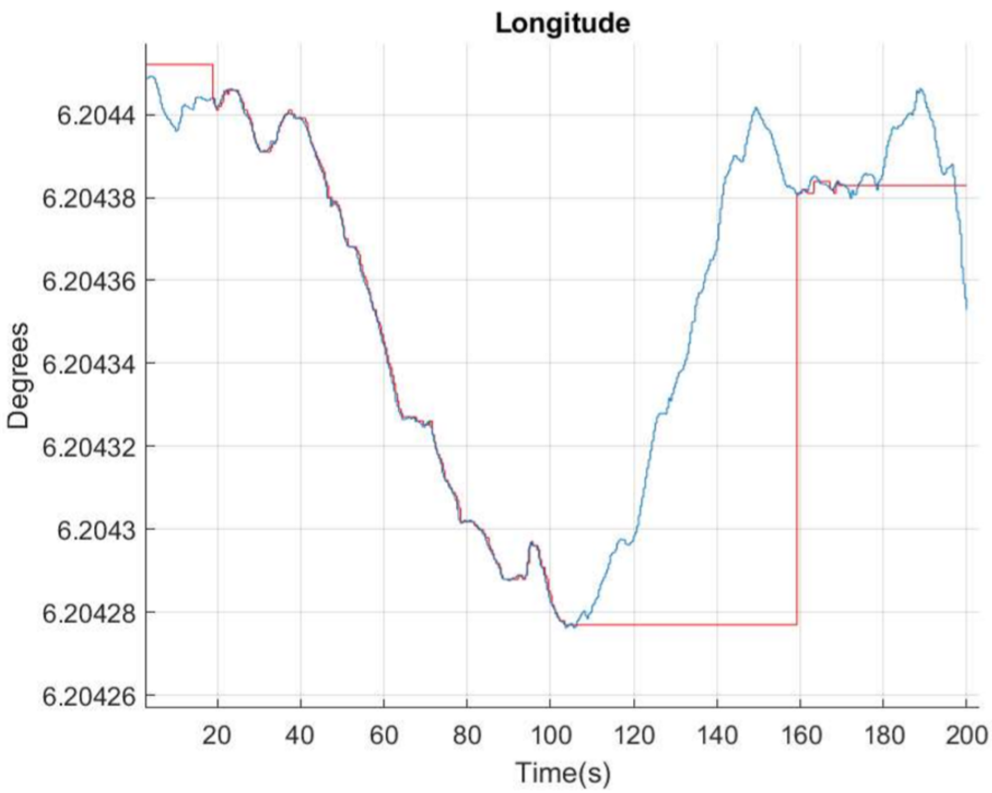


Figure 4.31: In this graph the fault in the DP system is shown

4.9.6 Sea Trial Day Six - Final system test

Sea trial day six, the system was further optimized and tested more. After the data from day, five was analyzed at the university. A bug was found in the client software that changed the setpoint of the DP system to the same coordinates as the placement of the USV, this resulted in the USV drifting since there was no difference between the setpoint and current position. The bug in the client code was corrected, and the system was now ready. The result will now be presented, first autopilot then DP.

Autopilot - Result

The autopilot was tested with plotting different destinations on the control device GUI. Then the USV was sat in autopilot, and it began to drive to the destination. When the USV reached the destination, it turned 180 degrees and continued to the next destination.The autopilot worked well. Figure 4.32 the GUI with plotted destinations is shown. The figure 4.33 the recorded motion of the USV is plotted to show where it went. This motion is data logged from the GPS. In figure 4.35 the distance from the destination is shown. At 22 seconds, it receives a new destination. When the USV is closer than 10 meters, the client will acknowledge the position and send a new coordinate. Figure 4.34 shows how the USV was holding the heading towards the destination.

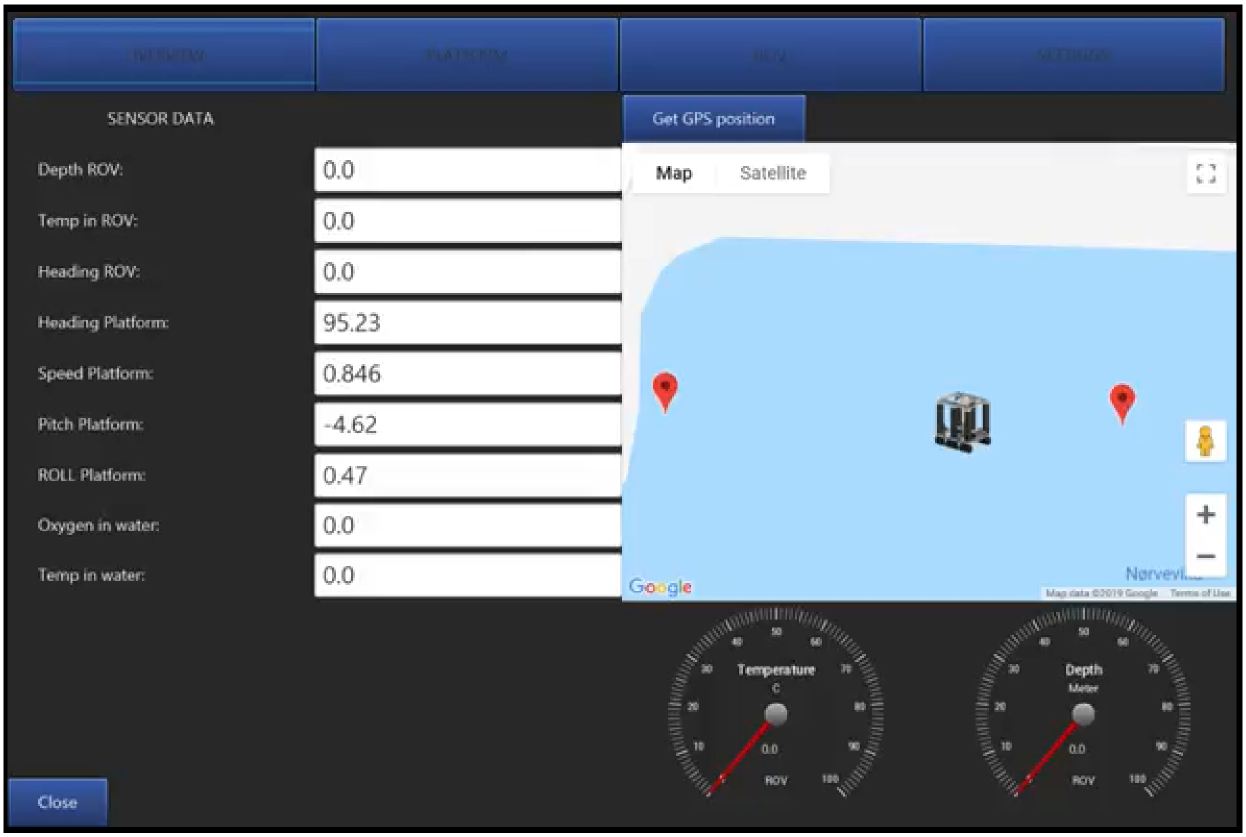


Figure 4.32: Autopilot GUI plot travel route

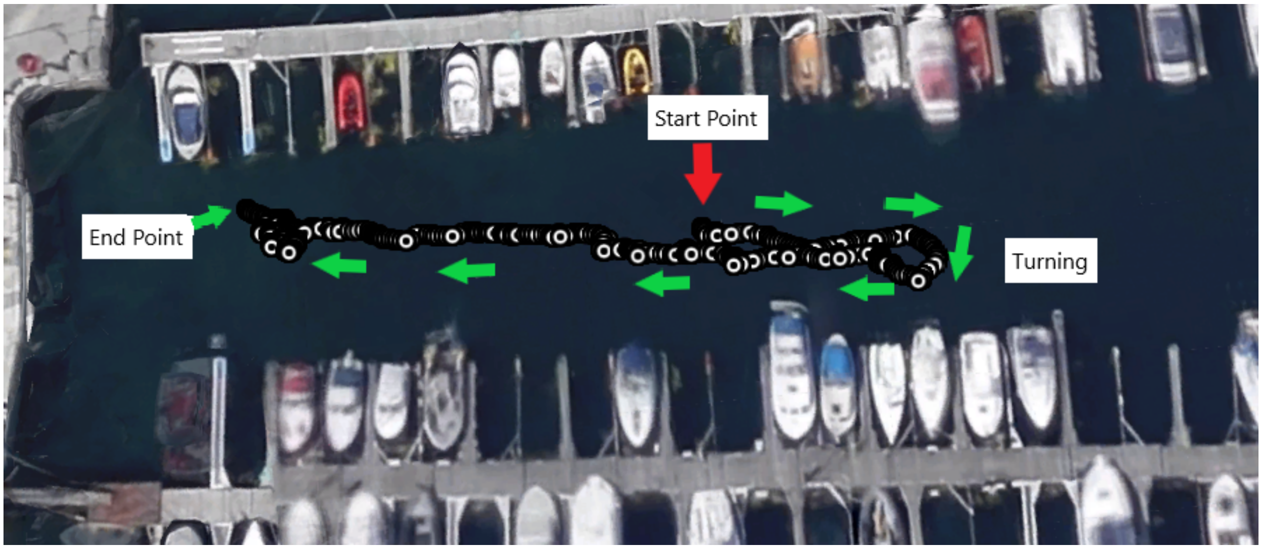


Figure 4.33: Autopilot travel route, when turning it have reached its first point and is heading for the second point

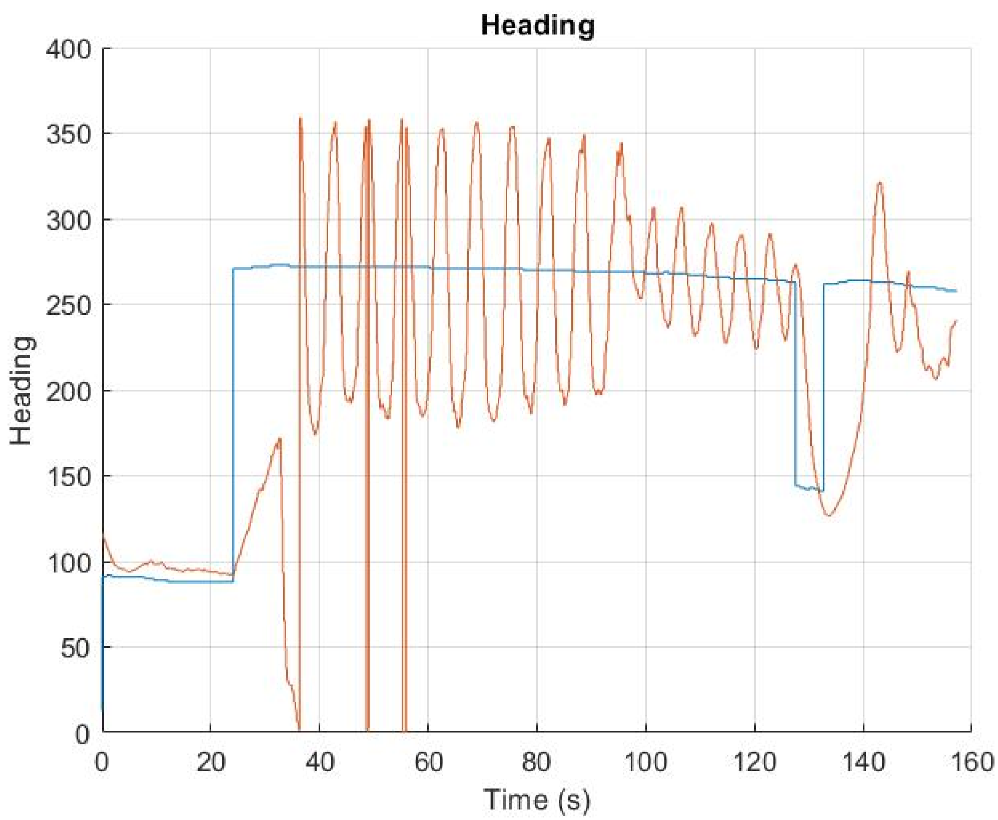


Figure 4.34: Heading Autopilot

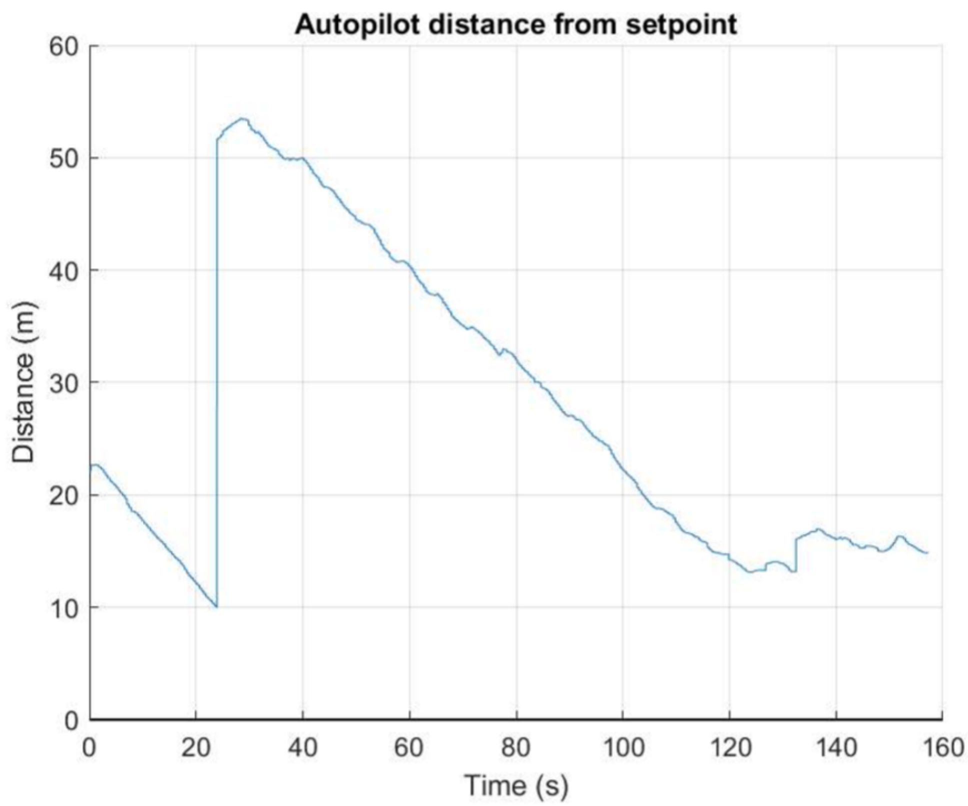


Figure 4.35: Distance From Setpoint, USV get a new destination at 22 seconds.

DP - Result

The DP was tested in two different ways. The firsts test was to put the USV in DP-mode to hold its current position, this day there were waves in the water, and boats driving past which affected the vessel, data from the GPS and set point was logged. In figure 4.36, the result from this test is shown. The second test was with attaching a rope to the platform and pull it away from its setpoint. In figure 4.40 shows the second test result.

DP test 1

DP test 1 was a static test. The platform was sat in DP to check if it would hold its position. The result was that it would draft and be around 1 meter from the setpoint. Figure **??** shows how far the USV was from the setpoint and figure 4.37 show five logged point plotted in a map to show the position to the USV from the setpoint.

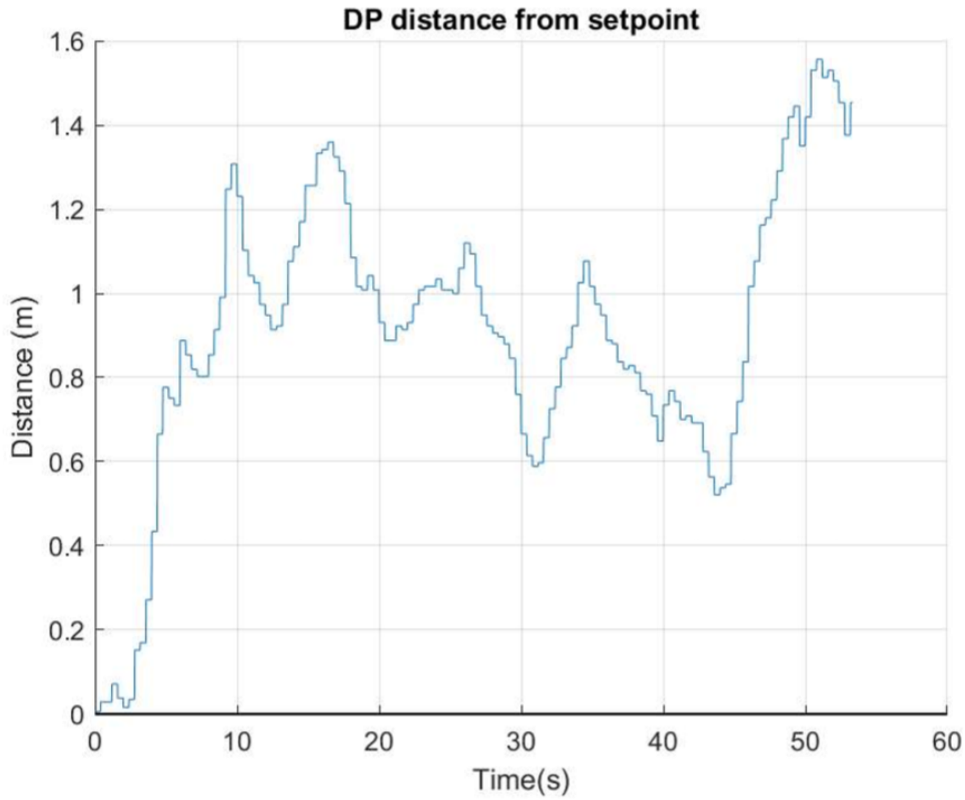


Figure 4.36: DP test, the graph shows that the platform will be about 1 meter away from the setpoint

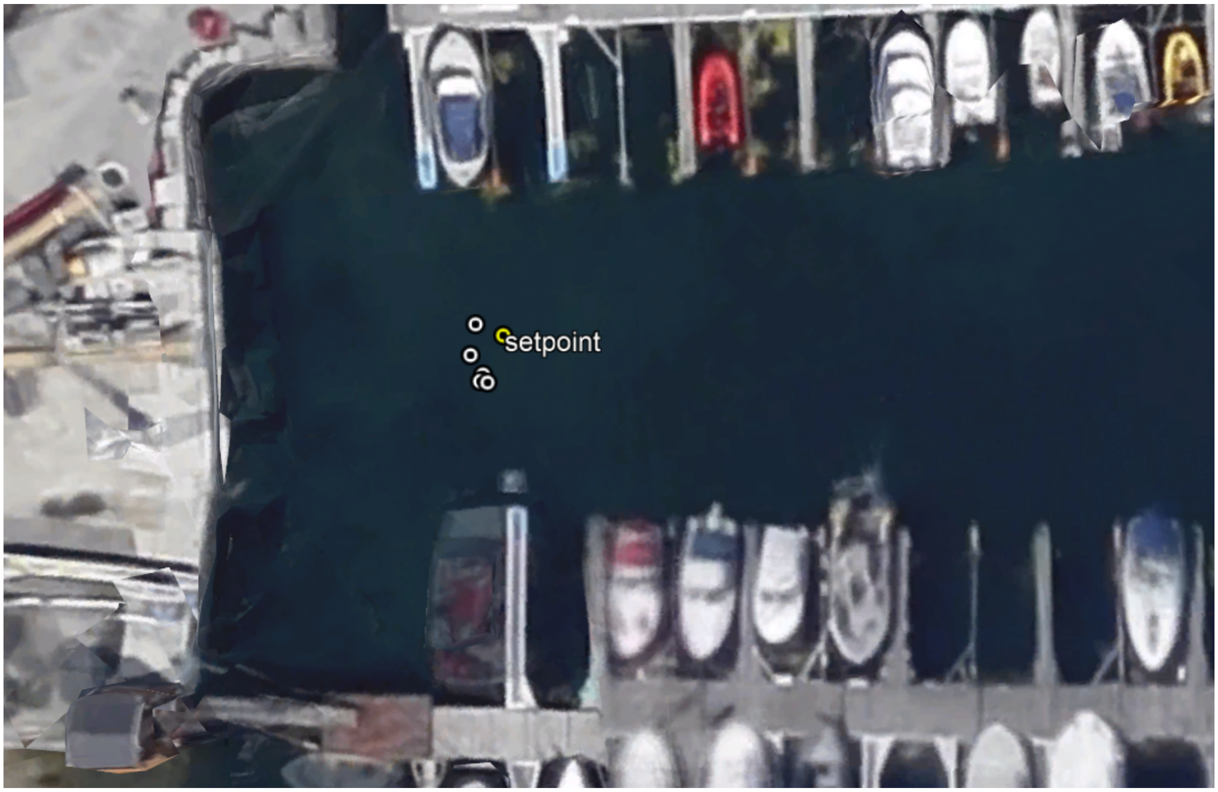


Figure 4.37: Graphical presentation of 5 logged points

DP test 2

The DP test 2 was performed by attaching a rope to the USV. The USV was then sat in DP mode and pulled away from its DP point. Figure 4.38 shows the setpoint the USV was sat in DP mode. Redpoint was the point where the USV was pulled to. Figure 4.39 shows show the platform maneuvered back to its position. Figure 4.40 shows how far away the USV was and the time it took to get back to the setpoint.

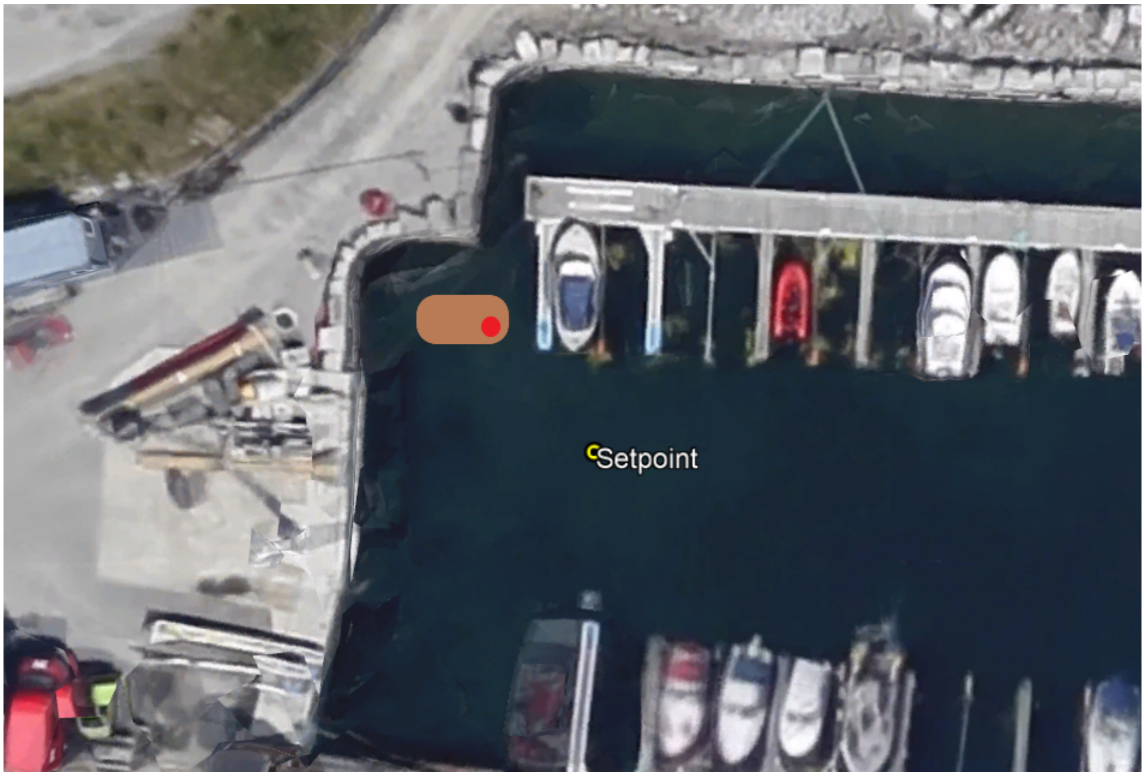


Figure 4.38: Start position

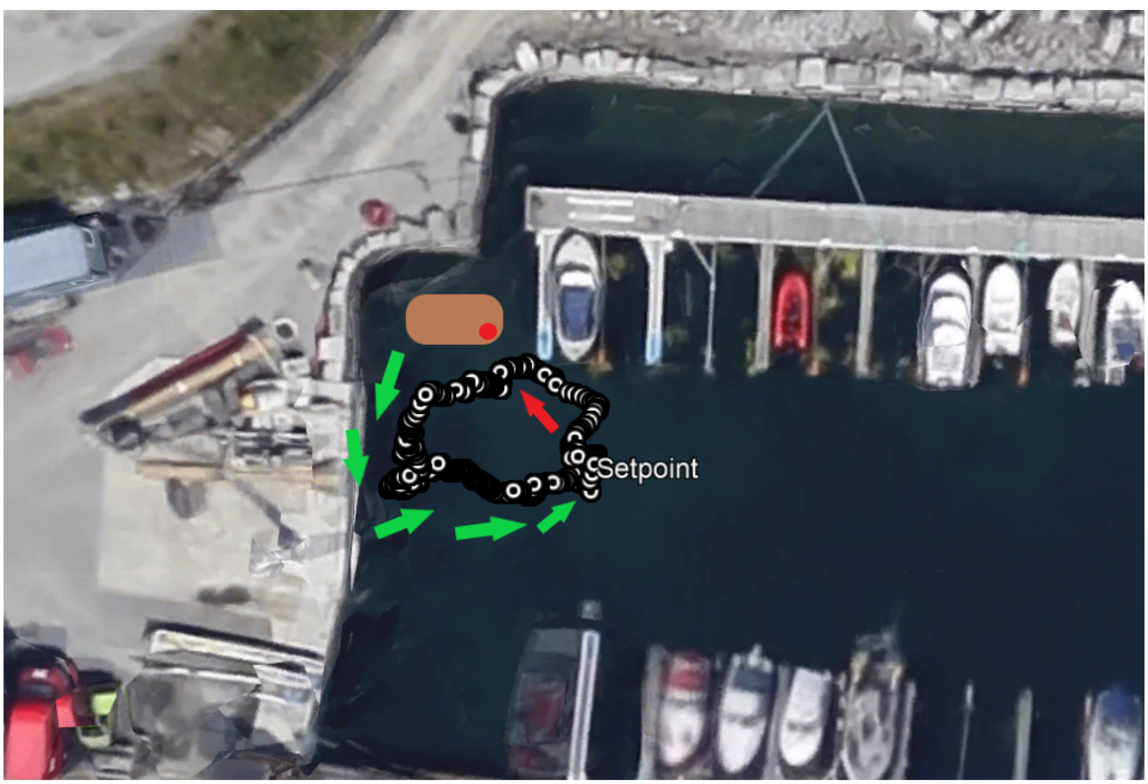


Figure 4.39: The USV Setpoint, Red point is the point it got pulled to.

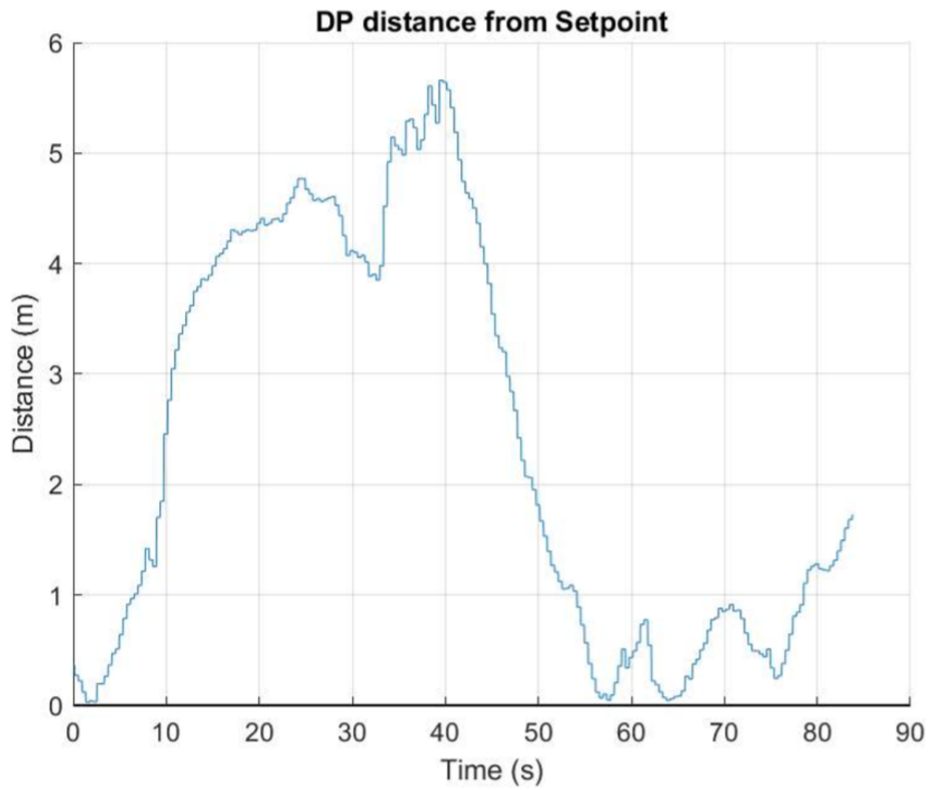


Figure 4.40: Shows distance from setpoint and time

4.10 ROV Test

The ROV was tested in shallow waters due the tide, the dept was measured to 1.90 meters. The ROV was released donwn in the water by using the GUI. All the different directions on the ROV was tested and was working, Figure 4.41 the ROV is launched down into the water.

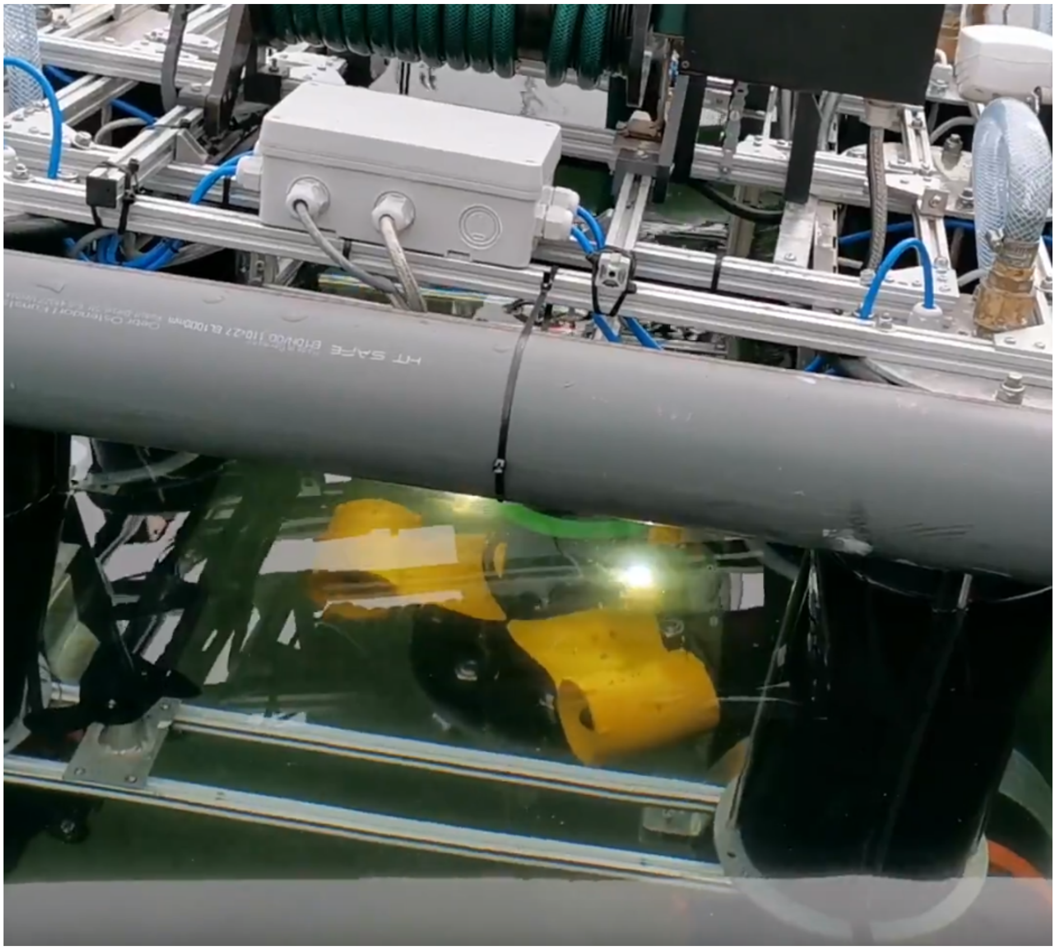


Figure 4.41: ROV in water

Chapter 5

Discussion

5.1 Results from testing

5.1.1 Mechanical components

No mechanical components was broken during testing, the pipes for the batteries stayed dry inside the whole sea trial, and the winch shows no signs of degradation. No electrical component got wet during the testing. This shows that the system is mechanically strong, even tho the platform is built using of the shelf parts.

5.1.2 GUI

Controlling the USV, ROV, and winch from the GUI showed to be a good solution, and was easy to use. With the low latency from the video stream, controlling the USV from a remote location would work. To achieve a better visual image from the USV it would be an idea to replace the camera with a wide angled camera that can be rotated from the GUI. As mentioned in section 4.6.1 the goal was that the user would not get overloaded with unnecessary information from the systems. During testing it came clear that some information could be added to the platform and ROV tab on the GUI. By doing this the user does not have to change to the overview tab to find the information they need. The information is there already, it just needs to be moved to a different tab on the GUI. Even though it is possible to log and save data in the PLC application. The GUI could have its own logger for sensor data, this would increase the user experience since the user would not have to rely on the PLC for logging data, this function is simple to implement, but was not a priority due to time restraints.

5.1.3 Control system

Using a PLC for the control system was indispensable during testing of the systems, both before and during sea trails. A visual programming tool where real time values can be read and changed made it easy to debug and correct issues during testing. Also with the built in ability to log the behavior of the systems, it was possible to see what was working, and not during testing as shown in figure 4.31 where the DP system failed.

5.1.4 Autopilot

The autopilot worked when plotting way-points on the GUI and watch it move to the given position. The PID controller for the autopilot could be adjusted better so the USV doesn't oscillate so much, as seen in figure 4.34. When reaching the first way-point the USV continues to the next one, but if the way-point is changed it still goes to the first way-point. During testing there was some limitation on space inside the harbour, therefore it was challenging to complete an autopilot test, often the test had to be aborted to avoid collision with other boats in the area. It was harder than expected to plot a accurate position on a map when dealing with small spaces, especially when it is at sea. There are no reference points like houses or buildings in the map to help the user to plot an accurate position of the way point.

5.1.5 Dynamic positioning

The dynamic positioning worked somewhat as planed. Although the USV were able to hold it's position, it used a long time to return to initial position when pulled far away from the set-point, ass seen in figure 4.40. When the USV is holding it's position it does not move the shortest way from current position to the set-point. This is because when in DP mode it always tries to keep the heading to North, therefore it only get movement in the X or Y direction, as seen in figure 4.39.

5.1.6 PID calibration

Due to limited battery power, and limited space in the harbour there were no time to calibrate the PID controllers to make the control system stable, the plan was to use Ziegler-Nichols method to calibrate the PID controllers, but it proved to be hard due to the short time the system could run before the test had to be aborted due to other activities in the harbour or low battery.

5.1.7 Test results ROV

The ROV was working well during testing. The controls were responsive,and it was possible with ease to maneuver the ROV to desired positions by using the video stream. Since the ROV is hanging from a cable, the ROV will twist the cable when rotating, the ROV rotates back to default orientation when the thrusters is set to zero. The function of the ROV would be better if the ROV could stay in its current position even if the user releases the button in the GUI. The thrusters on the ROV is powerful, during testing the thruster power on the ROV was set to 50 percent in the GUI.

5.1.8 Test result winch

The winch system worked well during testing, the winch parts that was replaced was holding the weight of the ROV without breaking. During testing the winch was responsive with the control from the GUI and lowered and docked the ROV. One issue that have to be addressed is that the ROV gets stuck on the way down in the water when the thruster hits the flanges on the vertical pipes. The sensor that indicates that the ROV is in upper position will sometimes not indicate due to the angle the ROV is coming up from the water. A new placement of this sensor should be considered.

5.2 Issues to be addressed

5.2.1 Stability

The stability should be addressed if there will be further work on the USV. The pumps could be replaced with some that can adjust the speed so it is easier to regulate the amount of water in the pipes. First and foremost the platform should be stable without the use of the stabilization system.

5.2.2 Battery capacity

To achieve longer run-time of the system the battery package should be upgraded. Larger batteries also means that the weight goes up, which is desirable as mentioned in section 4.9.2. This will lower the centre of gravity and gain additional run-time.

5.2.3 Plotting destinations on the map

Since it is hard to plot an exact position on the map in the sea, there should be added functions to save frequently visited positions. This function could save the current position of the platform, and the user could add a name to the position. The next time the user wants the platform to travel to this position, the user can pull the position out from a list and use it as a set point.

5.2.4 ROV

Orientation control

It should be added possibilities to lock the ROV heading or even position in the water. This could be done using the integrating IMU.

Chapter 6

Conclusions

The objectives were to integrate three control-systems to one and solve the challenges that occurred while doing so. The integration shows that by changing the control system to a more robust and industrial standard makes it easier to implement and test the functionality of the system, and increases the flexibility for further development. The results from testing described in chapter 4 shows that the USV can obtain it's position in DP mode, and move on autopilot between way-points plotted on a map in a user friendly GUI. By using the winch and docking system the ROV can be deployed and retrieved from the USV, and the system will lock the ROV in position automatically when the ROV is in correct position for docking. The ROV is integrated in the complete system, but the USV is not dependent of the ROV to be functional which is opening opportunities for different usage of the platform. Different solutions for controlling the stabilization system was tested, but the results was limited due to the vessels stability in the water.

6.1 Further work

- Change the batteries to batteries with larger capacity.
- Change size of the vertical pipes on the platform for better stability.
- Adjust the PID regulators on both the Autopilot system, and the DP system.

- Develop logging functionality on GUI.

Appendices

A Preproject report

PREPROJECT - REPORT

BACHELOR THESIS

NTNU
Department of ICT and Natural Sciences

TITLE:
Integration of Aquaculture inspection platform

CANDIDATES(NAMES):				
Lars Even Sætre, Sigurd Olav Liavåg, Ole Morken				
DATE: **09.01.2019**	SUBJECT CODE: **IE303612**	SUBJECT: **Bachelor thesis**		DOCUMENT ACCESS: - Open
STUDIUM: **AUTOMATION TECHNIQUE**			NR PAGES/APPENDIX: /	BIBL. NR: - Not in use -

PRINCIPALS/SUPERVISORS:
Ottar L. Osen, Houxiang Zhang

SUMMARY:

Postadresse	**Besøksadresse**	**Telefon**	**Telefax**	**Bankkonto**
Høgskolen i Ålesund	Larsgårdsvegen 2	70 16 12 00	70 16 13 00	7694 05 00636
N-6025 Ålesund	**Internett**	**Epostadresse**		**Foretaksregisteret**
Norway	www.hials.no	postmottak@hials.no		NO 971 572 140

CONTENTS

1 INTRODUCTION

In maritime industry, inspection and maintenance is an important task to keep equipment in good condition in order to reduce downtime in the production and prevent losses due to construction damage.

Underwater inspection is a demanding process due to rough conditions at sea. Traditionally the industry used divers to inspect equipment underwater, but this is both dangerous for the divers and it is ineffective. Divers can't work for a long period of time and need rest before they can work again.

The use of ROV technology to do inspection and maintenance is widely used today, but the size of the ROV and the equipment needed to operate it is expensive. Traditionally the ROV is launched and operated from a large offshore vessel. Because of this the use of ROVs are mostly limited to the offshore industry.

This project aims to make a system that can bring advanced ROV technology that personnel can use without much training. By making an easy to use automatic ROV system which can move and operate in rough conditions.

The ROV will be carried by a platform which will navigate around in the work area by using GPS, and laser sensors (lidar). The platform can deploy the ROV using a winch when it reaches the correct working position.

The platform and ROV will be remotely controlled by an operator on a work station that will display live camera feed and sensor data.

2 NOTION

DP – Dynamic Positioning
ROV – Remotely Operated Vehicle

3 PROJECT ORGANIZATION

3.1 Project members

Studentnummer(e)
Ole Grytdal Morken Lars Even Sætre Sigurd Olav Liavåg

3.1.1 Project tasks – organization

Members

Lars Even Sætre – Project leader
Ole Grytdal Morken - Secretary
Sigurd Olav Liavåg – Group member

Platform

- Check condition - Test Software/Hardware
- Electrical Wiring
- Buoyancy
- Stabilization
- DP - Dynamic Positioning
- Autonomous System
- Lider, Scanse Sweep
- Camera
- Integrate with control system

ROV

- Check condition - Test Software/Hardware
- Integrate with control system

WINCH

- Check condition - Test Software/Hardware
- DockingHead
- Integrate with control system
- Fix broken parts, 3D-Print?

Common Control Unit

- Design Common Control Unit
- Integrate with all three systems (Platform, ROV, Winch)

3.1.2 Project leader responsibilities

Keep the project plan up to date with progress on the ongoing tasks.

3.1.3 Secretary responsibilities

Write weekly reports on the progress of ongoing task and the progress.
Write a short report from each meeting with supervisors.

3.1.4 Other group members responsibilities

All group members will regularly update the progress of their task and assist the secretary on the weekly reports.

3.2 Supervisors (veileder og kontaktperson oppdragsgiver)

- Ottar L. Osen
- Houxiang Zhang

4 AGREEMENT

4.1 Agreement with supervisors

In the first meeting with the supervisors we agreed on some terms for the project.

- We will write a weekly report of what we have done in the week, and what we will do the following week. Also, we will show how we are progressing according to the project plan.
- If we have problems and needed help, we will contact the supervisors and set up a meeting.
- Every second week on Thursdays at 10 am, we will have a meeting with the supervisors, were we discuss progress and planning of the project. The meeting is set to last 30 minutes.
- If we need to buy any equipment or parts, we will contact the supervisors and get founds, within a reasonable limit.

4.2 Workplace and resources

During the beginning of the project the platform is located at "plastlab".

We have access to plastlab Mondays- Fridays between 08:00-16:00. The plan is to move the platform to "tunglab" later in the project, this is because we have more access to tools there and we have access to the room for longer periods of the day. Tunglab is also closer to the water tank, which we are using for testing.

We have also been granted access to L167 project room Tuesdays – Fridays the whole project period. This room will we use for project planning, and computer work. This room is equipped with monitors, mouse and keyboard.

4.3 Group Norms – Agreement on cooperation – Attitudes

To reach the goal of the bachelor we have agreed upon some rules. The workdays will be from 08:00 to 16:00. Location is NTNU Aalesund. If a group member cannot meet, he must notify the remaining group members accordingly, and update the members on the status of the work he's been working on.

There will be set of at least one hour in the end of the week to write weekly report.

All group members will treat each other with respect and listen to each member opinions on project maters.

All group members shall have insight in each other's work. This way it is easier for the rest of the group to help if a member is stuck in its work.

We should follow the progress plan, if we can see that we are not able to finish what we set out to do, we need to come together and select which parts of the project we should make a priority.

All members in the group are dependent of each other.

5 PROJECT DESCRIPTION|

5.1 Problem - goal – purpose

This project is based on and a continuation of a bachelor thesis done in 2017 called "Sea farm platform" done by graduates at NTNU Ålesund. Their project contained a platform which could maneuver around using GPS, and remote control from a tablet.

Since then, the platform has been upgraded with a winch which was built in mechatronics course in 2017, and a ROV which was built in mechatronics course in 2016. These components are driven by separate control systems, and they are not integrated to the control system of the platform.

The goal of this project is to integrate all the component into one system, that can be monitored and controlled from a remote location. And improve the current systems with more functionality.

The purpose of this project is to make a cheap and easy to use autonomous platform that can be used in the industry for underwater inspections.

5.2 Solution requirements or project results – specification

The finished results of this project will be a complete system integration of the previous systems. This includes the sea platform, the winch and the ROV. The system will have a common control unit with options to view the different video streams, remote manual control of winch, ROV and platform, autonomous mode and DP for the platform.

A new GUI will be implemented for video stream, to show sensor data and to switch between different operational modes for the platform, ROV and winch.

Make the electronics on the platform sea worthy by waterproofing some of the connection boxes. Also fix some of the cable wiring to make it look more presentable.

For the autonomous mode to work properly for the platform a 360-degree laser sensor will be fitted, and a collision avoidance system will be implemented to the system.

For the dynamic positioning (DP) mode a GPS signal that provides its real time position will be used so the platform can hold its position while the ROV is in operation.

When the ROV is not in operation mode it will be secured in a docking head mounted on the platform. The already implemented solution will be upgraded to handle the weight of the ROV and withstand corrosion from seawater.

5.3 Scheduled progress for development - method(s)

5.4 Information gathering

Since this project is a continuation of earlier projects, we have gathered reports and documentations from the Platform, ROV, and the Winch.

- Bachelor sea farm project. 02.06.2017
- Simple winch for seafarm. 23.11.2017
- Aquafarm inspection – ROV. 30.11.16
- A Novel Low Cost ROV for Aquaculture Application. *OTTAR L. OSEN*

We have also acquired the source code from these projects.

5.5 Risk analysis

There is allot that can go wrong when doing a group project, we are confident that all the group members will work hard to finish the project in time. But working hard will not help if we are not efficient or prioritizing the right tasks.

Another issue we must make sure that we are aware of is the time it takes for parts to arrive after we order them, if we are missing a crucial part it could halt the progress of the project, therefore it is important to order parts that we know we need as early as possible.

Everyone gets sick, sickness is very hard to predict. If a member is away for days due to sickness, it will have huge impact on the weekly goal of the project since we are only three students. If a member is sick, he must ask himself if he is able to work from home or do different work that he may be able to do even if he is sick.

Research before starting the work. We must do research to make sure that the plan we set is even possible. If we use weeks to make software for a controller, it could be complete waste if it shows that the hardware is not able to run the software, or support the other components in the system.

5.6 Main activities

- Beskrivelser av planlagte hovedaktiviteter og viktigste delaktiviteter for gjennomføring av prosjektet.

Nr	Hovedaktivitet	Ansvar	Kostnad	Tid/omfang
A1	Platform	Ole		1 uke
A2	ROV	Sigurd		1 uke
A3	Winch	Sigurd		1 uke
A4	Common Control Unit	Lars		1 uke

5.7 Progress management

5.7.1 Master plan

Report

- Weeklys update from all group members.

Platform

- **Test the software for the platform**
  Check the software

  Start date: 10.01.2019
  Estimated date of completion: 25.01.2019

- **Check hardware functionality and fix problems**
  Check hardware (Thrusters, IMU, Pumps)

  Start date: 10.01.2019
  Estimated date of completion: 25.01.2019

- **Replace batteries?**

  Replace batteries with new.

  Start date:
  Estimated date of completion:

Make electrical wiring more water-resistant
Replace wiring.

Start date:25.01.2019
Estimated date of completion: 06.03.2019

- **Design new Control system**
- The new control system will be using PLC and Raspberry pi 3B+.
- Start date: 28.01.2019
  Estimated date of completion: 06.03.2019

- **Bouyancy**

  Start date: 07.03.2019
  Estimated date of completion: 07.03.2019

- **Stabilisation**

  Start date: 08.03.2019
  Estimated date of completion: 15.03.2019

- **Continue development of the DP system**

  Start date: 18.03.2019
  Estimated date of completion: 05.04.2019

- **Autonomonus System**

  Start date: 05.04.2019
  Estimated date of completion: 26.04.2019

- **Implement laser sensor (Lidar) for collision avoidance in autonomous mode**

  Start date: 05.04.2019
  Estimated date of completion: 26.04.2019

- **Install a better camera solution.**

  Start date: 29.04.2019
  Estimated date of completion: 03.05.2019

- **Integrate with common control system**

  Start date: 29.04.2019
  Estimated date of completion: 06.05.2019

- **Testing/Tuning**

  Start date: 29.04.2019
  Estimated date of completion: 10.05.2019

ROV

- **Test existing software.**

  Start date: 10.01.2019
  Estimated date of completion: 25.01.2019

- **Replace old Raspberry pi with new one (Raspeberry pi 3B+)**

  Start date: 28.01.2019
  Estimated date of completion: 31.01.2019

- **Make changes to the software so it can be implemented in to the main system.**

  Start date: 01.02.2019
  Estimated date of completion: 05.02.2019

Winch

- **Check condition - Test Software/HardwareStart date:**
  Start date: 03.01.2019
  Estimated date of completion: 17.01.2019

- **Fix DockingHead, 3D-Print**

  Start date: 17.01.2019
  Estimated date of completion: 04.02.2019

- **Fix broken parts on winch, 3D-Print**

  Start date: 17.01.2019
  Estimated date of completion: 04.02.2019

- **Integrate with common control system**

  Start date: 15.04.2019
  Estimated date of completion: 18.04.2019

Common Control Unit

- **Design a common controller for all systems**
  Design a good solution that is portable and easy to use. Portable.

  Start date: 04.02.2019
  Estimated date of completion: 15.02.2019

- **GUI with video feed from ROV and Platform, and display relevant sensor data**

  Start date: 04.02.2019
  Estimated date of completion: 15.02.2019

- **Manual control for winch, ROV and platform**

  Start date: 04.02.2019
  Estimated date of completion: 15.02.2019

5.7.2 Project management tools

To keep an overview and a plan schedule, we will use a Gantt chart. This will show the dependency relationships between activities and current schedule status.

5.7.3 Development tool

- Netbeans IDE to develop java software.
- Arduino IDE to develop Arduino code.
- 3D drawing software.
- E!cockpit
- Python
-

5.7.4 Internal control – evaluation

Progress will be updates continuously, this will be done by all group members.

A progress is completed when the given task fulfils its requirement.

5.8 Decision-making process

Decisions will be made by all group members.

6 DOCUMENTATION

6.1 Reports and technical documents

Weekly report updates on how the progress of the week went.

7 PLANNED MEETINGSR AND REPORTS

7.1 Meetings

7.1.1 Meeting with supervisor

Meetings with supervisors are scheduled to every Thursday morning. There the progress report will be presented. And also, if there are some issues that has to be solved will be presented.

7.1.2 Project meetings

Project meeting every Monday morning and Friday afternoon. On Monday morning short recap on last week progress, and the goal for the week. On Friday afternoon it will be a meeting about how the week went.

7.2 *Periodic reports*

7.2.1 Progress reports (incl. milestones)

A short progress report will be written every week, this report will be shared with supervisors. Every two weeks when the group have a meeting with the supervisors the report will be more detailed.

8 TREATMENT OF NONCONFORMANCE

Each group member is responsible for their own tasks, if a group member has issues completing the task in due time he will ask for assistance from the group.

The group can decide to do changes to the overall plan while still maintaining the goal of the project.

9 EQUIPMENT REQUIREMENTS / CONDITIONS FOR IMPLEMENTATION

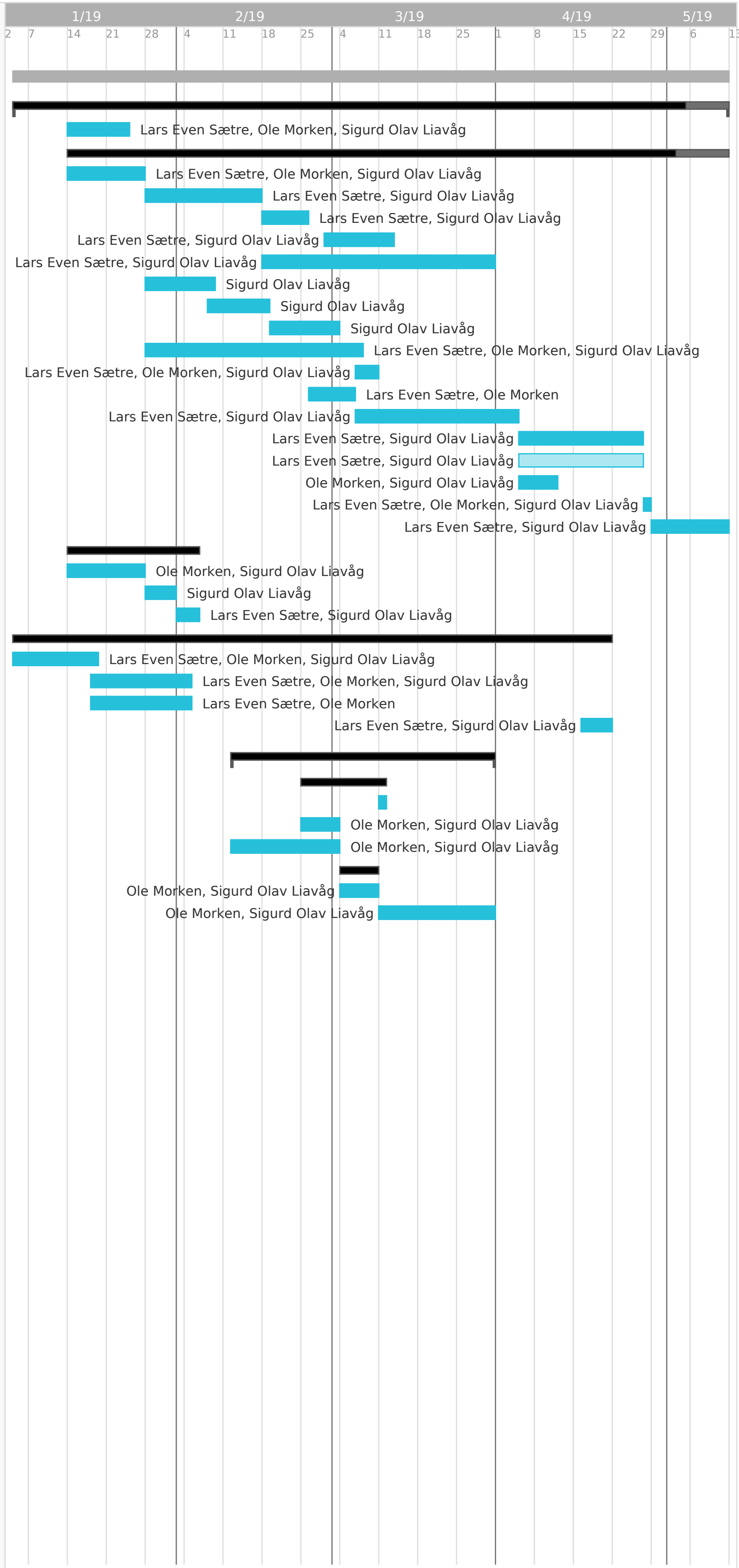
Testing the platform in water:

- Access to the water tank at NTNU
- Transport to sea trial
- Crane for moving the platform during sea trial
- Boat for accessing and monitoring the platform during sea trial.

B Gantt diagram

teamgantt
Created with Free Edition

Task	Hours	Progress
Aquaculture Platform	**0h**	**95%**
Aquaculture Platform Bachelor	**0h**	**94%**
Preproject	0	100%
Platform	**0h**	**92%**
Check condition - Test Software/Ha...	0	100%
PLC Research	0	100%
PLC component testing	0	100%
PLC GUI - Used for testing	0	100%
Wago simulation	0	100%
Raspberry pi Research	0	100%
Raspberry pi component testing	0	100%
Raspberry pi Complete Code	0	100%
Electrical Wiring Platform	0	100%
Bouyancy	0	100%
PLC Stabilization	0	100%
PLC/Raspberry - DP - Dynamic Posit...	0	100%
PLC/Raspberry - Autonomonus Sys...	0	100%
Lider	0	0%
Camera	0	100%
Integrate with common control sys...	0	100%
Testing/Tuning	0	100%
ROV	**0h**	**100%**
Check condition - Test Software/Ha...	0	100%
Replace Raspberry pi	0	100%
Integrate with common control sys...	0	100%
WINCH	**0h**	**100%**
Check condition - Test Software/Ha...	0	100%
Fix DockingHead, 3D-Print	0	100%
Fix broken parts on winch, 3D-Print	0	100%
Integrate with common control sys...	0	100%
Common Control Unit.	**0h**	**100%**
GUI research	**0h**	**100%**
Virtual Joustick	0	100%
Video Stream	0	100%
GUI Frontend	0	100%
GUI Backend research	**0h**	**100%**
UDP Multithreading	0	100%
GU Backend programming	0	100%



C Meeting report 10.01.19

Referat: Møte med veiledere Bachelor

Tilstede: Ole Morken, Lars Even Sætre, Sigurd Olav Liavåg, Dr Houxiang Zhang, Ottar L Osen

Ikke tilstede: N/A

Agenda:

Design a rough scope for the BSc thesis.

- Need a good working plan, should be reasonable, follow it strictly
- What reascorces we need, parts to be made, things to be orderd, help from others.
- Concrete working plan
- Need a project leader
- Help eachother, one responisble, one assistant
- Write an agreement on how to work, when to show up etc
- Need to know what we want to do
- Write down clearly what we want to do
- Present the work in a propper way, scientifically clearly

- Integration : Test everything, fix damage, minimum integration

- Improvement: Integration, New functions, Whats the wheigt?

- TEST: Whole system integration test? Measure the force on the platform

- Documentation

- Different modes, moving, docking, operation

- Should have DP, also waypoint autopilot movement

- Have several solutions to each improvement/solutions

- Weekly report, short, simple, what we have done, how we are according to plan

- Meatings every seacond week, Thursday 10:00-10:30

D Meeting report 24.01.19

Referat: Møte med veiledere Bachelor

Tilstede: Ole Morken, Lars Even Sætre, Sigurd Olav Liavåg, Dr Houxiang Zhang, Ottar L Osen

Ikke tilstede: N/A

Agenda:

1. What we have done so far.
   - Presented to the supervisors what we had done for the past couple of weeks.
   - The previous reports have been read to get a understanding of what they did, and why they did it.
   - Testing of the subsystems have been done, and some issues have been found regarding broken parts.
   - Presented some ideeas to what we wish to du, ex. Use a PLC to control the new system.
2. What we will do for the next weeks.
   - Continue to test the system
   - 3D print some broken parts
   - Get help to manufacture som parts
   - Make a list over the parts we need and present it to Ottar
   - Get Andres to order the parts we need

E Meeting report 07.02.19

Referat: Møte med veiledere Bachelor

Tilstede: Ole Morken, Lars Even Sætre, Sigurd Olav Liavåg, Dr Houxiang Zhang, Ottar L Osen

Ikke tilstede: N/A

Agenda:

1. Sponsor
   - There are no restrictions on sponsors to the project as far as we know.
2. Deler til WAGO evt ny PLC
   - If we can't find the PLC modules we need, they can be baught.
3. Skjerm
   - We can look at tablet alternatives for the platform and present them to the supervisors.

Buying a cabinet is not a problem

Bouyancy is a issue, we need to increese the boyancy

Ottar want to increese the height of the vertical pipes

Do calculations with NX, ask for help from shipdesign

Microsoft surface is a posibility for GUI

We can use the PLC we foud at the lab

Need a canopen/bus card for PLC to connect to the steppermotor controll card

Update the gantt scheme, add more subtasks

F Meeting report 08.03.19

Referat: Møte med veiledere Bachelor

Tilstede: Ole Morken, Lars Even Sætre, Sigurd Olav Liavåg, Dr Houxiang Zhang, Ottar L Osen

Ikke tilstede: N/A

A short recap of what we have done the last couple of weeks.

Increase bouyancy by increasing the diameter of the buttom pipes, this also opens up the oportunity to add larger batteries for more runtime.

Add calculations for the extra weight of larger batteries for further work on the platform.

Add «on a dime» turning for the platform.

Try and use wago plc for stepper motor control or make a stepper control program in e!cockpit.

G Meeting report 21.03.19

Referat: Møte med veiledere Bachelor

Tilstede: Ole Morken, Lars Even Sætre, Sigurd Olav Liavåg, Dr Houxiang Zhang, Ottar L Osen

Ikke tilstede: N/A

Planning to finish assembly next week

Mostly software work done the last few weeks

The stepper motor can can be turned, to make it pull to ground. Need a pull up resistor. Send to ottar, he Wil sketch up. Can also talk to Ivar.

Check with andre to plastic weld the new pipes

Check with andre about thick plexiglass

H Meeting report 05.04.19

Referat: Møte med veiledere Bachelor

Tilstede: Ole Morken, Lars Even Sætre, Sigurd Olav Liavåg, Dr Houxiang Zhang, Ottar L Osen

Ikke tilstede: N/A

Agenda:

1. Bachelor thesis report
   2 papers published in ocean conference. Ottar will link it to us.
   Only wrote what they did, not what they did not do, and why they want to do it.
   We need something, why we need it, different solutions, (scalable) set up a table(a, b, c ,d) advantage, disadvantage
   What you want to do, why you want to do it, what you did, how it turned out.

   Introduction is IMPORTANT, decides the grade

   Introduce fish industry first, easy to read.
   Compare against the last bachelor report
   Text on all figures!!!

   Important to reference to books/papers

   User manual for the user
   User manual for next group/maintenance

I Meeting report 02.05.19

Referat: Møte med veiledere Bachelor

Tilstede: Ole Morken, Lars Even Sætre, Sigurd Olav Liavåg, Dr Houxiang Zhang, Ottar L Osen

Ikke tilstede: N/A

Do a bullardpull test in all directions

Log the speed of the platform

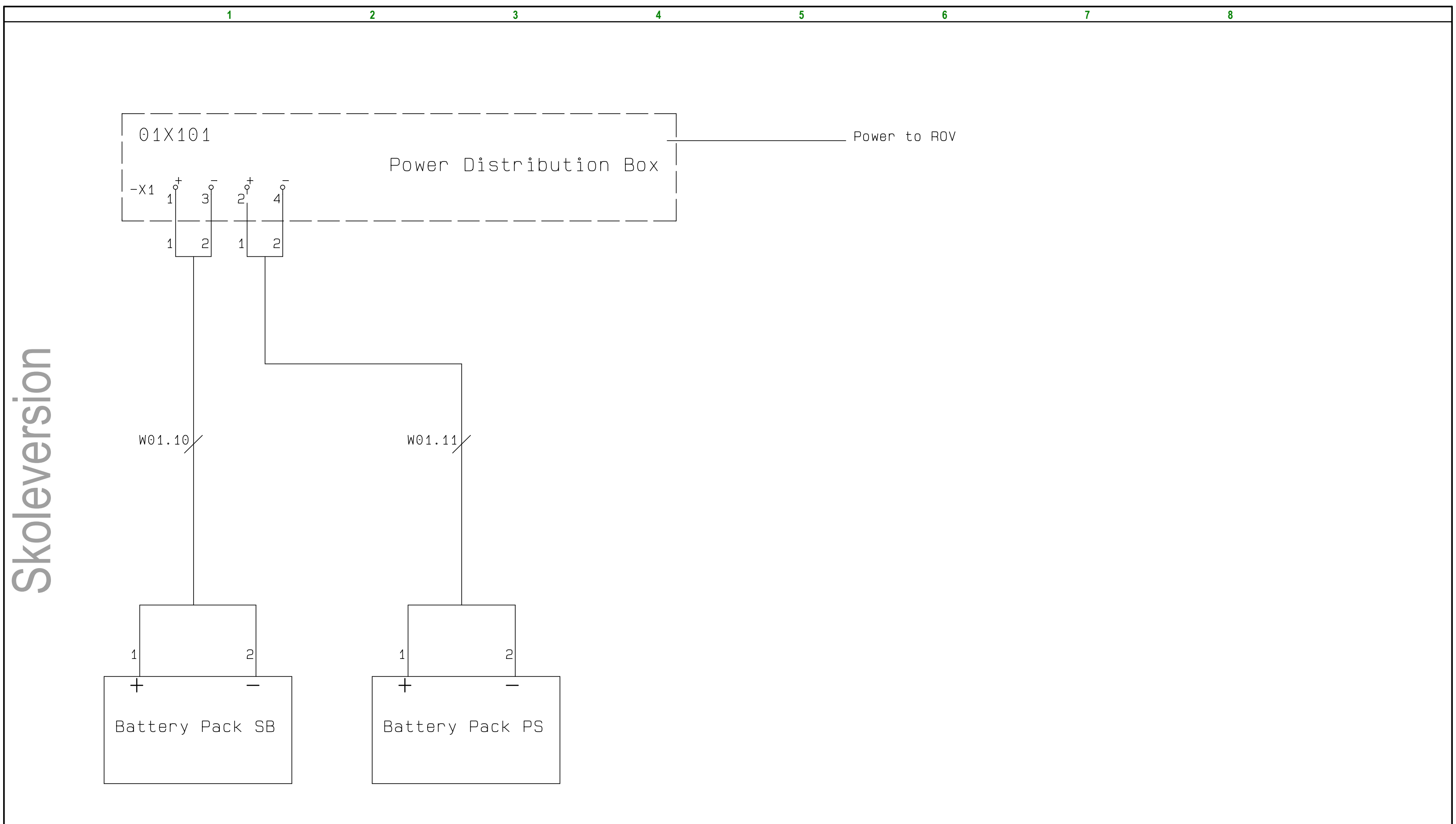
Test ROV while the platform is floating, and if the platform can compensate for the pull of the ROV

Log the power output

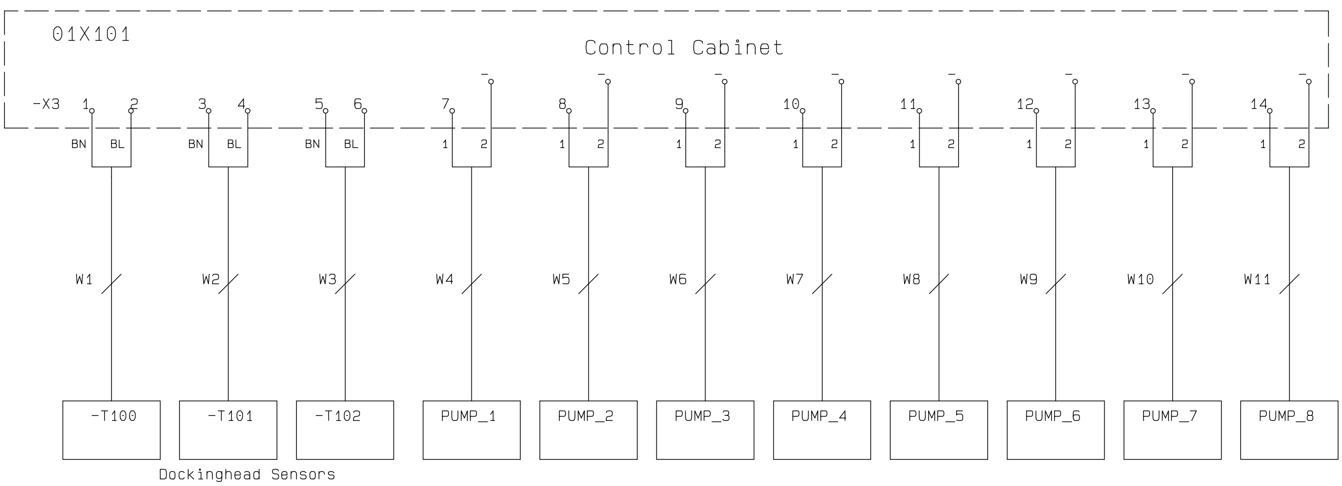
J Electrical drawings

Page 5 - 8

Diagrams

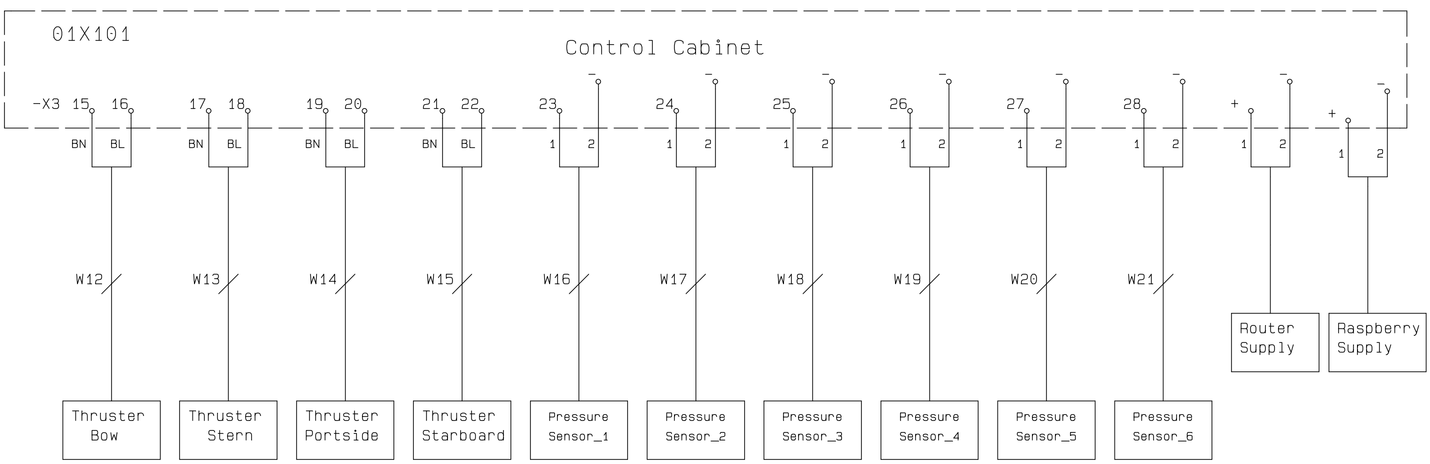

1
2
3
4
5
6
7
8
01X101
Power Distribution Box
Power to ROV
-X1
+
-
+
-
1
3
2
4
1
2
1
2
W01.10
W01.11
1
2
1
2
+
−
+
−
Battery Pack SB
Battery Pack PS

1 2 3 4 5 6 7 8

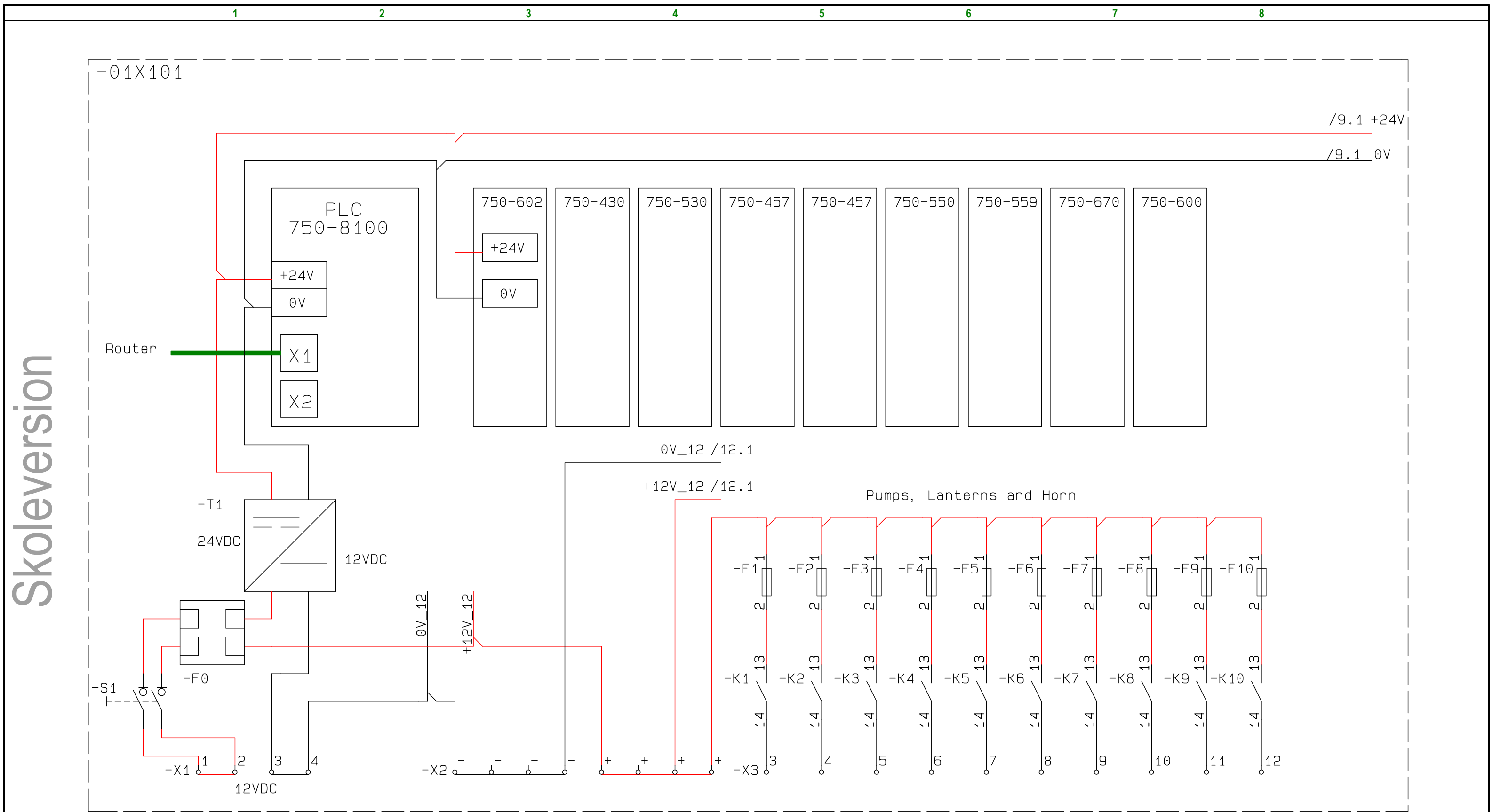


PCSCHEMATIC Automation

Project title:	**Integration of Aquaculture Inspection Platform**	**Project no.:**	**Project rev.:**		**Page**	**6**
Customer:	NTNU in Aalesund	DCC:			Scale:	1:1
Page title:	DH_Sensors and Pumps	Dwg. no.:	Page rev.:		Previous page:	5
File name:	AIP_ElectroDrawings	Eng. (proj/page):	Last print:	19.05.2019	Next page:	7
Page ref.:		Appr. (date/init):	Last edit:	24.02.2019	Total no. of pages:	32



Project title:	**Integration of Aquaculture Inspection Platform**	**Project no.:**	**Project rev.:**		**Page** 7
Customer:	NTNU in Aalesund	DCC:			Scale: 1:1
Page title:	Thrusters and Pressure Sensors	Dwg. no.:	Page rev.:		Previous page: 6
File name:	AIP_ElectroDrawings	Eng. (proj/page):	Last print:	19.05.2019	Next page: 8
Page ref.:		Appr. (date/init):	Last edit:	19.05.2019	Total no. of pages: 32



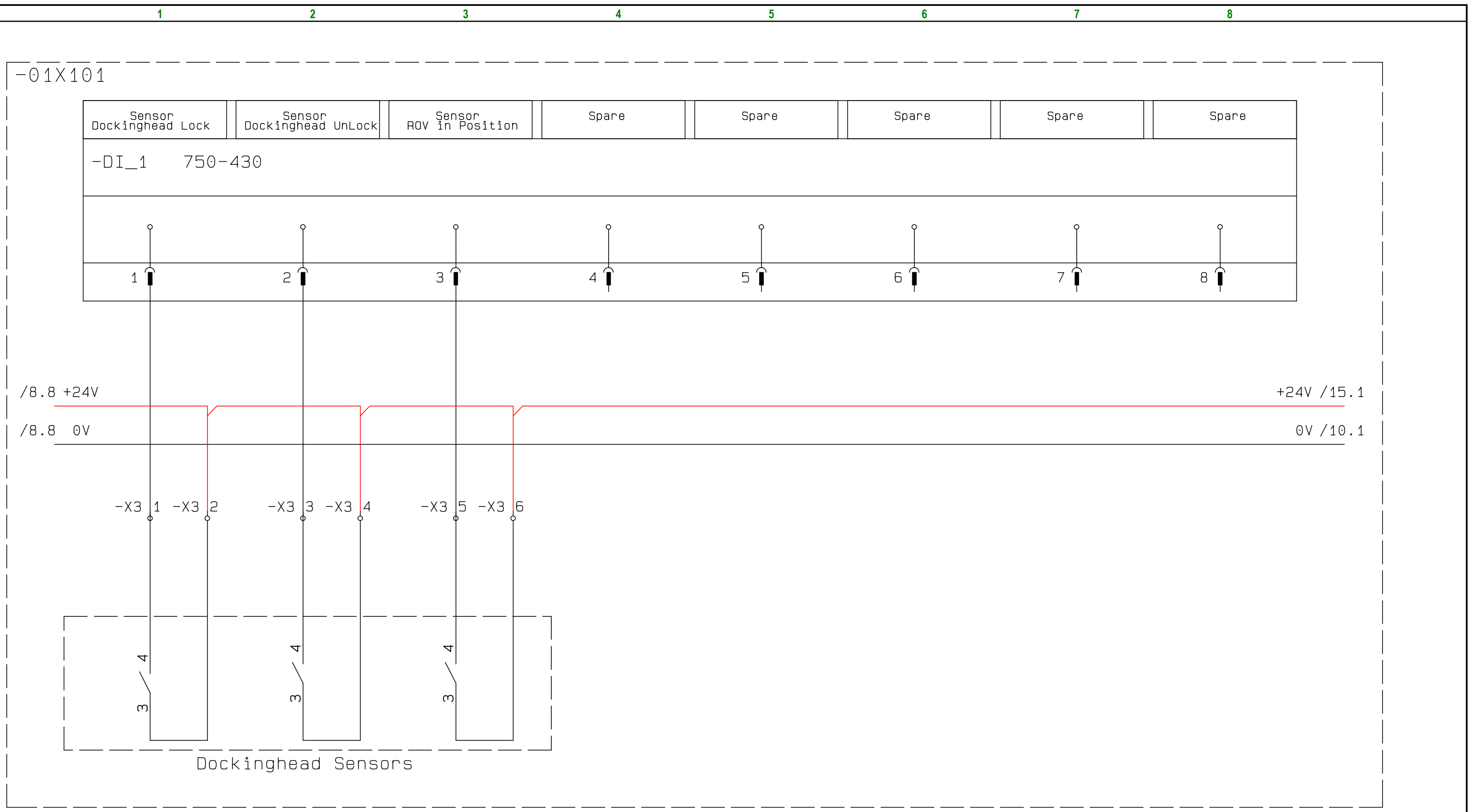
S1: MAIN SWITCH   T1: DC/DC CONVERTER

F0: FUSE 30A

Skoleversion

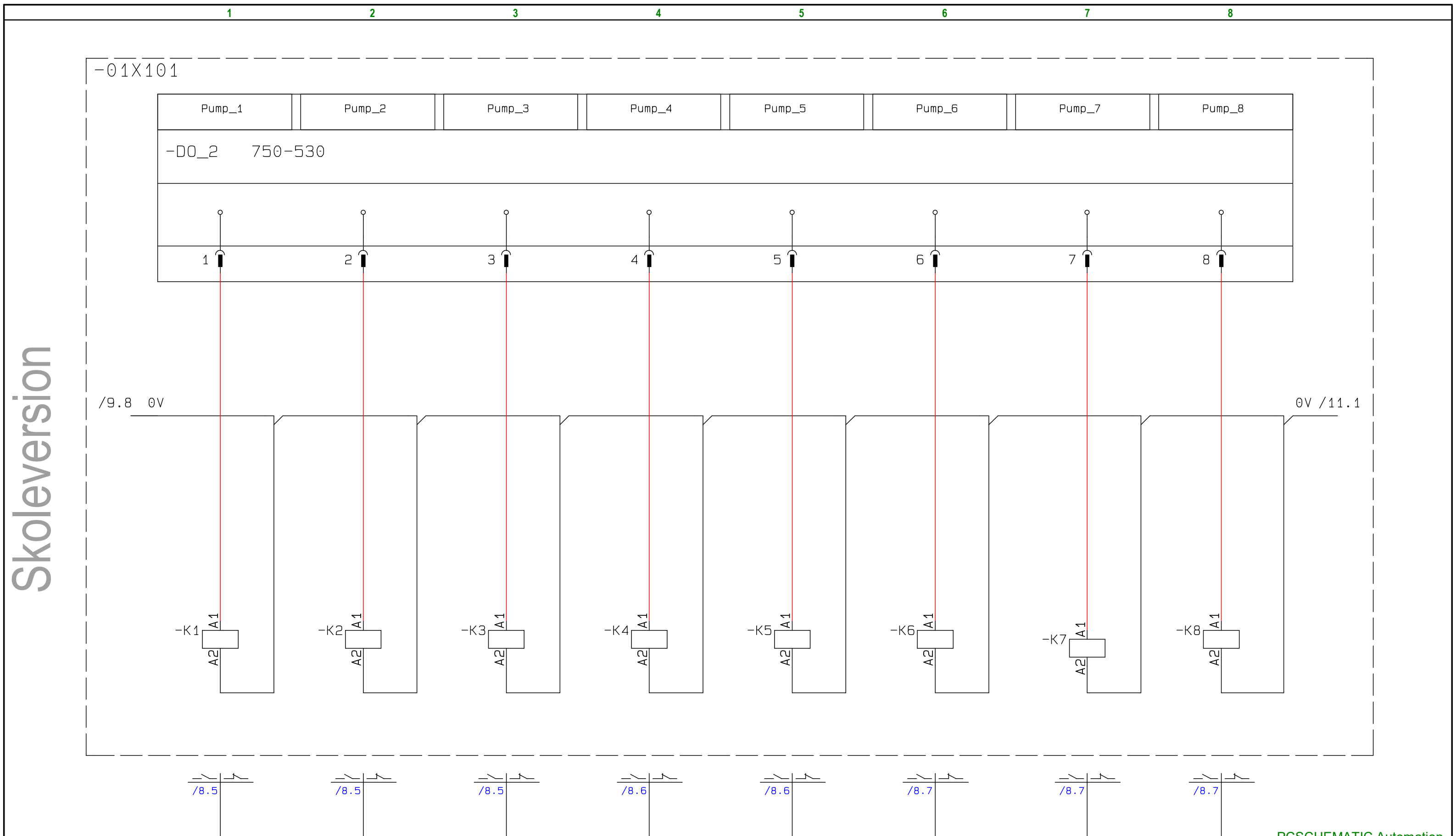
PCSCHEMATIC Automation

Project title:	**Integration of Aquaculture Inspection Platform**	**Project no.:**		**Project rev.:**		**Page**	**8**
Customer:	NTNU in Aalesund	DCC:				Scale:	1:1
Page title:	Control Cabinet	Dwg. no.:		Page rev.:		Previous page:	7
File name:	AIP_ElectroDrawings	Eng. (proj/page):		Last print:	19.05.2019	Next page:	9
Page ref.:		Appr. (date/init):		Last edit:	19.05.2019	Total no. of pages:	32



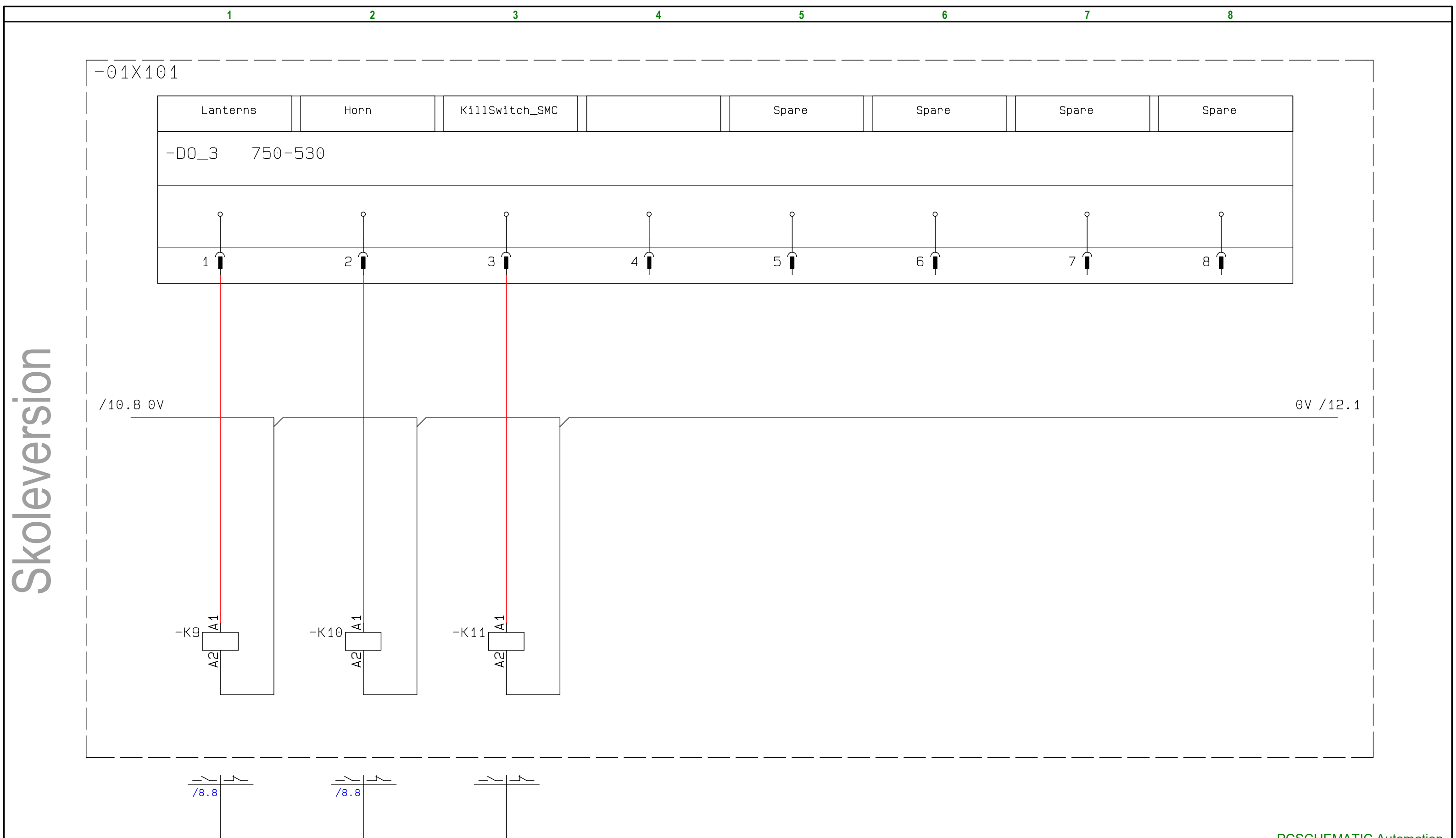
PCSCHEMATIC Automation

Project title:	**Integration of Aquaculture Inspection Platform**	**Project no.:**	**Project rev.:**		**Page**	**9**
Customer:	NTNU in Aalesund	DCC:			Scale:	1:1
Page title:	DI_1 Dockinghead Sensors	Dwg. no.:	Page rev.:		Previous page:	8
File name:	AIP_ElectroDrawings	Eng. (proj/page):	Last print:	19.05.2019	Next page:	10
Page ref.:		Appr. (date/init):	Last edit:	01.03.2019	Total no. of pages:	32



PCSCHEMATIC Automation

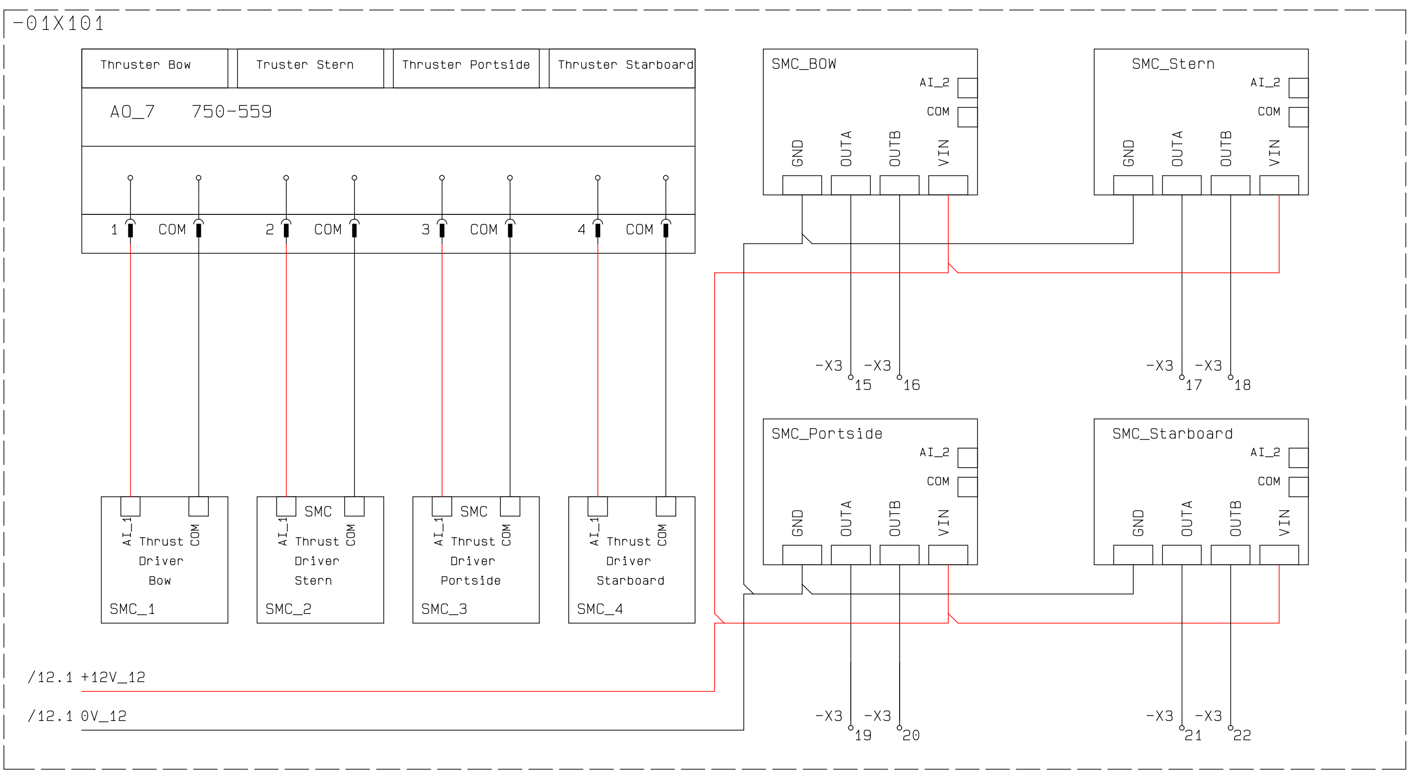
Project title:	**Integration of Aquaculture Inspection Platform**	**Project no.:**	**Project rev.:**		**Page**	**10**
Customer:	NTNU in Aalesund	DCC:			Scale:	1:1
Page title:	DO_2 Pumps	Dwg. no.:	Page rev.:		Previous page:	9
File name:	AIP_ElectroDrawings	Eng. (proj/page):	Last print:	19.05.2019	Next page:	11
Page ref.:		Appr. (date/init):	Last edit:	24.02.2019	Total no. of pages:	32



Project title:	**Integration of Aquaculture Inspection Platform**	**Project no.:**	**Project rev.:**			**Page**	**11**
Customer:	NTNU in Aalesund	DCC:				Scale:	1:1
Page title:	DO_3 Lanterns and Horn	Dwg. no.:	Page rev.:			Previous page:	10
File name:	AIP_ElectroDrawings	Eng. (proj/page):	Last print:	19.05.2019		Next page:	12
Page ref.:		Appr. (date/init):	Last edit:	14.05.2019		Total no. of pages:	32

PCSCHEMATIC Automation

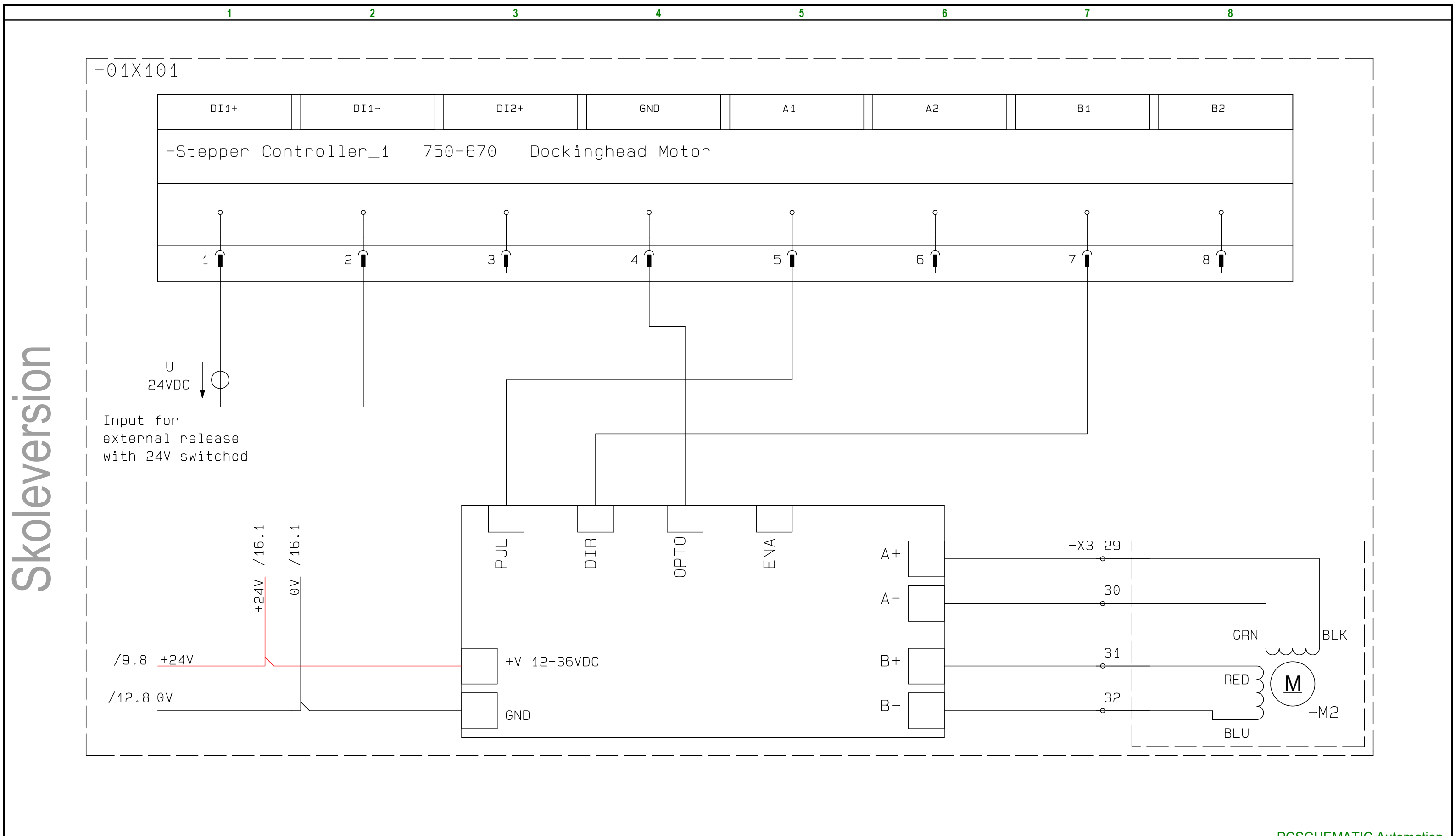
Projekttitel:	**Integration of Aquaculture Inspection Platform**	**Sagsnr.:**	**Projektrev.:**	**Side**	**12**
Kunde:		DCC:		Målestok:	1:1
Sidetitel:	AI_4 Pressure Sensors	Tegningsnr.:	Siderev.:	Forrige side:	11
Filnavn:	AIP_ElectroDrawings	Konstr. (projekt/side):	Sidst udskrevet: 19.05.2019	Næste side:	13
Sideref.:		Godk. (dato/init):	Sidst rettet: 19.05.2019	Antal sider ialt:	32



Skoleversion

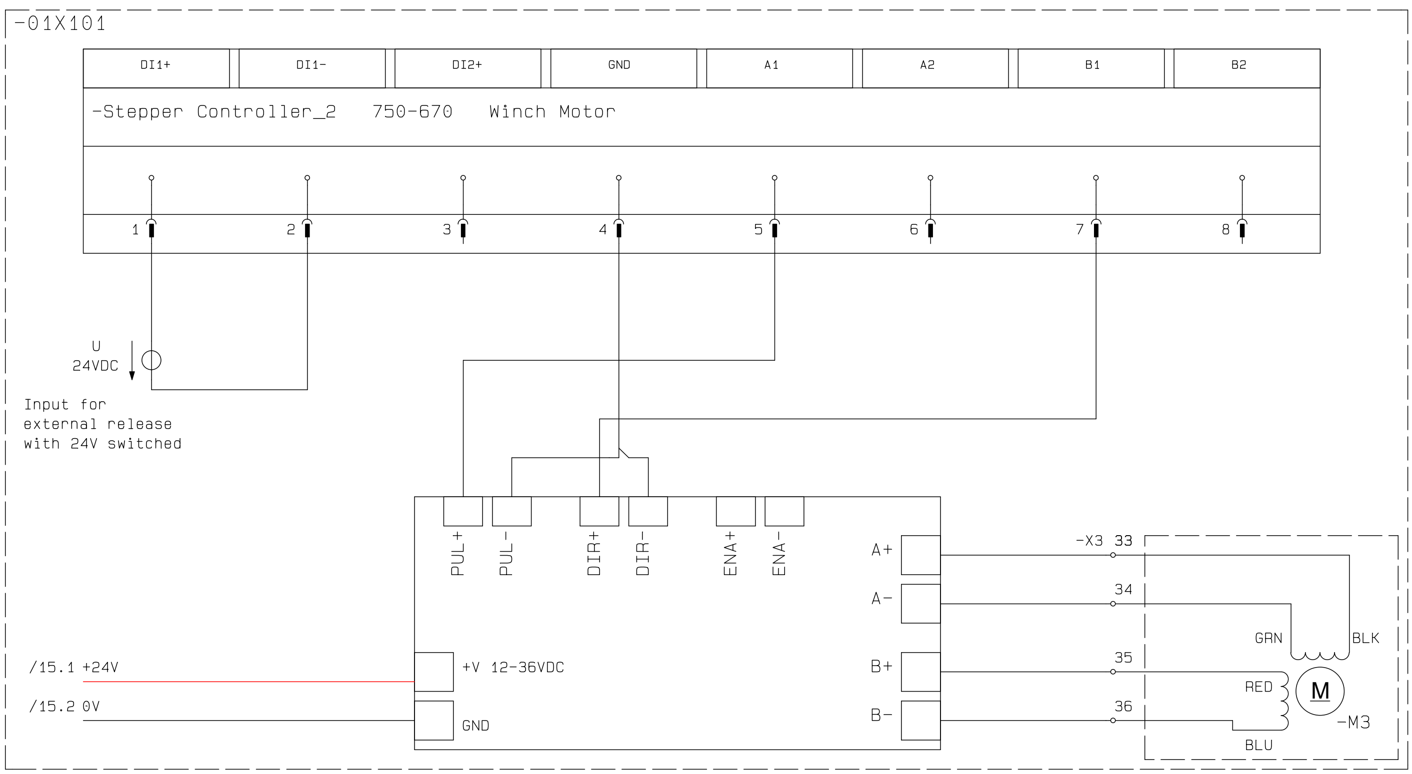
PCSCHEMATIC Automation

Projekttitel: **Integration of Aquaculture Inspection Platform**	**Sagsnr.:**	**Projektrev.:**	**Side**	**14**
Kunde:	DCC:		Målestok:	1:1
Sidetitel: AO_7 Thruster module and thruster controller	Tegningsnr.:	Siderev.:	Forrige side:	13
Filnavn: AIP_ElectroDrawings	Konstr. (projekt/side):	Sidst udskrevet: 19.05.2019	Næste side:	15
Sideref.:	Godk. (dato/init):	Sidst rettet: 19.05.2019	Antal sider ialt:	32



Projekttitel: **Integration of Aquaculture Inspection Platform**	**Sagsnr.:**	**Projektrev.:**	**Side** **15**
Kunde:	DCC:		Målestok: 1:1
Sidetitel: Stepper Controller_1 750-670 Dockinghead Motor	Tegningsnr.:	Siderev.:	Forrige side: 14
Filnavn: AIP_ElectroDrawings	Konstr. (projekt/side):	Sidst udskrevet: 19.05.2019	Næste side: 16
Sideref.:	Godk. (dato/init):	Sidst rettet: 01.03.2019	Antal sider ialt: 32

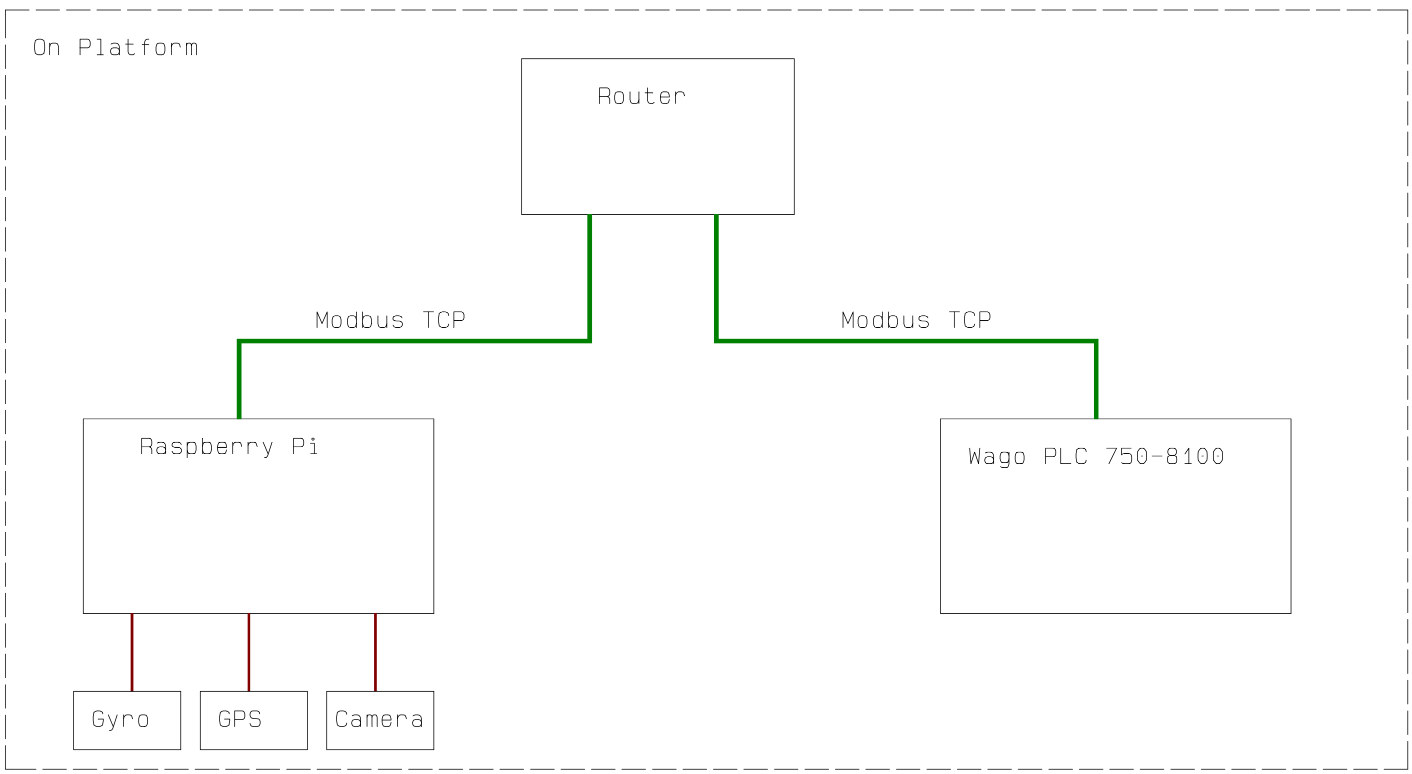
1	2	3	4	5	6	7	8



PCSCHEMATIC Automation

Projekttitel: **Integration of Aquaculture Inspection Platform**	**Sagsnr.:**	**Projektrev.:**	**Side** **16**
Kunde:	DCC:		Målestok: 1:1
Sidetitel: -Stepper Controller_2 750-670 Winch Motor	Tegningsnr.:	Siderev.:	Forrige side: 15
Filnavn: AIP_ElectroDrawings	Konstr. (projekt/side):	Sidst udskrevet: 19.05.2019	Næste side: 17
Sideref.:	Godk. (dato/init):	Sidst rettet: 01.03.2019	Antal sider ialt: 32

1 2 3 4 5 6 7 8



PCSCHEMATIC Automation

Projekttitel:	**Integration of Aquaculture Inspection Platform**	**Sagsnr.:**		**Projektrev.:**		**Side**	**17**
Kunde:		DCC:				Målestok:	1:1
Sidetitel:	Diagram	Tegningsnr.:		Siderev.:		Forrige side:	16
Filnavn:	AIP_ElectroDrawings	Konstr. (projekt/side):		Sidst udskrevet:	19.05.2019	Næste side:	7
Sideref.:		Godk. (dato/init):		Sidst rettet:	01.03.2019	Antal sider ialt:	32

K Progress reports

Lars Even Sætre, Ole Morken, Sigurd Olav Liavåg

Progress week 7 and 8.

Got all parts to the wago plc (except a 5V module). Also tested equipment on the platform to see if the plc can control them. Started some programming.

Communication between raspberry pi and plc is working. Using Modbus TCP.

Raspberry pi GPS and videostream up and running. With very good result.

Started on designing GUI. Using JavaFx to design the GUI.

Winch: 3D-printed all parts, but one of the parts is hard to print. Working on that to find a solution.

Made complete list of parts needed.

Future progress week 9 and 10

Raspberry pi on platform: Assemble IMU, GPS and Camerastream into one program.

Raspberry pi ROV: Replace with new Raspberry pi and test the new code.

Platform PLC: Program and map all variables. Map all Modbus TCP variables. Setup steppermotor control. Start on Stablilization code.

Finish electrical drawings.

Order all needed parts to the project early week 9.

GUI: First draft on front end design. After that start at backend design, and setup communication.

When parts received start mounting new parts and connect all.

Progress week 9 and 10

Winch: All partis inside winch finished. Assembled the winch and tested. The result was that the winch is working well. We made a new bracket to hold the winch on the engine side. This will be mounted during week 11, and a new bracket and cover to the winch stepper motor.

PLC: Stabilization done. Not tested in water yet, but have a simulator in e!cockpit and it seems to be working good. Also made a simulator to control the thrusters, this seems also to be working well.

Platform: Calculated on buoyancy. Made an excel sheet on the different values. We want to remove the pipes that was added during winch-project in 2017 because this will affect the performance on the platform. Not sure yet how the best solution is but have designed some solutions.

New brackets to hold the dockinghead have been made. The old ones were 3D printed in plastic and was broken. New one made in steel and aluminium.

DIMO AS sponsor us with a new cabinet that we need. This will arrive 07.03.2019.

Also ordered some small parts that wee needed with Anders. Got them week 10.

GUI: Good progress and is getting together. Looking very good and is user friendly.

Raspberry pi on the ROV is under testing. Some small issues to fix, but is in good progress

Electrical drawings are finished.

Future progress week 11 and 12

Bouyancy: Figure out the design and rebuild/add pipes on the platform to improve performance.

Install new Cabinet and wire the platform to complete installation.

Rebuild dockinghead with new brackets.

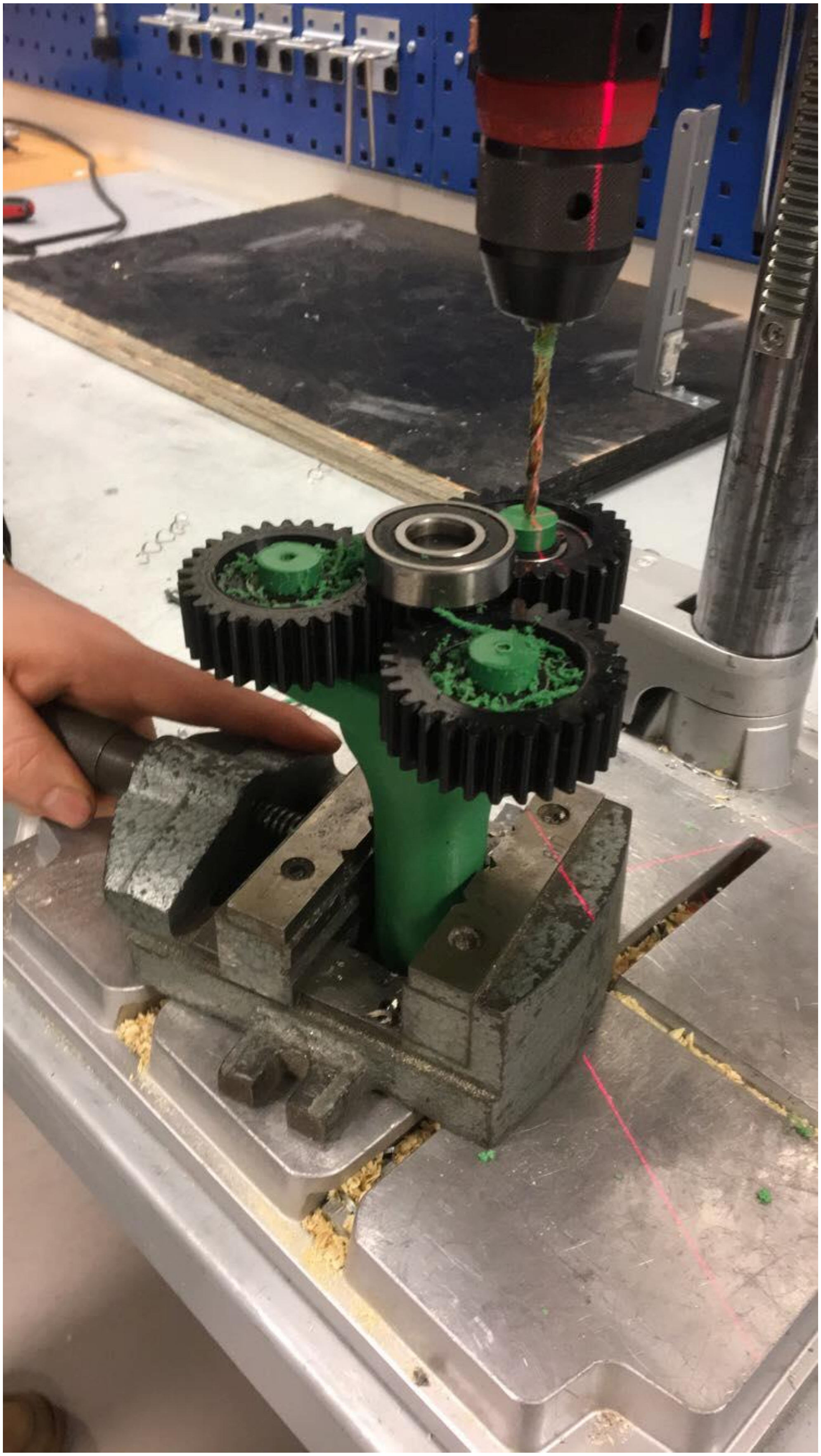
Rebuild ROV with new Raspberry pi and test.

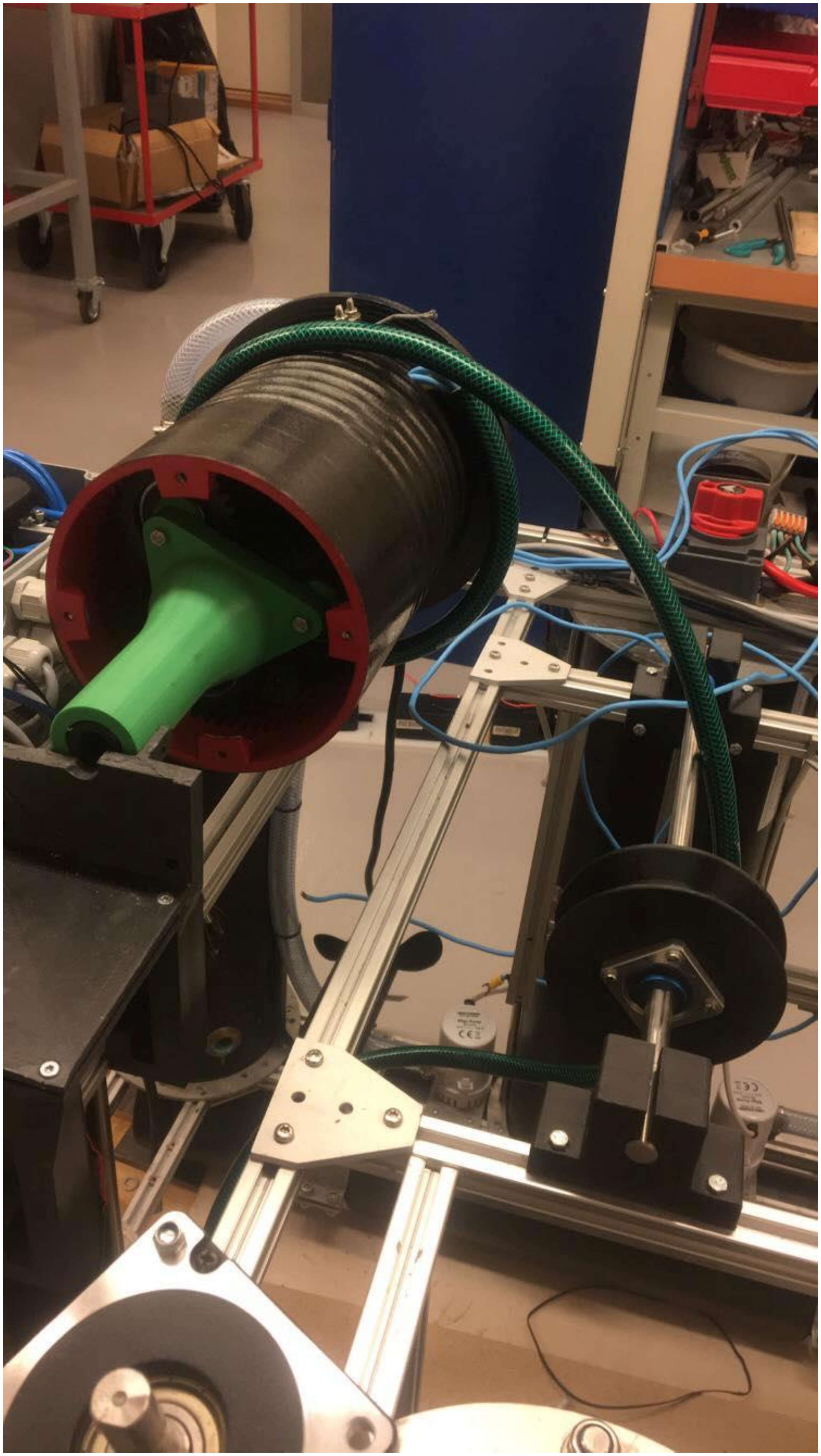
Winch: Mount new bracket and 3D-print new cover over the stepper motor.

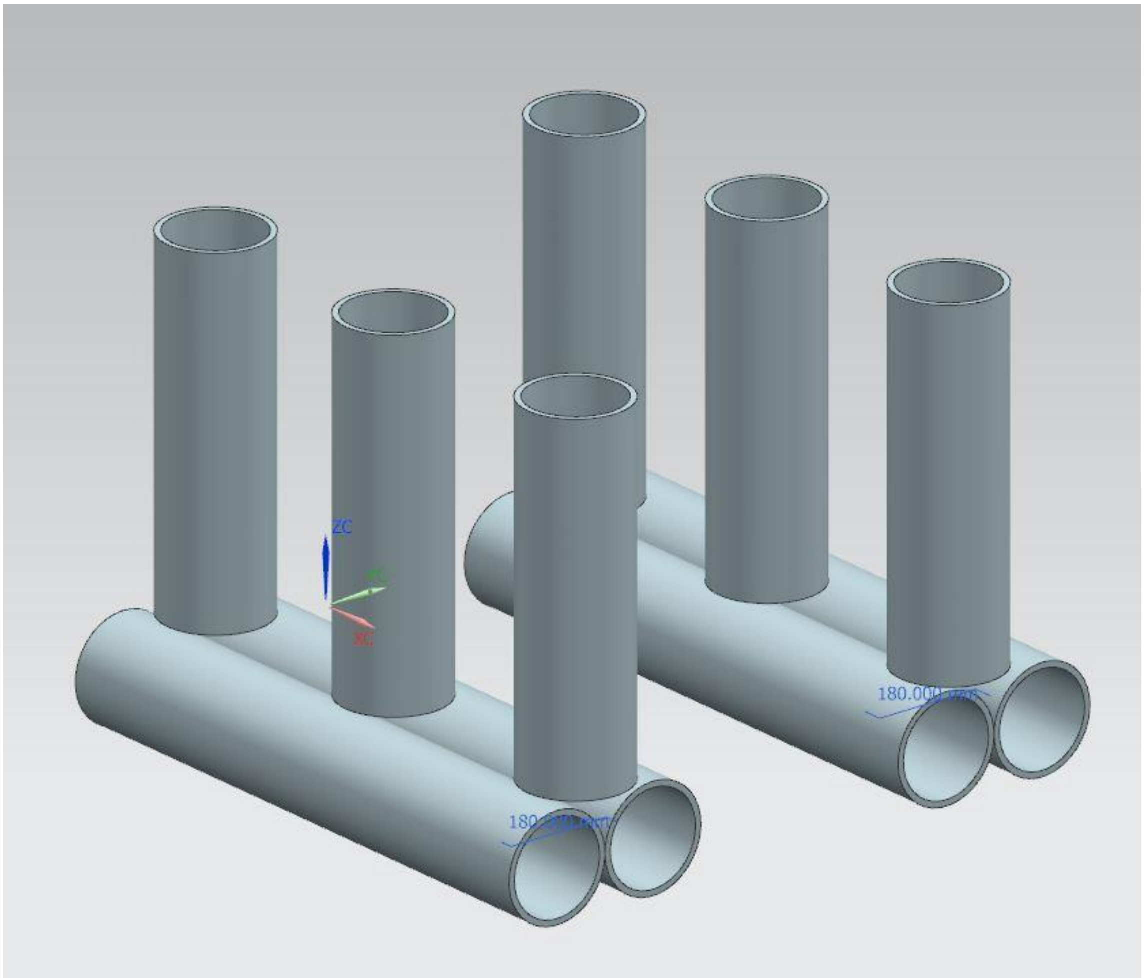
PLC: Start on Autopilot and after that DP.

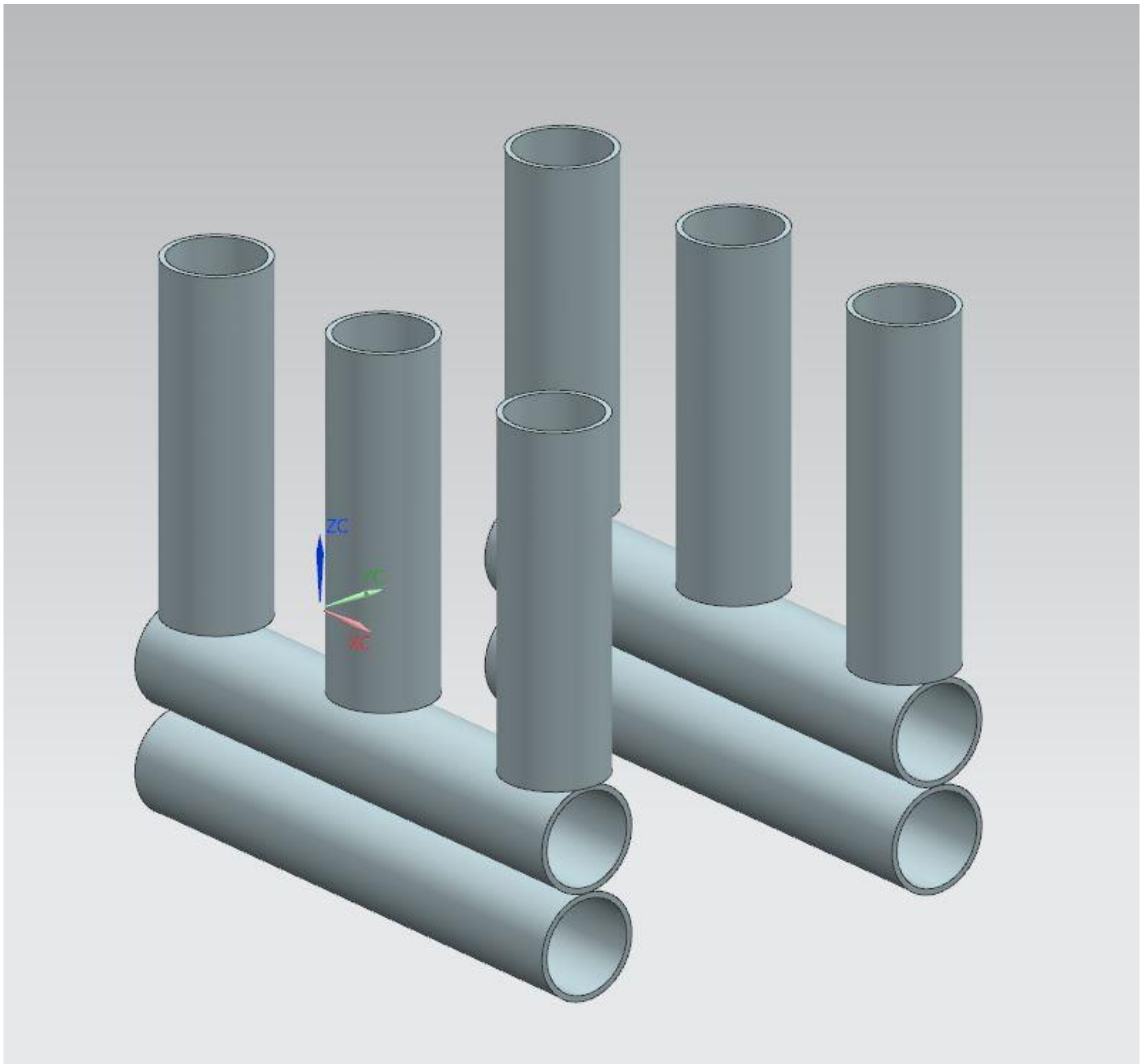
GUI: Backend programming. Setup communication.

Some pictures down below on what have been done:

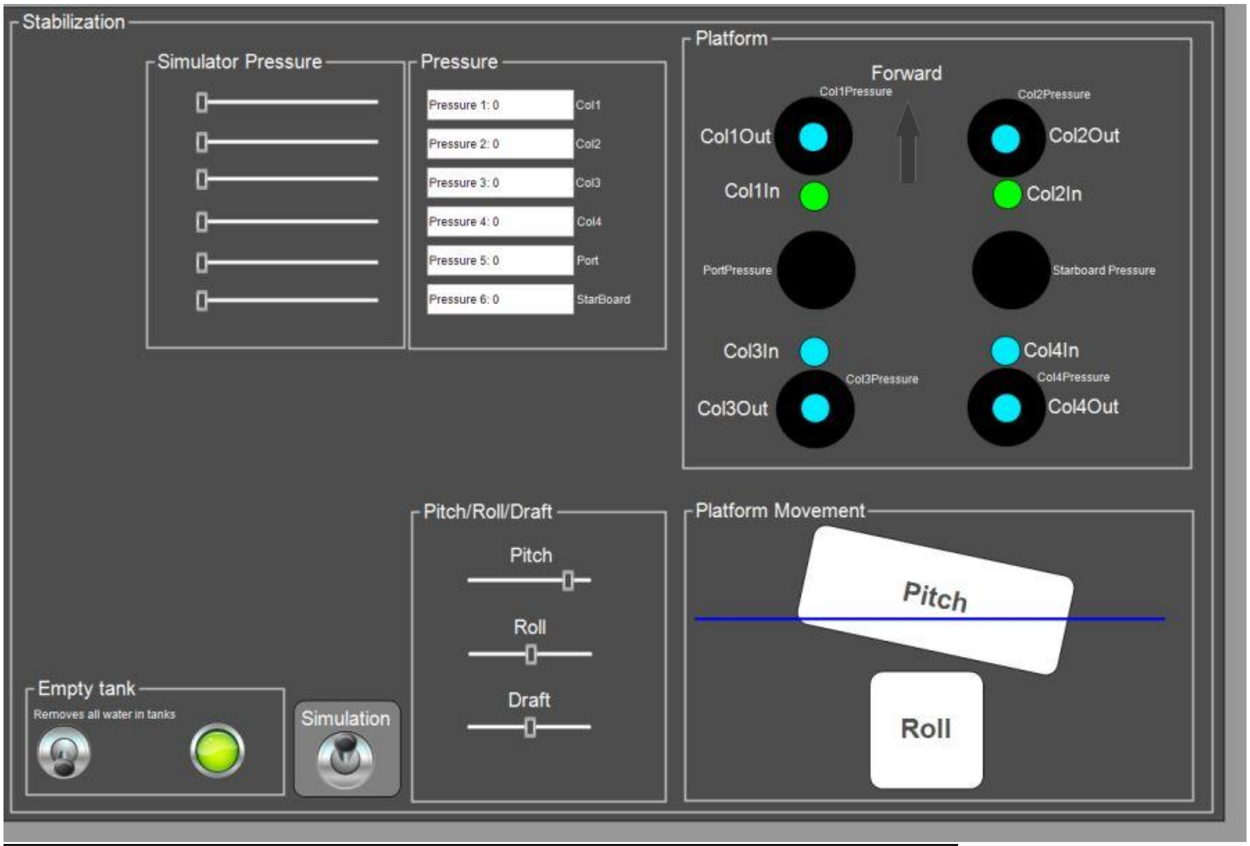


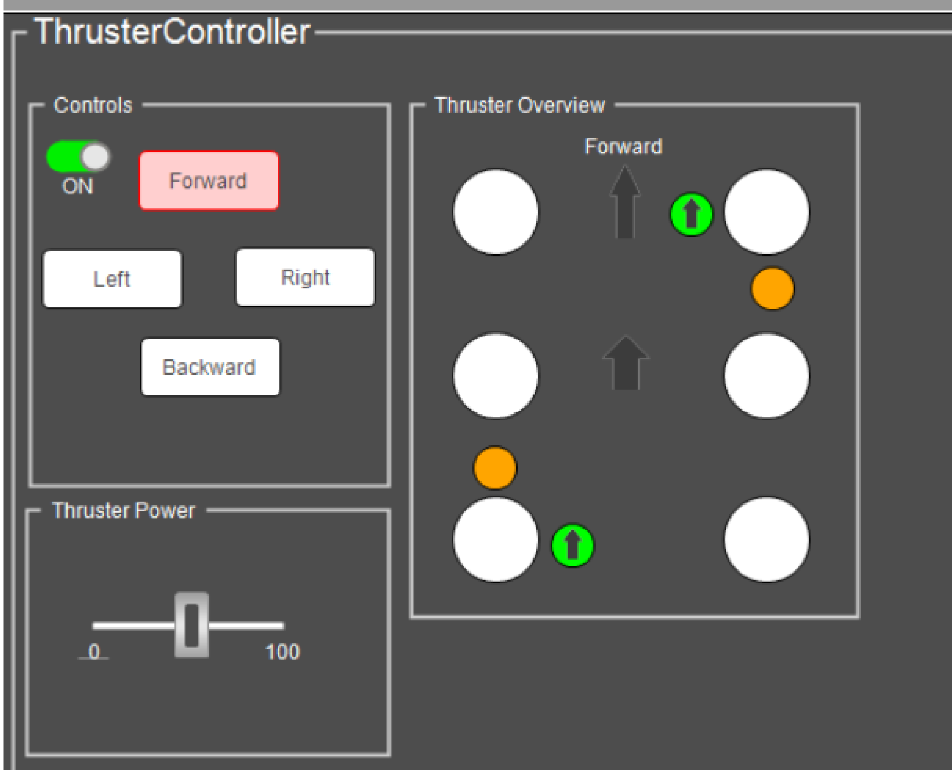



ZC
XC


ZC
YC
XC


608Z


Stabilization
Simulator Pressure
Pressure
Pressure 1: 0
Col1
Pressure 2: 0
Col2
Pressure 3: 0
Col3
Pressure 4: 0
Col4
Pressure 5: 0
Port
Pressure 6: 0
StarBoard
Platform
Forward
Col1Pressure
Col2Pressure
Col1Out
Col2Out
Col1In
Col2In
PortPressure
Starboard Pressure
Col3In
Col4In
Col3Pressure
Col4Pressure
Col3Out
Col4Out
Pitch/Roll/Draft
Pitch
Roll
Draft
Platform Movement
Pitch
Roll
Empty tank
Removes all water in tanks
Simulation


ThrusterController
Controls
ON
Forward
Left
Right
Backward
Thruster Overview
Forward
Thruster Power
0
100

Progress week 11 and 12

Bouyancy: Have calculated new buoyancy and got new pipes. The solution was to replace the two pipes vertical pipes. We will mount pipes that is 315mm in diameter. This will make sure that we will have good buoyancy. Then all the tanks can be filled up, and the platform will still float.

Cabinet delayed but expected delivery 22.03.2019. It was shipped on Wednesday 20.03.2019.

Dockinghead: Decided to make all brackets in aluminium. Brackets was done in week 12, and the dockinghead is now ready to be mounted back. This will be done 21.03.2019.

We have disconnected all old electrical wiring. The platform is now ready for new installation.

PLC: Some work done on autopilot. Can now calculate the angle to a point to the map. From platoform to point.

GUI: Frontend finished

ROV: New program coded and tested in week 12. Now the ROV is fully working.

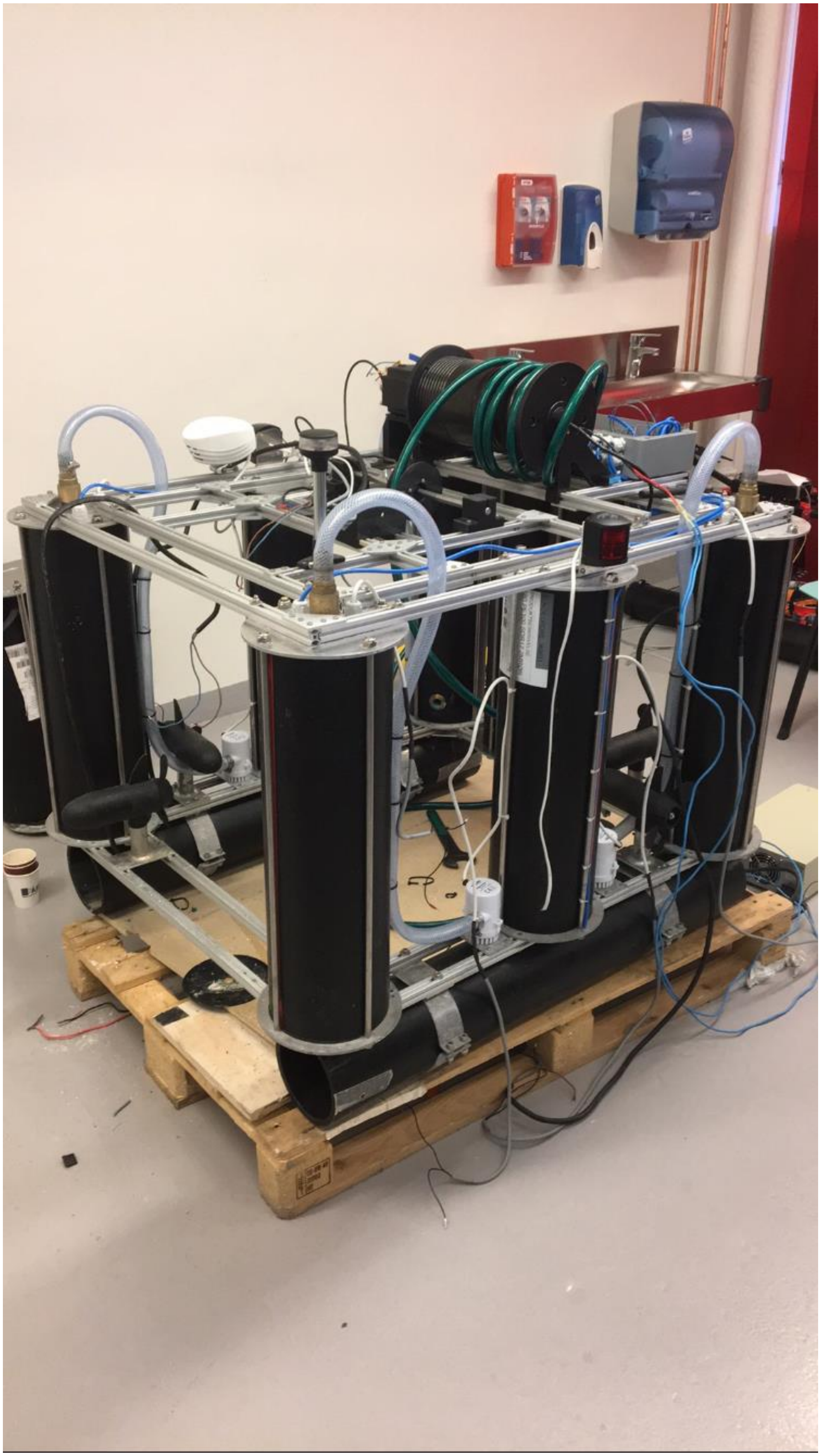
Future progress week 13 and 14

Mount new pipes.

Mount Cabinet and wire all electrical installation.

Surface: Backend programming

PLC: Program more on autopilot. Test communication between Surface and PLC.



Progress week 13 and 14

Cabinet mounted and electrical installation almost done.

Pipes cutted and new brackets are made.

Surface: google maps working and can get coordinates from waypoints.

ROV: Ready and fully working, raspberry pi replaced with a new one. With good result.

Future progress week 15 and 16

Program autopilot and get everything on the platform working (Equipment).

Surface: Communication and backend programming.

Map pressure sensors and mount sensors on dockinghead.

Mount winch steppermotor.

IE303612 Bacheloroppgave	Project Aquaculture Inspection Platform	Number of meeting this period 1). 0 planned	Firma - Oppdragsgiver NTNU In Aalesund	Side 1 av 2
Progress report	Period/week(s) Week 15, 16, 17 and 18	Number of hours this period. (from log) Approx.	Prosjektgruppe (navn) Lars Even Sætre, Ole Morken, Sigurd Olav Liavåg	Dato 01.05.19

Main goal/purpose for this periods work

- Platform: build the platform to a complete build. All equipment on the platform should work and communicate with each other.
- Surface: Finishing the GUI and setup the communication.
- Seatrail week 17/18 (end of April).

Planned activities this period

- Connect the remaining electrical installation.
- Test to control the platform and ROV from the GUI.
- Perform a seatrail and test the control system.
-

Actually conducted activites this period

- Electrical installation is done. Everything is working with no problems. Installed batteries on the platform.
- During testing of the GUI, the response and communication is working fast. Camerafeed to the GUI is working well.
- Went to seatrail week 17/18.

Description of/ justification for potential deviation between planned and real activities

- N/A

Description of/ justification for changes that is desired in the projects content or in the further plan of action – or progress report

- N/A

Main experience from this period

- During seatrail we had a problem with the platform stabilization. The new pipes under the plarform gives too much buoyancy force upwards, because of this the platform became very unstable. We added weight under the platform to make it more stable. This improved the stabilization, but it was still too unstable to be used without it hanging in the crane. We decided to make a floating ring around the whole platform om the top. This will prevent the platform to roll over. After this we could now test the control system.
- Stabilization system did not work as expected.
- IMU is unstable with the heading direction.
- Autopilot need more tuning but seems to be working well. Same with the DP.

Main purpose/focus next period

- Tuning of the control system parameters.
- Fix the stabilization system.
- Fix the IMU.
- Do a new seatrail at the end of week 19.
- During seatrail we will now store the data we get from the test.

Planned activities next period

- Week 19 and 20.
- Perform seatrail and test the system. Adjust all parameters so that the platform and ROV is working 100%.

1) Noter her kort tilbakemelding om antall møter – fordelt på typer (interne, styringsgruppe, møte med veileder) - i denne rapportperioden

IE303612 Bacheloroppgave	Project Aquaculture Inspection Platform	Number of meeting this period 1). 0 planned	Firma - Oppdragsgiver NTNU In Aalesund	Side 2 av 2
Progress report	Period/week(s) Week 15, 16, 17 and 18	Number of hours this period. (from log) Approx.	Prosjektgruppe (navn) Lars Even Sætre, Ole Morken, Sigurd Olav Liavåg	Dato 01.05.19

Other	
Wish/need for counceling - Nothing in particular	
Approval/signature group leader Lars Even Sætre	Signature other group participants Ole Morken, Sigurd Olav Liavåg.

1) Noter her kort tilbakemelding om antall møter – fordelt på typer (interne, styringsgruppe, møte med veileder) - i denne rapportperioden

L GUI source code

Main class GUI

```
/*
 * 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 seafarm;



//Java FX////
import javafx.beans.value.ChangeListener;
import javafx.beans.value.ObservableValue;
import com.sun.deploy.trace.TraceLevel;
import javafx.application.Application;
import javafx.geometry.HPos;
import javafx.geometry.VPos;
import javafx.scene.Scene;
import javafx.scene.control.Button;
import javafx.scene.control.Label;
import javafx.scene.layout.ColumnConstraints;
import javafx.scene.layout.GridPane;
import javafx.scene.layout.RowConstraints;
import javafx.stage.Stage;
import eu.hansolo.medusa.Gauge;
import eu.hansolo.medusa.GaugeBuilder;
import java.io.ByteArrayInputStream;
import java.text.DateFormat;
import java.text.SimpleDateFormat;
import java.util.Date;
import java.util.Observable;
import java.util.Observer;
import javafx.animation.AnimationTimer;
import javafx.application.Platform;
import javafx.geometry.Insets;
import javafx.scene.canvas.Canvas;
import javafx.scene.canvas.GraphicsContext;
import javafx.scene.control.Slider;
import javafx.scene.control.Tab;
import javafx.scene.control.TabPane;
import javafx.scene.control.TextField;
import javafx.scene.image.Image;
import javafx.scene.layout.BorderPane;
import javafx.scene.layout.HBox;
```

```java
import javafx.scene.layout.VBox;
import javafx.scene.paint.Color;
import javafx.scene.shape.Circle;
//Open CV//
import org.opencv.core.Core;
import org.opencv.core.Mat;
import org.opencv.core.MatOfByte;
import org.opencv.core.Size;
import org.opencv.imgcodecs.Imgcodecs;
import org.opencv.imgproc.Imgproc;
import org.opencv.videoio.VideoCapture;
import org.opencv.videoio.VideoWriter;
import de.re.easymodbus.modbusclient.ModbusClient;
/**
 *
 * @author Ole Morken
 */
public class SeaFarm extends Application implements Observer {

    //VideoStreamReciever
    //udpVideoReciever vidUdpOverview;
    udpVideoReciever vidUdpPlatform;
    udpVideoReciever vidUdpROV;
    maps maps;

    // MODBUS comunication///
    ModbusClient modbusClientRead;
    ModbusClient modbusClientSend;
    static final String MODBUSIPADRESS = "192.168.0.112";
    //recieve modbus
    ModbusReciever recieverModbus;
    RecieveDataObserver recieveObserver;
    //send modbus
    ModbusSender sendModbus;
    SendDataObserver sendObserver;


    //ROV variables///
    static final String ROVIPADDRESS = "192.168.0.100";
    static final int ROVRECEIVEPORT = 8765;
    static final int ROVSENDPORT = 9876;
    static final int ROVVIDEOPORT = 12342;
    static RovSendEventState enumStateEvent = RovSendEventState.FALSE;
    RovGUIController rovGuicontroller;
```

```
RovDatahandler rovDatahandler;
RovUDPreceiver rovUdpReceiver;
RovReceiveDataObservable rovReceiveDataObserver;



//platform video
static final int PLATVIDEOPORT = 12345;
autoMode autMode;









private static final int SCENE_W = 640;
private static final int SCENE_H = 480;

boolean stoprecPlatform;
boolean startRecPlatform;
boolean stoprecROV;
boolean startRecROV;
//VideoCapture videoCapture;
//Canvas canvas;

GraphicsContext g2dPlatform;
GraphicsContext g2dROV;
Stage window;

AnimationTimer timerPlatform;
AnimationTimer timerROV;
VideoWriter writerPlatform;
VideoWriter writerROV;

Gauge depthMeter;
Gauge temp;
double autoModeLatSetpoint;
double autoModeLongSetpoint;
float longitude;
float latitude;
float currentSpeed=0;
float heading=0;
```

```
float pitch=0;
float roll=0;
boolean enabled=false;
//winch variables//
boolean rovLocked=false;
boolean rovUpperPos=false;
boolean dpModeActivated = false;
boolean autoModeActivated = false;
boolean manualModeActivated = false;
int platformSpeedVal=0;

//mode indicators platform

Circle recCirclePlatform ;
Circle dpCirclePlatform ;
Circle autoCirclePlatform ;
Circle manualCirlcePlatform ;
Circle lockCircleROV ;
Circle upperCircleROV;

//textFields for sensordata

//text fields
   TextField headingPlatTextField ;
   TextField curretnSpeenTextField ;
   TextField pitchTextField ;
   TextField rollTextField;

   //rov
   TextField deptRovTextField;
   TextField tempRovTextField;
   TextField headingRovTextField;
   TextField tempInWaterTextField;
   TextField oxygenInWaterTextField;

@Override
public void start(Stage primaryStage) {
   window = primaryStage;
   initOpenCv();

   depthMeter = makeGauge("Depth", "Meter");
   temp = makeGauge("Temperature", "C");
   temp.setSubTitle("ROV");
   depthMeter.setSubTitle("ROV");
```

```
        temp.setValue(0);

         //modbus threads
        modbusClientRead= new ModbusClient(MODBUSIPADRESS, 502);
        modbusClientSend= new ModbusClient(MODBUSIPADRESS, 502);
        recieveObserver= new RecieveDataObserver();
        recieveObserver.addObserver(this);
        recieverModbus = new ModbusReciever(modbusClientRead
,recieveObserver);
        recieverModbus.start();

        //sender modbus
        sendModbus=new ModbusSender(modbusClientSend);
        sendObserver = new SendDataObserver();
        sendObserver.addObserver(sendModbus);
        sendModbus.start();

        //ROV ///
        rovDatahandler=new RovDatahandler();
        rovDatahandler.setThreadStatus(true);
        rovGuicontroller=new RovGUIController();
        rovGuicontroller.setDatahandler(rovDatahandler);
        rovGuicontroller.setStart(true);

        // create the object  of receiving data to GUI
        rovReceiveDataObserver = new RovReceiveDataObservable();
        // create the UDP thread, puts data into subject
        rovUdpReceiver = new RovUDPreceiver(rovReceiveDataObserver,
ROVRECEIVEPORT);
        rovUdpReceiver.start();
        rovReceiveDataObserver.addObserver(this);

        //creates a map object
        maps = new maps();
        /// UDP VideoStream
        vidUdpPlatform = new udpVideoReciever(PLATVIDEOPORT);
        vidUdpPlatform.start();
        vidUdpROV = new udpVideoReciever(ROVVIDEOPORT);
        vidUdpROV.start();
        DateFormat dateFormat = new SimpleDateFormat("yyyy_MM_dd_HH_mm_ss");

        //making automode object...//
        autMode=new autoMode();
```

```
        //vidUdp.start();
        // TAB PANE
        TabPane tabPane = new TabPane();
        tabPane.setTabClosingPolicy(TabPane.TabClosingPolicy.UNAVAILABLE);
        Tab overviewTab = new Tab("OVERVIEW");
        Tab platformTab = new Tab("PLATFORM");
        Tab rovTab = new Tab("ROV");
        Tab settingsTab = new Tab("SETTINGS");

        ///////// Overview



        // Buttons for GPS data
        Button closebtn = new Button("Close");
        closebtn.setOnAction(e -> closeProgram());
        Button getLat = new Button("Get GPS position");
        getLat.setOnAction(e -> System.out.println("Latitude: " +
maps.getNextPosLat() + " Langditude: " + maps.getNextPosLng()));


        // Labels
        Label sensorData = new Label("SENSOR DATA");
        Label rov = new Label("ROV");
        Label platform = new Label("PLATFORM");
        Label depthRov = new Label("Depth ROV: ");
        Label rovTemp = new Label("Temp in ROV:");
        Label headingRov = new Label("Heading ROV:");
        Label headingPlat = new Label("Heading Platform:");
        Label currenSpeedPlat = new Label("Speed Platform:");
        Label pitchPlat = new Label("Pitch Platform:");
        Label rollPlat = new Label("ROLL Platform:");
        Label oxygenLabel = new Label("Oxygen in water:");
        Label waterTempLabel = new Label("Temp in water:");
        //text fields in overview tab
         headingPlatTextField = new TextField(Float.toString(heading));
         curretnSpeenTextField = new
TextField(Float.toString(currentSpeed));
         pitchTextField = new TextField(Float.toString(pitch));
         rollTextField = new TextField(Float.toString(roll));
         deptRovTextField = new TextField(Float.toString(0));
         tempRovTextField = new TextField(Float.toString(0));
         headingRovTextField = new TextField(Float.toString(0));
```

```
        tempInWaterTextField = new TextField(Float.toString(0));
        oxygenInWaterTextField = new TextField(Float.toString(0));




       // Panes in overview tab
        GridPane gridpaneOverview = new GridPane();
        GridPane gridpaneValues = new GridPane();
        BorderPane mapPane = new BorderPane();
        mapPane.setCenter(maps.getMapView());
        mapPane.setPrefSize(640, 480);

        //Canvas overviewCanvas = new Canvas(SCENE_W, SCENE_H);
        // Overall constraints
        gridpaneOverview.getColumnConstraints().add(new
ColumnConstraints(300));
        gridpaneOverview.getColumnConstraints().add(new
ColumnConstraints(300));
        gridpaneOverview.getColumnConstraints().add(new
ColumnConstraints(300));
        gridpaneOverview.getColumnConstraints().add(new
ColumnConstraints(300));
        gridpaneOverview.getRowConstraints().add(new RowConstraints(50));

        // Constraints Buttons
        GridPane.setConstraints(closebtn, 0, 4);
        GridPane.setValignment(closebtn, VPos.BOTTOM);
        GridPane.setHalignment(closebtn, HPos.LEFT);
        GridPane.setConstraints(getLat, 2, 0, 1, 1, HPos.LEFT,
VPos.CENTER);

        // Constraints Lables
        GridPane.setConstraints(sensorData, 0, 0);
        GridPane.setHalignment(sensorData, HPos.CENTER);
        GridPane.setConstraints(depthMeter, 3, 4);
        GridPane.setConstraints(temp, 2, 4);

        // Panes inside GridPane
        //GridPane.setConstraints(overviewCanvas, 2, 1);
        //GridPane.setHalignment(overviewCanvas, HPos.RIGHT);
       GridPane.setConstraints(mapPane, 2, 1, 2, 1, HPos.RIGHT,
VPos.CENTER);
       GridPane.setConstraints(gridpaneValues, 0, 1, 2, 2, HPos.CENTER,
```

```
VPos.CENTER);
       gridpaneValues.getColumnConstraints().add(new
ColumnConstraints(300));
       gridpaneValues.getColumnConstraints().add(new
ColumnConstraints(300));
       gridpaneValues.getRowConstraints().add(new RowConstraints(50));
       gridpaneValues.getRowConstraints().add(new RowConstraints(50));
       gridpaneValues.getRowConstraints().add(new RowConstraints(50));
       gridpaneValues.getRowConstraints().add(new RowConstraints(50));
       gridpaneValues.getRowConstraints().add(new RowConstraints(50));
       gridpaneValues.getRowConstraints().add(new RowConstraints(50));
       gridpaneValues.getRowConstraints().add(new RowConstraints(50));

       GridPane.setConstraints(depthRov, 0, 0);
       GridPane.setConstraints(rovTemp, 0, 1);
       GridPane.setConstraints(headingRov, 0, 2);
       GridPane.setConstraints(headingPlat, 0, 3);
       GridPane.setConstraints(currenSpeedPlat, 0, 4);
       GridPane.setConstraints(pitchPlat, 0, 5);
       GridPane.setConstraints(rollPlat, 0, 6);
       GridPane.setConstraints(oxygenLabel, 0, 7);
       GridPane.setConstraints(waterTempLabel, 0, 8);

       GridPane.setConstraints(deptRovTextField, 1, 0);
       GridPane.setConstraints(tempRovTextField, 1, 1);
       GridPane.setConstraints(headingRovTextField, 1, 2);
       GridPane.setConstraints(headingPlatTextField, 1, 3);
       GridPane.setConstraints(curretnSpeenTextField, 1, 4);
       GridPane.setConstraints(pitchTextField, 1, 5);
       GridPane.setConstraints(rollTextField, 1, 6);
       GridPane.setConstraints(oxygenInWaterTextField, 1, 7);
       GridPane.setConstraints(tempInWaterTextField, 1, 8);

        // Videostream


        // Add to the differet panes
        gridpaneValues.getChildren().addAll(depthRov, rovTemp, headingRov,
                headingPlat, currenSpeedPlat, pitchPlat,
rollPlat,oxygenLabel,waterTempLabel,deptRovTextField,tempRovTextField,headi
ngRovTextField, headingPlatTextField,
                curretnSpeenTextField, pitchTextField,
rollTextField,oxygenInWaterTextField,tempInWaterTextField);
        //mapPane.getChildren().addAll(maps.getMapView());
```

```
        //valueBox.getChildren().addAll(headingPlatValue);
        //dataBox.getChildren().addAll(rov, depthRov, rovTemp, headingRov,
platform,
                //headingPlat, currenSpeedPlat, yawPlat, rollPlat);
        gridpaneOverview.getChildren().addAll(closebtn, sensorData,
gridpaneValues, depthMeter,
        mapPane, temp, getLat);

        ///////////// Platform TAB

         recCirclePlatform = new Circle(10, Color.TRANSPARENT);
         dpCirclePlatform = new Circle(10, Color.RED);
         autoCirclePlatform = new Circle(10, Color.RED);
         manualCirlcePlatform = new Circle(10, Color.RED);

       Color redDotPlatform = Color.RED;
       Color orangeDotPlatform = Color.ORANGE;
       Color notRecPlatform = Color.TRANSPARENT;




        // Buttons
        Button fwdButtonPlatform = new Button("FWD");
        fwdButtonPlatform.setMinSize(100, 75);
        fwdButtonPlatform.setOnTouchPressed(e -> {
                                   sendObserver.setFwdMotion(true);
                                   sendObserver.notifyObs();});
        fwdButtonPlatform.setOnTouchReleased(e -> {
                                   sendObserver.setFwdMotion(false)
;
                                   sendObserver.notifyObs();});

        Button bckButtonPlatform = new Button("AFT");
        bckButtonPlatform.setMinSize(100, 75);
        bckButtonPlatform.setOnTouchPressed(e -> {
                                   sendObserver.setBckMotion(true);
                                   sendObserver.notifyObs();});
        bckButtonPlatform.setOnTouchReleased(e -> {
                                   sendObserver.setBckMotion(false)
;
                                   sendObserver.notifyObs();});
        Button lftButtonPlatform = new Button("LEFT");
        lftButtonPlatform.setMinSize(125, 75);
```

```
        lftButtonPlatform.setOnTouchPressed(e -> {
                                   sendObserver.setLeftMotion(true)
;
                                   sendObserver.notifyObs();});
        lftButtonPlatform.setOnTouchReleased(e -> {
                                   sendObserver.setLeftMotion(false
);
                                   sendObserver.notifyObs();});
        Button rgtButtonPlatform = new Button("RIGHT");
        rgtButtonPlatform.setMinSize(125, 75);
        rgtButtonPlatform.setOnTouchPressed(e -> {
                                   sendObserver.setRightMotion(true
);
                                   sendObserver.notifyObs();});
        rgtButtonPlatform.setOnTouchReleased(e -> {
                                   sendObserver.setRightMotion(fals
e);
                                   sendObserver.notifyObs();});

        Button clwButtonPlatform = new Button("CLW");
        clwButtonPlatform.setMinSize(100, 50);
        clwButtonPlatform.setOnTouchPressed(e -> {
                                   sendObserver.setClockWMotion(tru
e);
                                   sendObserver.notifyObs();});
        clwButtonPlatform.setOnTouchReleased(e -> {
                                   sendObserver.setClockWMotion(fal
se);
                                   sendObserver.notifyObs();});

        Button cClwButtonPlatform = new Button("CCLW");
        cClwButtonPlatform.setMinSize(100, 50);
        cClwButtonPlatform.setOnTouchPressed(e -> {
                                   sendObserver.setCounterClockWMot
ion(true);
                                   sendObserver.notifyObs();});
        cClwButtonPlatform.setOnTouchReleased(e -> {
                                   sendObserver.setCounterClockWMot
ion(false);
                                   sendObserver.notifyObs();});
        Button startPump = new Button("Empety tanks");
        startPump.setOnTouchPressed(e -> {
                                  sendObserver.setStartPump(true);
                                  sendObserver.notifyObs(); });
```

```
        startPump.setOnTouchReleased(e -> {
                                          sendObserver.setStartPump(false)
;
                                          sendObserver.notifyObs();});
            Button flute = new Button("Flute");
        flute.setOnTouchPressed(e -> {
                                         sendObserver.setEnableFlute(true)
;
                                         sendObserver.notifyObs(); });
        flute.setOnTouchReleased(e -> {
                                          sendObserver.setEnableFlute(fals
e);
                                          sendObserver.notifyObs();});

        Button dpModeButton = new Button("DP-Mode");
        dpModeButton.setMaxWidth(160);
        dpModeButton.setOnTouchPressed(e -> {
            autoModeLatSetpoint=latitude;
            autoModeLongSetpoint=longitude;
            sendObserver.setDpModeEnable(true);
            sendObserver.setEnableAuto(false);
            sendObserver.setEnableManual(false);
            sendObserver.notifyObs();
            dpCirclePlatform.setFill(orangeDotPlatform);
            autoCirclePlatform.setFill(redDotPlatform);
            manualCirlcePlatform.setFill(redDotPlatform);
                });

        Button autopilotModeButton = new Button("Autopilot");
        autopilotModeButton.setMaxWidth(160);
        autopilotModeButton.setOnTouchPressed(e -> {
            autoModeLatSetpoint=latitude;
            autoModeLongSetpoint=longitude;
            sendObserver.setDpModeEnable(false);
            sendObserver.setEnableAuto(true);
            sendObserver.setEnableManual(false);
            sendObserver.notifyObs();
            dpCirclePlatform.setFill(redDotPlatform);
            autoCirclePlatform.setFill(orangeDotPlatform);
            manualCirlcePlatform.setFill(redDotPlatform);
                });

        Button manualModeButton = new Button("Manual Mode");
        manualModeButton.setMaxWidth(160);
```

```
        manualModeButton.setOnTouchPressed(e -> {
            sendObserver.setDpModeEnable(false);
            sendObserver.setEnableAuto(false);
            sendObserver.setEnableManual(true);
            sendObserver.notifyObs();
            dpCirclePlatform.setFill(redDotPlatform);
            autoCirclePlatform.setFill(redDotPlatform);
            manualCirlcePlatform.setFill(orangeDotPlatform);
                });


       Button startRecordBtnPlatform = new Button("Record");
       startRecordBtnPlatform.setOnAction(e -> {
           writerPlatform = new
VideoWriter("C:\\Users\\Platform\\Desktop\\ROVVideo\\Platform\\"
                   +dateFormat.format(new Date())+".avi",
VideoWriter.fourcc
                    ('M', 'J', 'P', 'G'), 15,  new Size(320, 240));
           startRecPlatform = true;
           recCirclePlatform.setFill(redDotPlatform);
       });
       startRecordBtnPlatform.setMaxSize(150, 20);

       Button stopRecordBtnPlatform = new Button("Stop");
       stopRecordBtnPlatform.setMaxSize(150, 20);
       stopRecordBtnPlatform.setOnMousePressed(e -> {
           stoprecPlatform = true;
           startRecordBtnPlatform.disarm();
           startRecPlatform = false;
           recCirclePlatform.setFill(notRecPlatform);
       });
       stopRecordBtnPlatform.setOnMouseReleased(e -> stoprecPlatform =
false);

        // Slider
        Slider speedPlatform = new Slider(0, 100, 50);

        speedPlatform.setShowTickMarks(true);
        speedPlatform.setShowTickLabels(true);
        speedPlatform.setMajorTickUnit(25f);
        speedPlatform.setBlockIncrement(10f);
        Label platformLabel = new Label("Platform Speed");
```

```
        speedPlatform.valueProperty().addListener(new
ChangeListener<Number>(){
        @Override
        public void changed(ObservableValue<? extends Number> observable,
                Number oldValue, Number newValue){
            System.out.println("New Value: " + newValue);

            sendObserver.setThrusterSpeed(newValue.intValue());
            sendObserver.notifyObs();
        }
        });


        // Panes
        GridPane platformLayout = new GridPane();
        GridPane platformGrid = new GridPane();
        Canvas platformCanvas = new Canvas(SCENE_W, SCENE_H);


        // Constraints
        platformLayout.getColumnConstraints().add(new
ColumnConstraints(300));
        platformLayout.getColumnConstraints().add(new
ColumnConstraints(250));
        platformLayout.getColumnConstraints().add(new
ColumnConstraints(300));
        platformLayout.getColumnConstraints().add(new
ColumnConstraints(350));
        platformLayout.getRowConstraints().add(new RowConstraints(50));
        platformLayout.getRowConstraints().add(new
RowConstraints(SCENE_H));
        platformLayout.getRowConstraints().add(new RowConstraints(150));


        GridPane.setConstraints(platformCanvas, 0, 1);
        GridPane.setHalignment(platformCanvas, HPos.LEFT);
        GridPane.setConstraints(fwdButtonPlatform, 3, 2, 1, 1, HPos.CENTER,
VPos.TOP);
        GridPane.setConstraints(bckButtonPlatform, 3, 2, 1, 1, HPos.CENTER,
VPos.BOTTOM);
        GridPane.setConstraints(lftButtonPlatform, 3, 2, 1, 1, HPos.LEFT,
VPos.BOTTOM);
        GridPane.setConstraints(rgtButtonPlatform, 3, 2, 1, 1, HPos.RIGHT,
VPos.BOTTOM);
```

```
        GridPane.setConstraints(clwButtonPlatform, 3, 2, 1, 1, HPos.RIGHT,
VPos.TOP);
        GridPane.setConstraints(speedPlatform, 2, 2, 1, 1, HPos.LEFT,
VPos.BOTTOM);
        GridPane.setConstraints(cClwButtonPlatform, 3, 2, 1, 1, HPos.LEFT,
VPos.TOP);
        GridPane.setConstraints(startPump, 0, 2, 1, 1, HPos.CENTER,
VPos.BOTTOM);
        GridPane.setConstraints(flute, 1, 2, 1, 1, HPos.CENTER,
VPos.BOTTOM);
        GridPane.setConstraints(startRecordBtnPlatform, 0, 0, 1, 1,
HPos.LEFT, VPos.CENTER);
        GridPane.setConstraints(stopRecordBtnPlatform, 0, 0, 1, 1,
HPos.RIGHT, VPos.CENTER);
        GridPane.setConstraints(recCirclePlatform, 0, 1, 1, 1, HPos.LEFT,
VPos.TOP);
        GridPane.setConstraints(platformLabel, 2, 2, 1, 1, HPos.CENTER,
VPos.TOP);
        GridPane.setConstraints(platformGrid, 2, 1, 2, 1, HPos.LEFT,
VPos.CENTER);


        platformGrid.getColumnConstraints().add(new
ColumnConstraints(300));
        platformGrid.getColumnConstraints().add(new
ColumnConstraints(350));
        platformGrid.getRowConstraints().add(new RowConstraints(50));
        platformGrid.getRowConstraints().add(new RowConstraints(50));
        platformGrid.getRowConstraints().add(new RowConstraints(50));
        platformGrid.getRowConstraints().add(new RowConstraints(50));
        platformGrid.getRowConstraints().add(new RowConstraints(50));


        GridPane.setConstraints(dpModeButton, 1, 0, 1, 1, HPos.CENTER,
VPos.CENTER);
        GridPane.setConstraints(autopilotModeButton, 1, 1, 1, 1,
HPos.CENTER, VPos.CENTER);
        GridPane.setConstraints(manualModeButton, 1, 2, 1, 1, HPos.CENTER,
VPos.CENTER);
        GridPane.setConstraints(dpCirclePlatform, 1, 0, 1, 1, HPos.RIGHT,
VPos.CENTER);
        GridPane.setConstraints(autoCirclePlatform, 1, 1, 1, 1, HPos.RIGHT,
VPos.CENTER);
        GridPane.setConstraints(manualCirlcePlatform, 1, 2, 1, 1,
```

```
HPos.RIGHT, VPos.CENTER);


        g2dPlatform = platformCanvas.getGraphicsContext2D();
        timerPlatform = new AnimationTimer() {
        Mat matPlatform = new Mat();
        Image imagePlatform;
            @Override
            public void handle(long now) {
               Mat vidPlatform = vidUdpPlatform.getImage();
                if (vidPlatform == null){
                       imagePlatform = new
Image(getClass().getResourceAsStream("disconnectedScreen2.png"));

                }
                else if(vidPlatform != null){
                    imagePlatform = mat2Image(vidPlatform);

                }

                g2dPlatform.drawImage(imagePlatform, 0, 0);

                if (startRecPlatform && !stoprecPlatform) {
                    try {
                         writerPlatform.write(vidPlatform);
                    } catch (Exception e) {
                        writerPlatform.release();
                        System.out.println("stoppingThe recording
Platform");
                        stoprecPlatform = true;
                        startRecordBtnPlatform.disarm();
                        startRecPlatform = false;
                        recCirclePlatform.setFill(notRecPlatform);
                    }
                }
                if (stoprecPlatform) {
                    writerPlatform.release();
                }
            }
        };
        timerPlatform.start();

        // Add children
        platformGrid.getChildren().addAll(dpCirclePlatform, dpModeButton,
```

```
manualCirlcePlatform,
                manualModeButton, autoCirclePlatform, autopilotModeButton);

        platformLayout.getChildren().addAll(speedPlatform,
fwdButtonPlatform, bckButtonPlatform,
                lftButtonPlatform, rgtButtonPlatform, platformCanvas,
cClwButtonPlatform, clwButtonPlatform,
                startPump,flute, startRecordBtnPlatform,
stopRecordBtnPlatform, recCirclePlatform,
                platformLabel, platformGrid);

        ///////////// ROV TAB
        // Buttons
        Button fwdButtonROV = new Button("FWD");
        fwdButtonROV.setMinSize(100, 75);
        fwdButtonROV.setOnTouchPressed(e -> {
                                   rovGuicontroller.setFwd(true);})
;
        fwdButtonROV.setOnTouchReleased(e -> {
                                   rovGuicontroller.setFwd(false);}
);

        Button bckButtonROV = new Button("AFT");
        bckButtonROV.setMinSize(100, 75);
        bckButtonROV.setOnTouchPressed(e -> {
                                   rovGuicontroller.setRev(true);
                                   System.out.println("test av ROV
AFT knapp");});
        bckButtonROV.setOnTouchReleased(e -> {
                                   rovGuicontroller.setRev(false);}
);
        Button lftButtonROV = new Button("LEFT");
        lftButtonROV.setMinSize(125, 75);
        lftButtonROV.setOnTouchPressed(e -> {
                                   rovGuicontroller.setLeft(true);}
);
        lftButtonROV.setOnTouchReleased(e -> {
                                   rovGuicontroller.setLeft(false);
});
        Button rgtButtonROV = new Button("RIGHT");
        rgtButtonROV.setMinSize(125, 75);
         rgtButtonROV.setOnTouchPressed(e -> {
                                   rovGuicontroller.setRight(true);
});
```

```
         rgtButtonROV.setOnTouchReleased(e -> {
                                          rovGuicontroller.setRight(false)
;});
        Button clwButtonROV = new Button("CLW");
        clwButtonROV.setMinSize(100, 50);
        clwButtonROV.setOnTouchPressed(e -> {
                                          rovGuicontroller.setSlideLeft(tr
ue);});
        clwButtonROV.setOnTouchReleased(e -> {
                                          rovGuicontroller.setSlideLeft(fa
lse);});

        Button cClwButtonROV = new Button("CCLW");
        cClwButtonROV.setMinSize(100, 50);
        cClwButtonROV.setOnTouchPressed(e -> {
                                          rovGuicontroller.setSlideRight(t
rue);});
        cClwButtonROV.setOnTouchReleased(e -> {
                                          rovGuicontroller.setSlideRight(f
alse);});
        Button upButtonROV = new Button("UP");
        upButtonROV.setMinSize(125, 75);
         upButtonROV.setOnTouchPressed(e -> {
                                           sendObserver.setWinchUp(true);
                                           sendObserver.notifyObs();});
         upButtonROV.setOnTouchReleased(e -> {
                                           sendObserver.setWinchUp(false);
                                           sendObserver.notifyObs();});
        Button downButtonROV = new Button("DOWN");
        downButtonROV.setMinSize(125, 75);
        downButtonROV.setOnTouchPressed(e -> {
                                           sendObserver.setWinchDown(true)
;

                                           sendObserver.notifyObs();});
        downButtonROV.setOnTouchReleased(e -> {
                                           sendObserver.setWinchDown(false
);

                                           sendObserver.notifyObs();});





       Circle recCircleROV = new Circle(10, Color.TRANSPARENT);
```

```
        lockCircleROV = new Circle(10, Color.ORANGE);
        upperCircleROV = new Circle(10, Color.ORANGE);
        Color redDotROV = Color.RED;
        Color clearDotROV = Color.TRANSPARENT;
        Color greenDotROV = Color.GREEN;



        Button startRecordBtnROV = new Button("Record");
        startRecordBtnROV.setOnAction(e -> {
            writerROV = new
VideoWriter("C:\\Users\\Platform\\Desktop\\ROVVideo\\ROV\\"
                   +dateFormat.format(new Date())+".avi",
VideoWriter.fourcc
                    ('M', 'J', 'P', 'G'), 15,  new Size(320, 240));
            startRecROV = true;
            recCircleROV.setFill(redDotROV);
        });
        startRecordBtnROV.setMaxSize(125, 20);

        Button stopRecordBtnROV = new Button("Stop");
        stopRecordBtnROV.setMaxSize(125, 20);
        stopRecordBtnROV.setOnMousePressed(e -> {
            stoprecROV = true;
            startRecordBtnROV.disarm();
            startRecROV = false;
            recCircleROV.setFill(clearDotROV);
        });
        stopRecordBtnROV.setOnMouseReleased(e -> stoprecROV = false);

        // Labels
        Label rovLockLabel = new Label("ROV is LOCKED:");
        Label rovUpperPosLabel = new Label("Upper position:");

          // Slider
         Slider speedROV = new Slider(0, 100, 50);
         speedROV.setShowTickMarks(true);
         speedROV.setShowTickLabels(true);
         speedROV.setMajorTickUnit(25f);
         speedROV.setBlockIncrement(10f);


         //Speed for the ROV slider
         speedROV.valueProperty().addListener(new ChangeListener<Number>(){
```

```
        @Override
        public void changed(ObservableValue<? extends Number> observable,
                Number oldValue, Number newValue){
            System.out.println("New Value: " + newValue);

            rovGuicontroller.setThrusterValue(newValue.intValue());

        }
        });

         // Slider
        Slider lightRovSlider = new Slider(0, 100, 50);
        lightRovSlider.setShowTickMarks(true);
        lightRovSlider.setShowTickLabels(true);
        lightRovSlider.setMajorTickUnit(25f);
        lightRovSlider.setBlockIncrement(10f);


        //Speed for the ROV slider
        lightRovSlider.valueProperty().addListener(new
ChangeListener<Number>(){
        @Override
        public void changed(ObservableValue<? extends Number> observable,
                Number oldValue, Number newValue){
            System.out.println("New Value: " + newValue);

            rovGuicontroller.setLightValue(newValue.intValue());

        }
        });







        Slider speedWinch = new Slider(0, 100, 50);
        speedWinch.setShowTickMarks(true);
        speedWinch.setShowTickLabels(true);
        speedWinch.setMajorTickUnit(25f);
        speedWinch.setBlockIncrement(10f);

           speedWinch.valueProperty().addListener(new
```

```
ChangeListener<Number>(){
        @Override
        public void changed(ObservableValue<? extends Number> observable,
                Number oldValue, Number newValue){
            System.out.println("New Value: " + newValue);

            sendObserver.setWinchSpeed(newValue.intValue());
            sendObserver.notifyObs();
        }
        });

        Label winchLabel = new Label("Winch Speed");
        Label rovLabel = new Label("ROV Speed");
        Label rovLightLabel = new Label("Light Power");
        // Panes
        GridPane ROVLayout = new GridPane();
        GridPane ROVValuePane = new GridPane();
        Canvas ROVCanvas = new Canvas(SCENE_W, SCENE_H);

        // Constraints
        ROVLayout.getColumnConstraints().add(new ColumnConstraints(250));
        ROVLayout.getColumnConstraints().add(new ColumnConstraints(300));
        ROVLayout.getColumnConstraints().add(new ColumnConstraints(300));
        ROVLayout.getColumnConstraints().add(new ColumnConstraints(350));
        ROVLayout.getRowConstraints().add(new RowConstraints(50));
        ROVLayout.getRowConstraints().add(new RowConstraints(SCENE_H));
        ROVLayout.getRowConstraints().add(new RowConstraints(150));


        GridPane.setConstraints(ROVCanvas, 0, 1);
        GridPane.setHalignment(ROVCanvas, HPos.LEFT);
        GridPane.setConstraints(fwdButtonROV, 3, 2, 1, 1, HPos.CENTER,
VPos.TOP);
        GridPane.setConstraints(bckButtonROV, 3, 2, 1, 1, HPos.CENTER,
VPos.BOTTOM);
        GridPane.setConstraints(lftButtonROV, 3, 2, 1, 1, HPos.LEFT,
VPos.BOTTOM);
        GridPane.setConstraints(rgtButtonROV, 3, 2, 1, 1, HPos.RIGHT,
VPos.BOTTOM);
        GridPane.setConstraints(clwButtonROV, 3, 2, 1, 1, HPos.RIGHT,
VPos.TOP);
        GridPane.setConstraints(speedROV, 2, 2, 1, 1, HPos.LEFT,
VPos.BOTTOM);
        GridPane.setConstraints(cClwButtonROV, 3, 2, 1, 1, HPos.LEFT,
```

```
VPos.TOP);
        GridPane.setConstraints(startRecordBtnROV, 0, 0, 1, 1, HPos.LEFT,
VPos.CENTER);
        GridPane.setConstraints(stopRecordBtnROV, 0, 0, 1, 1, HPos.RIGHT,
VPos.CENTER);
        GridPane.setConstraints(recCircleROV, 0, 1, 1, 1, HPos.LEFT,
VPos.TOP);
        GridPane.setConstraints(upButtonROV, 0, 2, 1, 1, HPos.CENTER,
VPos.TOP);
        GridPane.setConstraints(downButtonROV, 0, 2, 1, 1, HPos.CENTER,
VPos.BOTTOM);
        GridPane.setConstraints(speedWinch, 1, 2, 1, 1, HPos.LEFT,
VPos.BOTTOM);
        GridPane.setConstraints(winchLabel, 1, 2, 1, 1, HPos.CENTER,
VPos.TOP);
        GridPane.setConstraints(rovLabel, 2, 2, 1, 1, HPos.CENTER,
VPos.TOP);
        GridPane.setConstraints(ROVValuePane, 3, 1, 1, 1);

        ROVValuePane.getColumnConstraints().add(new
ColumnConstraints(350));
        ROVValuePane.getRowConstraints().add(new RowConstraints(50));
        ROVValuePane.getRowConstraints().add(new RowConstraints(50));
        ROVValuePane.getRowConstraints().add(new RowConstraints(50));
        ROVValuePane.getRowConstraints().add(new RowConstraints(200));
        ROVValuePane.getRowConstraints().add(new RowConstraints(50));
        ROVValuePane.getRowConstraints().add(new RowConstraints(50));

        GridPane.setConstraints(lockCircleROV, 0, 0, 1, 1, HPos.RIGHT,
VPos.CENTER);
        GridPane.setConstraints(upperCircleROV, 0, 1, 1, 1, HPos.RIGHT,
VPos.CENTER);
         GridPane.setConstraints(lightRovSlider, 0, 5, 1, 1, HPos.CENTER,
VPos.BOTTOM);
        GridPane.setConstraints(rovLightLabel, 0, 4, 1, 1, HPos.CENTER,
VPos.TOP);
        GridPane.setConstraints(rovLockLabel, 0, 0, 1, 1, HPos.CENTER,
VPos.CENTER);
        GridPane.setConstraints(rovUpperPosLabel, 0, 1, 1, 1, HPos.CENTER,
VPos.CENTER);
```

```
        g2dROV = ROVCanvas.getGraphicsContext2D();
        timerROV = new AnimationTimer() {
        Mat matROV = new Mat();
        Image imageROV;
            @Override
            public void handle(long now) {
               Mat vidROV = vidUdpROV.getImage();
                if (vidROV == null){
                       imageROV = new
Image(getClass().getResourceAsStream("disconnectedScreen2.png"));
                }
                else if(vidROV != null){
                    imageROV = mat2Image(vidROV);
                }
                g2dROV.drawImage(imageROV, 0, 0);
                if (startRecROV && !stoprecROV) {
                    try {
                         writerROV.write(vidROV);
                    } catch (Exception e) {
                        writerROV.release();
                        System.out.println("stoppingThe recording");
                        stoprecROV = true;
                        startRecordBtnROV.disarm();
                        startRecROV = false;
                        recCircleROV.setFill(clearDotROV);
                    }

                }
                if (stoprecROV) {
                    writerROV.release();
                }
            }
        };
        timerROV.start();

        // Add children
        ROVValuePane.getChildren().addAll(rovLockLabel,rovLightLabel,lightR
ovSlider, rovUpperPosLabel,
                upperCircleROV, lockCircleROV);

        ROVLayout.getChildren().addAll(fwdButtonROV, bckButtonROV,
lftButtonROV,
                rgtButtonROV, clwButtonROV, cClwButtonROV, speedROV,
ROVCanvas,
```

```
                stopRecordBtnROV, startRecordBtnROV,recCircleROV,
upButtonROV,
                downButtonROV, speedWinch, winchLabel, rovLabel,
ROVValuePane);
        ///////////// Settings TAB

        //Buttons
        Button saveFileButton = new Button("Select");
        Button setRovIpButton = new Button("Set IP");
        Button setPlatformIpButton = new Button("Set IP");
        // TextFields
        TextField setSaveLocation = new TextField();
        TextField setRovIp = new TextField();
        TextField setPlatformIp = new TextField();
        GridPane settingsLayout = new GridPane();
        // Constraints
        settingsLayout.getColumnConstraints().add(new
ColumnConstraints(300));
        settingsLayout.getColumnConstraints().add(new
ColumnConstraints(150));

        settingsLayout.getRowConstraints().add(new RowConstraints(75));
        settingsLayout.getRowConstraints().add(new RowConstraints(75));
        settingsLayout.getRowConstraints().add(new RowConstraints(75));


        GridPane.setConstraints(setSaveLocation, 0, 0);
        GridPane.setConstraints(setRovIp, 0, 1);
        GridPane.setConstraints(setPlatformIp, 0, 2);

        GridPane.setConstraints(saveFileButton, 1, 0, 1, 1,
HPos.CENTER,VPos.CENTER);
        GridPane.setConstraints(setRovIpButton, 1, 1, 1, 1,
HPos.CENTER,VPos.CENTER);
        GridPane.setConstraints(setPlatformIpButton, 1, 2, 1, 1,
HPos.CENTER,VPos.CENTER);

        // Padding
        settingsLayout.setPadding(new Insets(20, 20, 20, 20));



        settingsLayout.getChildren().addAll(setSaveLocation,saveFileButton,
                setRovIp, setRovIpButton, setPlatformIp,
```

```
setPlatformIpButton);


        gridpaneOverview.setGridLinesVisible(false);
        settingsLayout.setGridLinesVisible(false);
        platformLayout.setGridLinesVisible(false);
        ROVLayout.setGridLinesVisible(false);
        ROVValuePane.setGridLinesVisible(false);
        platformGrid.setGridLinesVisible(false);

        // Add panes to tab
        overviewTab.setContent(gridpaneOverview);
        settingsTab.setContent(settingsLayout);
        platformTab.setContent(platformLayout);
        rovTab.setContent(ROVLayout);
        tabPane.getTabs().addAll(overviewTab, platformTab, rovTab,
settingsTab);

        Scene scene = new Scene(tabPane);
        scene.getStylesheets().add
        (SeaFarm.class.getResource("seafarm.css").toExternalForm());



        window.setOnCloseRequest(e -> closeProgram());
        window.setTitle("Aquaculture Seafarm");
        window.setFullScreen(true);
        window.setScene(scene);
        window.show();

    }

    //change this to, check each sensor and set value, so more  than one
can be
    // green at the same time

    private void updateMOdeIndicators(boolean dp,boolean auto, boolean
manual){
        Color redDotPlatform = Color.RED;
        Color greenDotPlatform = Color.GREEN;


        if (manual) {
```

```
            this.dpCirclePlatform.setFill(redDotPlatform);
            this.autoCirclePlatform.setFill(redDotPlatform);
            this.manualCirlcePlatform.setFill(greenDotPlatform);
        }

          else if (auto) {

            this.dpCirclePlatform.setFill(redDotPlatform);
            this.autoCirclePlatform.setFill(greenDotPlatform);
            this.manualCirlcePlatform.setFill(redDotPlatform);
        }
            else if (dp) {

            this.dpCirclePlatform.setFill(greenDotPlatform);
            this.autoCirclePlatform.setFill(redDotPlatform);
            this.manualCirlcePlatform.setFill(redDotPlatform);
        }
            else  {

            this.dpCirclePlatform.setFill(redDotPlatform);
            this.autoCirclePlatform.setFill(redDotPlatform);
            this.manualCirlcePlatform.setFill(redDotPlatform);
        }


    }

        private void updateRovLockingIndicator(boolean upperPos,boolean
locked){
        Color redDotRov = Color.RED;
        Color greenDotRov = Color.GREEN;





        if (upperPos) {

            this.upperCircleROV.setFill(greenDotRov);

        }

          else  {
```

```
            this.upperCircleROV.setFill(redDotRov);

        }


        if (locked) {

            this.lockCircleROV.setFill(greenDotRov);

        }
            else  {

            this.lockCircleROV.setFill(redDotRov);

        }


    }



    private void initOpenCv() {
        System.loadLibrary(Core.NATIVE_LIBRARY_NAME);

        //videoCapture = new VideoCapture();
        //videoCapture.open(0);

        //System.out.println("Camera open: " + videoCapture.isOpened());

    }

    public static Image mat2Image(Mat mat) {
        Mat resizeimage = new Mat();
        Size sz = new Size(640,480);
        Imgproc.resize( mat, resizeimage, sz );

        MatOfByte buffer = new MatOfByte();
        Imgcodecs.imencode(".jpg", resizeimage, buffer);
        //Image imToRe = new Image(new
ByteArrayInputStream(buffer.toArray()));

        return new Image(new ByteArrayInputStream(buffer.toArray()));
    }
```

```
    private Gauge makeGauge(String title, String unit){
              Gauge gauge = GaugeBuilder.create()
               .foregroundBaseColor(Color.WHITE)
               .title(title)
               .titleColor(Color.WHITE)
               .subTitle("Subtitle")
               .unit(unit)
               .unitColor(Color.WHITE)
               .valueColor(Color.WHITE)

               .build();

        return gauge;
    }

    private void setGaugeValue(double temperature,double dept){
        temp.setValue(temperature);
        depthMeter.setValue(dept);
    }


   private void setRovTextField(float headingVal,float t, float dpt,float
tW,float oxy){
        //text fields
        headingRovTextField.setText(Float.toString(headingVal));
        tempRovTextField.setText(Float.toString(t));
        deptRovTextField.setText(Float.toString(dpt));
        tempInWaterTextField.setText(Float.toString(tW));
        oxygenInWaterTextField.setText(Float.toString(oxy));
    }

    private void setPlatformTextField(float headingVal,float speed, float
ptc,float roll){
        //text fields
        headingPlatTextField.setText(Float.toString(headingVal));
        curretnSpeenTextField.setText(Float.toString(speed));
        pitchTextField.setText(Float.toString(ptc));
        rollTextField.setText(Float.toString(roll));
    }

    private void closeProgram(){
```

```
        //timerOverview.stop();
        timerPlatform.stop();
        timerROV.stop();
        //vidUdpOverview.stop();
        vidUdpPlatform.stop();
        vidUdpROV.stop();
        System.out.println("Printing positions");
        maps.printList();
        //videoCapture.release();

        window.close();
        System.out.println("Closed program");
        Platform.exit();
        System.exit(0);
    }

    @Override
    public void update(Observable o, Object arg) {
        if(o instanceof RecieveDataObserver){
            //System.out.println("updatemodbus in seafarm class");
            RecieveDataObserver recieve = (RecieveDataObserver) o;
            longitude = recieve.getCurrentLongitude();
            latitude = recieve.getCurrentLatitude();
            dpModeActivated = recieve.getDpMode();
            autoModeActivated = recieve.getAutoMode();
            manualModeActivated = recieve.getManualMode();

            //check if maps i initialized, will be false with no internet
            if (maps != null){
            //System.out.println("map is not 0");
                try {
                  maps.setLng(longitude);
                  maps.setLat(latitude);
                  maps.setRovPosition();



                //Function for calculating if the plaform should get a new
position.
                //while in DP or Autopilot
                if(dpModeActivated || autoModeActivated ){

                    if
((!dpModeActivated)&&((autMode.shouldGetNewPos(latitude, longitude,
```

```
                     autoModeLongSetpoint,
autoModeLatSetpoint))||((autoModeLatSetpoint==0)||(autoModeLongSetpoint==0)
)))
                        //if the platform is close enough to the desire
point,
                        //check if there is another point plotted on the
map
                  {     double mapSetpointLat=maps.getNextPosLat();
                        double mapSetpointLong=maps.getNextPosLng();
                        //if there is no marker plotted stay in current
position
                        if((mapSetpointLat==0)||(mapSetpointLong==0)){
                            autoModeLatSetpoint= latitude;
                            autoModeLongSetpoint=longitude;
                        }
                        else{
                            //Request the platform to hold a new position
                            autoModeLatSetpoint= mapSetpointLat;
                            autoModeLongSetpoint=mapSetpointLong;
                         }

                  }
                        sendObserver.setLatitude((float)autoModeLatSetpoint
);
                        sendObserver.setLongitude((float)autoModeLongSetpoi
nt);
                        //System.out.println(autoModeLongSetpoint
+"longSet");
                        sendObserver.notifyObs();
                }

                }
                catch (Exception e) {

                    //System.out.println(e+"in update maps");

                }



            updateMOdeIndicators(dpModeActivated, autoModeActivated,
manualModeActivated);
            currentSpeed = recieve.getCurrentSpeed();
            heading = recieve.getCurrentYaw();
```

```
            pitch = recieve.gettCurrentPitch();
            roll = recieve.getCurrentRoll();
            rovLocked = recieve.getRovIsLocked();
            rovUpperPos = recieve.getRovInupperPos();
            updateRovLockingIndicator(rovUpperPos, rovLocked);
            setPlatformTextField(heading,currentSpeed,pitch,roll);


            }
            }
                if(o instanceof RovReceiveDataObservable){
            float dpt=0;
            float tmpInrov=0;
            float tmpInSea=0;
            float headrov=0;
            float oxygen=0;



            RovReceiveDataObservable recieve = (RovReceiveDataObservable)
o;
            dpt=recieve.getDepth();
            tmpInrov=recieve.getTempInROV();
            tmpInSea=recieve.getTemperature();
            headrov=recieve.getHeading();
            oxygen=recieve.getOxygen();
            sendObserver.setDpt(dpt);
            sendObserver.setTmpInSea(tmpInSea);
            sendObserver.setTmpInrov(tmpInrov);
            sendObserver.setHeadrov(headrov);
            sendObserver.setOxygen(oxygen);
            sendObserver.notifyObs();

            //updating the textField for the ROV
            setRovTextField(headrov,tmpInrov,dpt,tmpInSea,oxygen);
            setGaugeValue((double)recieve.getTempInROV(),(double)recieve.ge
tDepth());



        }


    }
```

```
    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        launch(args);
    }

}
```

Maps class

```java
/*
 * 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 seafarm;

import com.teamdev.jxmaps.GeocoderCallback;
import com.teamdev.jxmaps.GeocoderRequest;
import com.teamdev.jxmaps.GeocoderResult;
import com.teamdev.jxmaps.GeocoderStatus;
import com.teamdev.jxmaps.InfoWindow;
import com.teamdev.jxmaps.LatLng;
import com.teamdev.jxmaps.Map;
import com.teamdev.jxmaps.Icon;
import com.teamdev.jxmaps.MapMouseEvent;
import com.teamdev.jxmaps.MapReadyHandler;
import com.teamdev.jxmaps.MapStatus;
import com.teamdev.jxmaps.MapViewOptions;
import com.teamdev.jxmaps.Marker;
import com.teamdev.jxmaps.Size;
import com.teamdev.jxmaps.PlaceSearchRequest;
import com.teamdev.jxmaps.MouseEvent;
import com.teamdev.jxmaps.Point;
import com.teamdev.jxmaps.Polyline;
import com.teamdev.jxmaps.PolylineOptions;
import com.teamdev.jxmaps.javafx.MapView;
import java.util.Iterator;
import java.util.LinkedList;
import java.util.List;
import java.util.ListIterator;

/**
 *
 * @author Platform
 */
public class maps {
    public MapView mapView;
    public Map map;
    public Marker platformMarker;
    public List<Marker> listMarker;
    public int lngCnt = 0;
    public int latCnt = 0;
```

```
    public double lat = 62.471948;
    public double lng = 6.235196;
    double nextPos1 = 0.0;
    private Icon platformIcon;
    public maps() {

        listMarker = new LinkedList<>();

        MapViewOptions options = new MapViewOptions();
        options.importPlaces();
        options.setApiKey("AIzaSyB_sLP3KV14XnGFYL5gAFAC6ErzJ-FrO48");
        mapView = new MapView(options);

        mapView.setOnMapReadyHandler(new MapReadyHandler() {
            @Override
            public void onMapReady(MapStatus status) {
                if (status == MapStatus.MAP_STATUS_OK) {
                    map = mapView.getMap();
                    map.setZoom(15.0);
                    GeocoderRequest request = new GeocoderRequest();
                    request.setAddress("Nørvevika, NO");

                    mapView.getServices().getGeocoder().geocode(request,
new GeocoderCallback(map) {
                        @Override
                        public void onComplete(GeocoderResult[] result,
GeocoderStatus status) {
                            if (status == GeocoderStatus.OK) {
                                //map.setCenter(result[0].getGeometry().get
Location());
                                map.setCenter(new LatLng(lat, lng));
                                platformMarker = new Marker(map);
                                  //Creating Icon for the platform
                                platformIcon = new Icon();
                                platformIcon.loadFromStream((getClass().get
ResourceAsStream("PlatformImg.png")),"png");

                                Size size=new Size(200,200);
                                platformIcon.setScaledSize(size);
                                platformIcon.setAnchor(new Point(100,100));



                                platformMarker.setIcon(platformIcon);
```

```
                                        //marker.setPosition(result[0].getG
eometry().getLocation());
                                platformMarker.setPosition(new LatLng(lat,
lng));

                                listMarker.add(platformMarker);


                                //

                      map.addEventListener("click", new MapMouseEvent() {
                        @Override
                        public void onEvent(MouseEvent mouseEvent) {
                            // Closing initially created info window

                            // Creating a new marker
                            final Marker marker2 = new Marker(map);
                            // Move marker to the position where user
clicked
                            marker2.setPosition(mouseEvent.latLng());
                            listMarker.add(marker2);

                            System.out.println("Adding position: " +
marker2.getPosition().toString());

                            // Adding event listener that intercepts
clicking on marker
                            marker2.addEventListener("click", new
MapMouseEvent() {
                                @Override
                                public void onEvent(MouseEvent mouseEvent)
{
                                    // Removing marker from the map
                                    marker2.remove();


                                    System.out.println("Removing position:
" + marker2.getPosition().toString());
                                    listMarker.remove(marker2);
                                    latCnt--;
                                    lngCnt--;
                                }
                            });
```

```
                        }
                    });

                            }
                        }
                    });
                }
            }
        });
    }

    public MapView getMapView() {
        return mapView;
    }
    public void printList(){
        Iterator<Marker> iterator = listMarker.iterator();
        while(iterator.hasNext()){
            System.out.println("Positions in list at closing: " +
iterator.next().getPosition().toString());
        }
    }

    public double getNextPosLat(){
        System.out.println("get next Pos lat called MAPSclass ");
        double nextPos = 0.00;
        if(listMarker.iterator().hasNext()){
          if(latCnt == listMarker.size()){
              latCnt = listMarker.size() - 1;
             // System.out.println(latCnt);
              nextPos = listMarker.get(latCnt).getPosition().getLat();
          }
          else if(latCnt  < listMarker.size()){
              nextPos = listMarker.get(latCnt).getPosition().getLat();
              latCnt++;
              //System.out.println(latCnt);
          }
        }
        return nextPos;
    }
    public double getNextPosLng(){
        double nextPos = 0.00;
        if(listMarker.iterator().hasNext()){
          if(lngCnt == listMarker.size()){
              lngCnt = listMarker.size() - 1;
```

```
             // System.out.println(lngCnt);
              nextPos = listMarker.get(lngCnt).getPosition().getLng();
          }
          else if(lngCnt  < listMarker.size()){
              nextPos = listMarker.get(lngCnt).getPosition().getLng();
              lngCnt++;
             // System.out.println(lngCnt);
          }
        }
        return nextPos;
    }

    public double getNext(){

        for(int i = 0; i < listMarker.size();i++){
            if(nextPos1 == 0.0){
                nextPos1 = listMarker.get(0).getPosition().getLat();
                //System.out.println("If1");
                break;

            }
            if((listMarker.get(i).getPosition().getLat() == nextPos1) && i
+ 1 != listMarker.size()){
                nextPos1 = listMarker.get(i+1).getPosition().getLat();
                //System.out.println("If2");
                break;
            }
            else if(i + 1 == listMarker.size()){
                System.out.println("else");
                break;
            }
        }

        return nextPos1;
    }

    public void setLat(float latitude){
        lat = (float) Double.parseDouble(Float.toString(latitude));


    }
    public void setLng(float longitude){
        lng = (float) Double.parseDouble(Float.toString(longitude));
    }
```

```
    public void setRovPosition(){
      //  platformMarker.setCenter(new LatLng(lat, lng));







        //marker.setPosition(result[0].getGeometry().getLocation());

        platformMarker.setPosition(new LatLng(lat, lng));
        //listMarker.add(marker);
    }




}
```

Modbus Sender

```
/*
 * 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 seafarm;

import de.re.easymodbus.modbusclient.ModbusClient;
import java.util.Observable;
import java.util.Observer;
import java.util.logging.Level;
import java.util.logging.Logger;
   //modbus client, set to the same client as the sender in constructor//

/**
 *
 * @author Platform
 */
public class ModbusSender implements Observer,Runnable {

     private ModbusClient client;
     private Thread t;
     private boolean connected=false;
     private int timeBetweenSending =200;
     private long timeVar=0;
     private boolean varHasBeenUpdated=false;
     private boolean timeElapsed=false;

    //platform movement commands//
     int thrusterSpeed=0;
     boolean fwdMotion = false;
     boolean bckMotion = false;
     boolean rightMotion = false;
     boolean leftMotion = false;
     boolean clockWMotion = false;


     boolean counterClocMotion = false;
     boolean enableLight=false;
     boolean enableFlute=false;
     boolean platformEnable=false;
     boolean enableAuto=false;
     boolean enableManual=false;
```

```
 //DpMode commands//
 boolean dpModeEnable=false;
 float latitude=0;
 float longitude=0;



 //winch commands//
 boolean winchUp=false;
 boolean winchDown=false;
 int winchSpeed=0;
 boolean winchLockOn=false;
 boolean winchLockOff=false;

 //Pump commands//
 boolean startPump = false;

 //ROV data for logging on plc
  //rov data for logging
 float rovDpt=0;
 float rovTmpInrov=0;
 float rovTmpInSea=0;
 float rovHeading=0;
 float rovOxygen=0;

 // var for plc to check if gui is not connected
 boolean guiConcheckVar=false;
 //Modbus Registers to write on///
static int thrusterSpeerReg=32040;
static int fwdMotionReg=32769;
static int bckMotionReg=32770;
static int rightMotionReg=32771;
static int leftMotionReg=32772;
static int clockMotionReg=32773;
static int counterClMotionReg=33057;
static int enableLightReg=32774;
static int enableFluteReg=32775;
static int platfromEnableReg=32776;
static int enableAturoReg=32777;
static int enableManualReg=32778;
static int enableDpModeReg=32779;
static int longitudeReg=32048;
```

```
    static int latitudeReg=32044;
    static int startPumpReg=33056;
    static int winchUpReg=32780;
    static int winchDownReg=32781;
    static int winchLockOnReg=32782;
    static int winchLockOffReg=32783;
    static int winchSpeedReg=32042;
    static int guiConcheckVarReg= 33058;
    //rov
    static int rovHeadReg= 32034;
    static int rovInternalTempReg= 32022;
    static int rovExternalTempReg= 32030;
    static int rovDeptReg= 32028;
    static int rovOxygenReg= 32032;
public ModbusSender(ModbusClient modbusClient){
    this.client=modbusClient;

}


    public void start() {
            t = new Thread(this, "MODBUSreceiverThread");
            t.start();
            System.out.println("Starting");
        }



     public void run() {



    while(true){
        try {
            Thread.sleep(8000);
        } catch (InterruptedException ex) {
            Logger.getLogger(ModbusReciever.class.getName()).log(Level.SEVE
RE, null, ex);
        }



        while(!connected){
            try {
            System.out.println("sender Trying to connect to server");
            client.Connect();
            connected=true;
                }
```

```
            catch (Exception e) {
                connected=false;
                System.out.println(e);
                continue;

             }


        }


            while (connected) {

                if (System.currentTimeMillis()>= timeVar){
                    timeElapsed=true;
                }else{
                    timeElapsed=false;
                }


                if (varHasBeenUpdated || timeElapsed){
                try{

                    //sends true and false concheck so the plc can check if
it changes,
                    if(guiConcheckVar){
                        guiConcheckVar=false;
                    }
                    else{
                       guiConcheckVar=true;
                    }

                    sendData();
                   // System.out.println("Data sent to PLC");
                    varHasBeenUpdated=false;
                    timeVar=System.currentTimeMillis()+timeBetweenSending;
                     }

            catch (Exception e)
            {
                connected=false;
                System.out.println(e);
            }
            }
```

```
    }
     }




}




    public void sendData(){
        try {

            ///writing coils////booleans///
            //todoo: endre til wirte multiple coils.
            this.client.WriteSingleCoil(fwdMotionReg, fwdMotion);
            this.client.WriteSingleCoil(bckMotionReg, bckMotion);
            this.client.WriteSingleCoil(rightMotionReg, rightMotion);
            this.client.WriteSingleCoil(leftMotionReg, leftMotion);
            this.client.WriteSingleCoil(clockMotionReg, clockWMotion);
            this.client.WriteSingleCoil(counterClMotionReg,
counterClocMotion);
            this.client.WriteSingleCoil(enableLightReg, enableLight);
            this.client.WriteSingleCoil(enableFluteReg, enableFlute);
            this.client.WriteSingleCoil(platfromEnableReg, platformEnable);
            this.client.WriteSingleCoil(enableAturoReg, enableAuto);
            this.client.WriteSingleCoil(enableManualReg, enableManual);
            this.client.WriteSingleCoil(enableDpModeReg, dpModeEnable);
            this.client.WriteSingleCoil(winchUpReg, winchUp);
            this.client.WriteSingleCoil(winchDownReg, winchDown);
            System.out.println(winchDown);
            this.client.WriteSingleCoil(winchLockOnReg, winchLockOn);
            this.client.WriteSingleCoil(winchLockOffReg, winchLockOff);
            this.client.WriteSingleCoil(startPumpReg, startPump);
            this.client.WriteSingleCoil(guiConcheckVarReg, guiConcheckVar);

            ///writing numeric values///
            //send float to two registers
            this.client.WriteMultipleRegisters(latitudeReg,
ModbusClient.ConvertFloatToTwoRegisters((float) latitude));
            this.client.WriteMultipleRegisters(longitudeReg,
ModbusClient.ConvertFloatToTwoRegisters((float) longitude));
            //rov values for logging on plc
            this.client.WriteMultipleRegisters(rovOxygenReg,
```

```
ModbusClient.ConvertFloatToTwoRegisters((float) rovOxygen));
            this.client.WriteMultipleRegisters(rovDeptReg,
ModbusClient.ConvertFloatToTwoRegisters((float) rovDpt));
            this.client.WriteMultipleRegisters(rovExternalTempReg,
ModbusClient.ConvertFloatToTwoRegisters((float) rovTmpInSea));
            this.client.WriteMultipleRegisters(rovInternalTempReg,
ModbusClient.ConvertFloatToTwoRegisters((float) rovTmpInrov));
            this.client.WriteMultipleRegisters(rovHeadReg,
ModbusClient.ConvertFloatToTwoRegisters((float) rovHeading));
            //System.out.println(longitude +"longSetInmodbusSend");
            // send int values to registers//
            this.client.WriteSingleRegister(thrusterSpeerReg,
thrusterSpeed);
            this.client.WriteSingleRegister(winchSpeedReg, winchSpeed);




        } catch (Exception e) {

            System.out.println("cant send to mobusregisters");
            connected=false;
            run();
        }


     }





    @Override
    public void update(Observable o, Object arg) {
        if(o instanceof SendDataObserver){
           // System.out.println("Update has been called ");
            SendDataObserver sendOb = (SendDataObserver) o;
            this.fwdMotion=sendOb.isFwdMotion();
            this.bckMotion=sendOb.isBckMotion();
            this.leftMotion=sendOb.isLeftMotion();
            this.rightMotion=sendOb.isRightMotion();
            this.clockWMotion=sendOb.isClockWMotion();
```

```
            this.counterClocMotion=sendOb.isCounterClockWMotion();
            this.dpModeEnable=sendOb.isDpModeEnable();
            this.enableManual=sendOb.isEnableManual();
            this.enableAuto=sendOb.enableAuto;
            this.thrusterSpeed=sendOb.getThrusterSpeed();
            this.winchSpeed=sendOb.getWinchSpeed();
            this.longitude=sendOb.getLongitude();
            this.latitude=sendOb.getLatitude();
            this.enableFlute=sendOb.isEnableFlute();
            this.winchDown=sendOb.isWinchDown();
            this.winchUp=sendOb.isWinchUp();

            //rov data
            this.rovDpt=sendOb.getDpt();
            this.rovHeading=sendOb.getHeadrov();
            this.rovOxygen=sendOb.getOxygen();
            this.rovTmpInSea=sendOb.getTmpInSea();
            this.rovTmpInrov=sendOb.getTmpInrov();
            varHasBeenUpdated=true;
        }

    }
}
```

Modbus Receiver

```java
/*
 * 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 seafarm;
import de.re.easymodbus.modbusclient.*;
import java.io.IOException;
import java.net.DatagramPacket;
import java.net.DatagramSocket;
import java.util.Arrays;
import java.util.logging.Level;
import java.util.logging.Logger;
/**
 *
 * @author sigurdolav
 */
public class ModbusReciever implements Runnable{

    //modbus client, set to the same client as the sender in constructor//
    private ModbusClient client;
    private Thread t;
    private RecieveDataObserver observer;
    private boolean connected=false;
    //data to recieve from platform///
    private float platLat=0;
    private float platLong=0;
    private float currentSpeed=0;
    private float pitch=0;
    private float yaw=0;
    private float roll=0;
    private int thrusterPower=0;
    private boolean enabled=false;
    private boolean rovLocked=false;
    private boolean rovUpperPos=false;
    private boolean dpMode = false;
    private boolean autoMode = false;
    private boolean manualMode = false;


    //the register adress of the different variables on the PLC//
    //heading forandres til pitch
    private final int yawReg=4;
```

```
    private final int rollReg=8;
    private final int enabledReg=192;
    private final int pitchReg=14;
    private final int currenSpeedReg=12;
    private final int platLongReg=0;
    private final int platLatReg=28;
    private final int rovLockedReg=384;
    private final int rovUpperPosReg=400;
    private final int dpModeReg = 416;
    private final int autoModeReg = 387;
    private final int manualModeReg = 388;


    private int readFrequency=1; //2 times a secound if var =2

    public ModbusReciever(ModbusClient modbusClient,RecieveDataObserver
obs){

        this.client=modbusClient;
        this.observer=obs;


    }

     /**
     * create thread and starts it
     */
    public void start() {
        t = new Thread(this, "MODBUSreceiverThread");
        t.start();
        System.out.println("Starting");
    }

        /**
     * read the registers on the PLC and update the recieve data observer
     */
    public void run() {
        try {
            Thread.sleep(8000);
        } catch (InterruptedException ex) {
            Logger.getLogger(ModbusReciever.class.getName()).log(Level.SEVE
RE, null, ex);
        }
        if (observer != null) {
```

```
            while(!connected){
            try {
            System.out.println("Modbus recieverTrying to connect to
server");
            client.Connect();
            connected=true;
                }
            catch (Exception e) {
                connected=false;
                System.out.println("connecting exception Modbus reciever");
                continue;

            }
            }


            try{
                while (observer.shouldChildOfThisRun()) {

                    recieveDataFromServer();
                    //System.out.println("recieving data from server");
                    updateRecieveObserver();
                    //System.out.println("updatingRecieverObserver");
                    //System.out.println("modbus has been recieved");
                    Thread.sleep(1000/this.readFrequency);

                     }
                }
            catch (Exception e)
            {
                connected=false;
                System.out.println("exception reciev modbus under update
recieve"+e);
            }
        }

        else {
            System.out.println("receive datahandler not created in
udpreceiver thread");
        }
        try {
            client.Disconnect();
```

```
    }

    catch (Exception e)
    {
        System.out.println("modbustcp.ModbusReciever.disconnect()");
    }

}

private void recieveDataFromServer(){

   // System.out.println("Trying to read from plc");
    this.pitch=ReadFloat(pitchReg);
    this.platLat=ReadFloat(platLatReg);
    this.platLong=ReadFloat(platLongReg);
    //System.out.println(this.platLong);
    //System.out.println(this.platLat);

    this.roll = ReadFloat(rollReg);
    this.currentSpeed=ReadFloat(currenSpeedReg);
    this.yaw=ReadFloat(yawReg);
    this.enabled=ReadBool(enabledReg);
    this.rovLocked=ReadBool(rovLockedReg);
    this.rovUpperPos=ReadBool(rovUpperPosReg);
    this.dpMode = ReadBool(dpModeReg);
    this.autoMode = ReadBool(autoModeReg);
    this.manualMode = ReadBool(manualModeReg);
    // System.out.println("finsished RecieveDataFrom Server");

}

private void updateRecieveObserver(){


    observer.setCurrentPitch(this.pitch);

    observer.setCurrentLatitude(this.platLat);
    observer.setCurrentLongitude(this.platLong);
    observer.setCurrentRoll(this.roll);
    observer.setCurrentSpeed(this.currentSpeed);
    observer.setCurrentYaw(this.yaw);
    observer.setEnabled(this.enabled);
    observer.setRovLock(this.rovLocked);
    observer.setRovUpperPos(this.rovUpperPos);
```

```
        observer.setDpMode(this.dpMode);
        observer.setAutoMode(this.autoMode);
        observer.setManualMode(this.manualMode);

        observer.notifyObs();
        //System.out.println("observer call ");


    }
    //example         //Read Float Value from Register 10 and 11
                        //System.out.println(ModbusClient.ConvertRegistersT
oFloat(modbusClient.ReadHoldingRegisters(9, 2)));
    private float ReadFloat(int reg){
        int regval=reg;
        float val=0;
        try
        {
        val=
this.client.ConvertRegistersToFloat(this.client.ReadHoldingRegisters(regval
, 2));

        }
        catch(Exception e)
        {
            System.out.println("modbustcp.ModbusReciever.ReadFloat()");
        }
        return val;
    }

    private boolean ReadBool(int reg){
       boolean retVal=false;
       boolean[]  val;
        try {
         val=client.ReadCoils(reg, 1);
         retVal= val[0];
        } catch (Exception e)
        {
            System.out.println("modbustcp.ModbusReciever.ReadBool()");
            connected=false;
            run();

        }
        return retVal;
```

```
    }

}
```

Receive Data Observer

```
package seafarm;


import java.math.BigInteger;
import java.util.Arrays;
import java.util.Observable;
import java.util.Observer;

/**
 * the observer to check the data received from platform

 */
public class RecieveDataObserver extends Observable {

       //data to recieve from platform///
     float longitude=0;
     float latitude=0;
     float currentSpeed=0;
     float pitch=0;
     float yaw=0;
     float roll=0;
     boolean enabled=false;
     boolean dpMode = false;
     boolean autoMode = false;
     boolean manualMode = false;
     //winch variables//
     boolean rovLocked=false;
     boolean rovUpperPos=false;



    public RecieveDataObserver() {


    }

    /**
     * adds the observer to the object
     *
     * @param o
     */
    @Override
    public synchronized void addObserver(Observer o) {
```

```
        super.addObserver(o); //To change body of generated methods, choose
Tools | Templates.
    }


    /*functions to update the platform variables*/

    public void setCurrentSpeed(float speed){
        this.currentSpeed=speed;
    }

    public void setEnabled(boolean enabl){
        this.enabled=enabl;
    }

    public void setCurrentPitch(float ptc){
        this.pitch=ptc;
    }

    public void setCurrentLongitude(float lng){
        this.longitude=lng;
    }

    public void setCurrentLatitude(float lat){
        this.latitude=lat;
    }
    public void setCurrentYaw(float yaw){
        this.yaw=yaw;

    }
    public void setCurrentRoll(float roll){
        this.roll=roll;

    }

    public void setRovLock(boolean lockOn){
        this.rovLocked=lockOn;
}
    public void setRovUpperPos(boolean isInupper){
        this.rovUpperPos=isInupper;
    }

    ///Get the current platform variables//
```

```
    public float getCurrentSpeed(){
        return this.currentSpeed;
    }

    public boolean getEnabled(){
        return this.enabled;
    }

    public float gettCurrentPitch(){
        return this.pitch;
    }

    public float getCurrentLongitude(){
        return this.longitude;
    }

    public float getCurrentLatitude(){
        return this.latitude;
    }
    public float getCurrentYaw(){
        return this.yaw;

    }
    public float getCurrentRoll(){
        return this.roll;

    }

    public boolean getRovInupperPos(){
        return this.rovUpperPos;
    }

    public boolean getRovIsLocked(){
        return this.rovLocked;
    }

    public boolean getDpMode() {
        return dpMode;
    }

    public void setDpMode(boolean dpMode) {
        this.dpMode = dpMode;
    }
```

```
    public boolean getAutoMode() {
        return autoMode;
    }

    public void setAutoMode(boolean autoMode) {
        this.autoMode = autoMode;
    }

    public boolean getManualMode() {
        return manualMode;
    }

    public void setManualMode(boolean manualMode) {
        this.manualMode = manualMode;
    }



    //notify the observer that the data has changed///
    public  void notifyObs() {
         setChanged();
         notifyObservers();

    }

    public boolean shouldChildOfThisRun() {
        //return datahandler.shouldThreadRun();
        return true;
    }


}
```

Send Data Observer

```java
package seafarm;


import java.math.BigInteger;
import java.util.Arrays;
import java.util.Observable;
import java.util.Observer;

/**
 * the observer to check the data received from platform

 */
public class SendDataObserver extends Observable {

       //data to send to platform///

    //platform movement commands//
     int thrusterSpeed=0;
     boolean fwdMotion = false;
     boolean bckMotion = false;
     boolean rightMotion = false;
     boolean leftMotion = false;
     boolean clockWMotion = false;


     boolean counterClockWMotion = false;
     boolean enableLight=false;
     boolean enableFlute=false;
     boolean platformEnable=false;
     boolean enableAuto=false;
     boolean enableManual=false;


     //DpMode commands//
     boolean dpModeEnable=false;
     float latitude=0;
     float longitude=0;



     //winch commands//
     boolean winchUp=false;
     boolean winchDown=false;
     int winchSpeed=0;
```

```
     boolean winchLockOn=false;
     boolean winchLockOff=false;

     //Pump commands//
     boolean startPump = false;
     //rov data for logging
     float dpt=0;
     float tmpInrov=0;
     float tmpInSea=0;
     float headrov=0;
     float oxygen=0;










    public SendDataObserver() {



    }

    /**
     * adds the observer to the object
     *
     * @param o
     */
    @Override
    public synchronized void addObserver(Observer o) {
        super.addObserver(o); //To change body of generated methods, choose
Tools | Templates.
    }

    //set functions for variables that the observer hodlds;
    public  void setThrusterSpeed(int thrusterSpeed) {
         this.thrusterSpeed = thrusterSpeed;

    }
```

```java
public void setEnableLight(boolean enableLight) {
    this.enableLight = enableLight;
}

public void setEnableFlute(boolean enableFlute) {
    this.enableFlute = enableFlute;
}

public void setPlatformEnable(boolean platformEnable) {
    this.platformEnable = platformEnable;
}

public void setEnableAuto(boolean enableAuto) {
    this.enableAuto = enableAuto;
}

public void setEnableManual(boolean enableManual) {
    this.enableManual = enableManual;
}

public void setDpModeEnable(boolean dpModeEnable) {
    this.dpModeEnable = dpModeEnable;
}

public void setLatitude(float latitude) {
    this.latitude = latitude;
}

public void setLongitude(float longitude) {
    this.longitude = longitude;
}

public void setWinchUp(boolean winchUp) {
    this.winchUp = winchUp;
}

public void setWinchDown(boolean winchDown) {
    this.winchDown = winchDown;
}

public void setWinchSpeed(int winchSpeed) {
    this.winchSpeed = winchSpeed;
}
```

```java
public void setWinchLockOn(boolean winchLockOn) {
    this.winchLockOn = winchLockOn;
}


public void setWinchLockOff(boolean winchLockOff) {
    this.winchLockOff = winchLockOff;
}


///functions to get the observer variables///

public int getThrusterSpeed() {
    return thrusterSpeed;
}


public boolean isEnableLight() {
    return enableLight;
}

public boolean isEnableFlute() {
    return enableFlute;
}

public boolean isPlatformEnable() {
    return platformEnable;
}

public boolean isEnableAuto() {
    return enableAuto;
}

public boolean isEnableManual() {
    return enableManual;
}

public boolean isDpModeEnable() {
    return dpModeEnable;
}

public float getLatitude() {
    return latitude;
}
```

```
public float getLongitude() {
    return longitude;
}

public boolean isWinchUp() {
    return winchUp;
}

public boolean isWinchDown() {
    return winchDown;
}

public int getWinchSpeed() {
    return winchSpeed;
}

public boolean isWinchLockOn() {
    return winchLockOn;
}

public boolean isWinchLockOff() {
    return winchLockOff;
}

    public boolean isFwdMotion() {
    return fwdMotion;
}

public void setFwdMotion(boolean fwdMotion) {
    this.fwdMotion = fwdMotion;
}

public boolean isBckMotion() {
    return bckMotion;
}

public void setBckMotion(boolean bckMotion) {
    this.bckMotion = bckMotion;
}

public boolean isRightMotion() {
    return rightMotion;
}
```

```
public void setRightMotion(boolean rightMotion) {
    this.rightMotion = rightMotion;
}

public boolean isLeftMotion() {
    return leftMotion;
}

public void setLeftMotion(boolean leftMotion) {
    this.leftMotion = leftMotion;
}

public boolean isClockWMotion() {
    return clockWMotion;
}

public void setClockWMotion(boolean clockWMotion) {
    this.clockWMotion = clockWMotion;
}

public boolean isCounterClockWMotion() {
    return counterClockWMotion;
}

public void setCounterClockWMotion(boolean counterClockWMotion) {
    this.counterClockWMotion = counterClockWMotion;
}

public boolean isStartPump() {
    return startPump;
}

public void setStartPump(boolean startPump) {
    this.startPump = startPump;
}

//rovdata

 public float getDpt() {
    return dpt;
}

public void setDpt(float dpt) {
```

```
        this.dpt = dpt;
    }

    public float getTmpInrov() {
        return tmpInrov;
    }

    public void setTmpInrov(float tmpInrov) {
        this.tmpInrov = tmpInrov;
    }

    public float getTmpInSea() {
        return tmpInSea;
    }

    public void setTmpInSea(float tmpInSea) {
        this.tmpInSea = tmpInSea;
    }

    public float getHeadrov() {
        return headrov;
    }

    public void setHeadrov(float headrov) {
        this.headrov = headrov;
    }

    public float getOxygen() {
        return oxygen;
    }


    public void setOxygen(float oxygen) {
        this.oxygen = oxygen;
    }



    //notify the observer that the data has changed///
    public void notifyObs() {
        setChanged();
        notifyObservers();
    }
```

```
    public boolean shouldChildOfThisRun() {
        //return datahandler.shouldThreadRun();
        return true;
    }



}
```

ROV Data Handler

```
package seafarm;

import java.math.BigInteger;
import java.util.Arrays;


/**
 *
 * @author Jørgen
 */

/*

Data transfer Protocoll:

//////FROM GUI TO RASBERRY PI///////

Byte 0
    - Bit [0] = Stopp
    - Bit [1] = Forward
    - Bit [2] = Reverse
    - Bit [3] = Left
    - Bit [4] = Right
    - Bit [5] = glideLeft
    - Bit [6] = glideRight
    - Bit [7] = **free**

Byte 1
    - light Value (mapped from 0 - 100)

Byte 2
    - thruster Value   (mapped from 0 - 100)


Byte 3
    - Bit [0] = **free**
    - Bit [1] = **free**
    - Bit [2] = Start/stop ROV   (value 1 = Start, value 0 = stop)
    - Bit [3] = **free**
    - Bit [4] = **free**
    - Bit [5] = **free**
    - Bit [6] = **free**
    - Bit [7] = **free**
```

```
Byte 4
    - **RESERVED**


//////FROM RASBERRY TO GUI///////

Receiving protocol:
Byte 0: Pixy x value lowbyte
Byte 1: Pixy x value highbyte
Byte 2: Pixy y value lowbyte
Byte 3: Pixy y value highbyte
Byte 4: Distance sensor
Byte 5: reserved


 */
public class RovDatahandler {

    private byte[] dataFromArduino;
    private byte[] dataFromGui;
    private boolean dataFromArduinoAvaliable = false;
    private boolean dataFromGuiAvailable = false;
    private boolean threadStatus;
    // private int pixyXvalue;
    // private int pixyYvalue;
    private int distanceSensor;
    private byte requestCodeFromArduino;
    private boolean enableROV;


    public RovDatahandler() {
        this.dataFromArduino = new byte[23];
        this.dataFromGui = new byte[6];
        //this.sendData = new SendData();
    }

    //*****************************************************************
    //********** PRIVATE METHODS AREA**********************************
    private byte setBit(byte b, int bit) {
        return b |= 1 << bit;
    }

    private byte releaseBit(byte b, int bit) {
        return b &= ~(1 << bit);
```

```
}

//************************************************************
//********************* THREAD STATUS METHODS*******************
public boolean shouldThreadRun() {
    return threadStatus;
}

public void setThreadStatus(boolean threadStatus) {
    this.threadStatus = threadStatus;
}

//***********************************************************
//************** FROM GUI METHODS*******************************
/**
 * returns the byte array protokoll sendt from GUI
 *
 * @return dataFromGui
 */
public byte[] getDataFromGui() {
    SeaFarm.enumStateEvent = RovSendEventState.FALSE;
    return this.dataFromGui;

}

/**
 * setting the byte from protocoll to high
 */
public void stopROV() {
    dataFromGui[0] = this.setBit(dataFromGui[0], 0);
    //this.fireStateChanged();
    this.sendData();
}

/**
 * setting the byte from protocoll to Low
 */
public void releaseStopROV() {
    dataFromGui[0] = this.releaseBit(dataFromGui[0], 0);
    // this.fireStateChanged();
    this.sendData();
}

/**
```

```
     * setting the byte from protocoll to high
     */
    public void goFwd() {
        dataFromGui[0] = this.setBit(dataFromGui[0], 1);
        // this.fireStateChanged();
        this.sendData();
    }

    /**
     * setting the byte from protocoll to Low
     */
    public void releaseGoFwd() {
        dataFromGui[0] = this.releaseBit(dataFromGui[0], 1);
        // this.fireStateChanged();
        this.sendData();
    }

    /**
     * setting the byte from protocoll to high
     */
    public void goRew() {
        dataFromGui[0] = this.setBit(dataFromGui[0], 2);
        // this.fireStateChanged();
        this.sendData();
    }

    /**
     * setting the byte from protocoll to Low
     */
    public void releaseGoRew() {
        dataFromGui[0] = this.releaseBit(dataFromGui[0], 2);
        // this.fireStateChanged();
        this.sendData();
    }

    /**
     * setting the byte from protocoll to high
     */
    public void goLeft() {
        dataFromGui[0] = this.setBit(dataFromGui[0], 3);
        //this.fireStateChanged();
        this.sendData();
    }
```

```
    /**
     * setting the byte from protocoll to Low
     */
    public void releaseGoLeft() {
        dataFromGui[0] = this.releaseBit(dataFromGui[0], 3);
        // this.fireStateChanged();
        this.sendData();
    }

    /**
     * setting the byte from protocoll to high
     */
    public void goRight() {
        dataFromGui[0] = this.setBit(dataFromGui[0], 4);
        //this.fireStateChanged();
        this.sendData();
    }

    public void releaseGoRight() {
        dataFromGui[0] = this.releaseBit(dataFromGui[0], 4);
        //this.fireStateChanged();
        this.sendData();
    }

    /**
     * setting the byte from protocoll to high
     */
    public void setSlideLeft() {
        dataFromGui[0] = this.setBit(dataFromGui[0], 5);
        //this.fireStateChanged();
        this.sendData();
    }

    /**
     * setting the byte from protocoll to Low
     */
    public void resetSlideLeft() {
        dataFromGui[0] = this.releaseBit(dataFromGui[0], 5);
        //this.fireStateChanged();
        this.sendData();
    }

    /**
     * setting the byte from protocoll to high
```

```
     */
    public void setSlideRight() {
        dataFromGui[0] = this.setBit(dataFromGui[0], 6);
        //this.fireStateChanged();
        this.sendData();
    }

    /**
     * setting the byte from protocoll to Low
     */
    public void resetSlideRight() {
        dataFromGui[0] = this.releaseBit(dataFromGui[0], 6);
        //this.fireStateChanged();
        this.sendData();
    }

    /**
     * setting the byte from protocoll to high
     */
    public void setLightValue(byte sensetivity) {
        dataFromGui[1] = sensetivity;
        //this.fireStateChanged();
        this.sendData();
    }

    /**
     *
     * @return dataFromGui on byte 1
     */
    public byte getLightValue() {
        return dataFromGui[1];
    }

    /**
     * setting the byte from protocoll to high
     */
    public void setThrusterValue(byte sensetivity) {
        dataFromGui[2] = sensetivity;
        //this.fireStateChanged();
        this.sendData();
    }

    /**
     *
```

```
     * @return dataFromGui on byte 2
     */
    public byte getThrusterValue() {
        return dataFromGui[2];
    }

    public void enableROV() {
        dataFromGui[3] = this.setBit(dataFromGui[3], 3);
        //this.fireStateChanged();
        this.sendData();
    }

    public void disableROV() {
        dataFromGui[3] = this.releaseBit(dataFromGui[3], 3);
        //this.fireStateChanged();
        this.sendData();
    }

    /**
     *
     * @return dataFromGui on byte 5
     */
    public byte getRequestCode() {
        return dataFromGui[5];
    }

    /**
     * increments the byte on array[5] with one for each run
     */
    public void incrementRequestCode() {
        dataFromGui[5]++;
        //this.fireStateChanged();
        this.sendData();
    }

    /**
     * creates a new udp connection and sends the data
     */
    public void sendData() {
        new RovUDPsender().send(SeaFarm.ROVIPADDRESS, dataFromGui,
SeaFarm.ROVSENDPORT);
    }

}
```

ROV GUI Controller

```java
/*
 * 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 seafarm;

//import java.nio.ByteBuffer;
import seafarm.RovDatahandler;
import java.util.TimerTask;

/**
 * runs on timertask. checs the functions each run
 *
 * @author Jørgen
 */
public class RovGUIController extends TimerTask {

    private RovDatahandler datahandler;

    public RovGUIController() {

    }

    /**
     * prints the statement :Reques data:"
     */
    @Override
    public void run() {
        System.out.println("Request data");
        this.datahandler.incrementRequestCode();
    }

    /**
     * setting the datahandler to the timertask
     *
     * @param datahandler
     */
    public void setDatahandler(RovDatahandler datahandler) {
        this.datahandler = datahandler;
    }

    /**
```

```
     * checs value if true, and ends the boolean value to datahandler
     *
     * @param value
     */
    public void setFwd(boolean value) {

        if (value) {
            System.out.println("Command : Fwd:, Aktivated thrusters");
            this.datahandler.goFwd();

        } else {
            System.out.println("Command : Released FWD, deAktivated
thrusters");
            this.datahandler.releaseGoFwd();

        }

    }

    /**
     * checs value if true, and ends the boolean value to datahandler
     *
     * @param value
     */
    public void setLeft(boolean value) {
        if (value) {
            System.out.println("Command : LEFT, Aktivated thrusters");
            this.datahandler.goLeft();
        } else {
            System.out.println("Command : Released LEFT, deAktivated
thrusters");
            this.datahandler.releaseGoLeft();
        }
    }

    /**
     * checs value if true, and ends the boolean value to datahandler
     *
     * @param value
     */
    public void setRev(boolean value) {
        if (value) {
            System.out.println("Command : Rev, Aktivated thrusters");
            this.datahandler.goRew();
```

```
        } else {
            System.out.println("Command : released Rev, deAktivated
thrusters");
            this.datahandler.releaseGoRew();
        }
    }

    /**
     * checs value if true, and ends the boolean value to datahandler
     *
     * @param value
     */
    public void setRight(boolean value) {
        if (value) {
            System.out.println("Command : RIGHT, Aktivated thrusters");
            this.datahandler.goRight();
        } else {
            System.out.println("Command : released RIGHT,  deAktivated
thrusters");
            this.datahandler.releaseGoRight();
        }
    }

    /**
     * checs value if true, and ends the boolean value to datahandler
     *
     * @param value
     */
    public void setSlideLeft(boolean value) {
        if (value) {
            System.out.println("Command : SLIDE LEFT, Aktivated
thrusters");
            this.datahandler.setSlideLeft();
        } else {
            System.out.println("Command :released SLIDE LEFT, deAktivated
thrusters");
            this.datahandler.resetSlideLeft();
        }

    }

    /**
     * checs value if true, and ends the boolean value to datahandler
     *
```

```
     * @param value
     */
    public void setSlideRight(boolean value) {
        if (value) {
            System.out.println("Command : SLIDE RIGHT, Activated
thrusters");
            this.datahandler.setSlideRight();
        } else {
            System.out.println("Command : relesaed SLIDE RIGHT, deActivated
thrusters");
            this.datahandler.resetSlideRight();
        }
    }

    /**
     * checs value if true, and ends the boolean value to datahandler
     *
     * @param value
     */
    public void setStart(boolean value) {
        if (value) {
            System.out.println("Command : STARTING ROV");
            this.datahandler.enableROV();
        } else if (!value) {
            System.out.println("Command : STOP ROV");
            this.datahandler.disableROV();
        }
    }

    /**
     * checs value if true, and ends the boolean value to datahandler
     *
     * @param value
     */
    public void setLightValue(int sens) {
        System.out.println("New lightValue = " + sens);
        this.datahandler.setLightValue((byte) sens);
    }

    /**
     * checs value if true, and ends the boolean value to datahandler
     *
     * @param value
     */
```

```
    public void setThrusterValue(int sens) {
        System.out.println("New thrusterValue = " + sens);
        this.datahandler.setThrusterValue((byte) sens);
    }


}
```

ROV UDP Sender

```
/*
 * 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 seafarm;

import java.io.IOException;
import java.net.DatagramPacket;
import java.net.DatagramSocket;
import java.net.InetAddress;
import java.net.SocketException;
import java.util.Arrays;
import java.util.logging.Level;
import java.util.logging.Logger;
/**
 *
 * @author Jørgen
 */
public class RovUDPsender {

    private DatagramSocket clientSocket;

   /**
    * creates new UDP client for sending data
    */
    public RovUDPsender() {
           // nothing to do here
    }

    /*
    * init method
    */
    private void init(){
        try {
            clientSocket = new DatagramSocket();
        }  catch (SocketException ex) {
            Logger.getLogger(RovUDPsender.class.getName()).log(Level.SEVERE
, null, ex);
        }

    }
```

```
   /**
    * Sends data to Rasberry through datagramsocket
    * @param ipAddress  the destination ipadress
    * @param data sending data
    * @param port destination port
    */
    public void send(String ipAddress, byte[] data, int port){
        this.init();
         try {

            DatagramPacket packet = new DatagramPacket(data,
                                        data.length,
                                        InetAddress.getByName(ipAddress),
                                        port);
            clientSocket.send(packet);
            //System.out.println(Arrays.toString(data) + " Fra GUI TIL
RASBERRY ");

        } catch (IOException ex) {
            Logger.getLogger(RovUDPsender.class.getName()).log(Level.SEVERE
, null, ex);
        } finally {
        clientSocket.close();
        }
    }




}
```

ROV UDP receiver

```
/*
 * 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 seafarm;

import seafarm.RovReceiveDataObservable;
import java.io.IOException;
import java.net.*;
import java.util.Arrays;

/**
 * Gets data from Rasberry throu UDP Connection
 * @author Jørgen
 */
public class RovUDPreceiver implements Runnable {

    private RovReceiveDataObservable observer;
    private DatagramSocket receiveSocket;
    private int port;
    private Thread t;


    /**
     * Sets ports for connection
     * @param receiveDataObs
     * @param port
     */
    public RovUDPreceiver(RovReceiveDataObservable receiveDataObs, int
port) {
        this.observer = receiveDataObs;
        this.port = port;
    }


    /**
     * create thread and starts it
     */
    public void start() {
        t = new Thread(this, "UDPReceiverThread");
        t.start();
    }
```

```
    /**
     * getting information on the UDP connection reading the sensor values
sendt from Raspberry
     */
    public void run() {
        if (observer != null) {
            try{
              Thread.sleep(10000);
            }catch (Exception e) {
                System.out.println(e);
            }
            try {

                receiveSocket = new DatagramSocket(port);
                DatagramPacket receivePacket = new DatagramPacket(new
byte[25], 25);

                while (observer.shouldChildOfThisRun()) {
                    receiveSocket.receive(receivePacket);
                    observer.setData(receivePacket.getData());
                    //System.out.println(Arrays.toString(receivePacket.getD
ata()) + " Fra Rasberry");

                }
            } catch (IOException e) {
                System.out.println(e);
            } finally {
                receiveSocket.close();
                System.out.println("receivesocket closed");
            }
        } else {
            System.out.println("receive datahandler not created in
udpreceiver thread");
        }
    }

}
```

UDP Video Receiver

```
/*
 * 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 seafarm;

import java.net.DatagramPacket;
import java.net.DatagramSocket;
import org.opencv.core.Core;
import org.opencv.core.Mat;
import org.opencv.core.MatOfByte;
import org.opencv.imgcodecs.Imgcodecs;

/**
 *
 * @author sigurdolav
 */
public class udpVideoReciever implements Runnable{


    private DatagramSocket receiveSocket;
    private int port;
    private Mat matImage;
    Thread t ;
    boolean exit = true;

    public udpVideoReciever(int port) {
        this.port = port;// Main.VIDEOPORT;
        matImage = null;
    }

    public void start() {
        t = new Thread(this);
        t.start();
    }

    public void run(){

        try {
        this.receiveSocket = new DatagramSocket(port);
        } catch (Exception e) {
        }
```

```
        byte[] buffer = new byte[65507];
        DatagramPacket receivePacket = new DatagramPacket(buffer,
buffer.length);


        while (exit) {
            try {
            receiveSocket.receive(receivePacket);
            } catch (Exception e) {
            }
            byte[] receivedImage = receivePacket.getData();
            MatOfByte mob = new MatOfByte(receivedImage);
            Mat img = Imgcodecs.imdecode(mob, Imgcodecs.IMREAD_UNCHANGED);

            this.matImage = img;

                }

    }


    public Mat getImage() {
        return this.matImage;

    }

    public void stop(){
        exit = false;
    }





    /*
        public static void main(String[] args) {

        System.loadLibrary(Core.NATIVE_LIBRARY_NAME);
        System.out.println(" Fra Rasberry");
        udpVideoReciever test= new udpVideoReciever();
        test.StartStream();
    }
     */
```

```
}
```

ROV Receive Data Observer

```
/*
 * 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 seafarm
        ;

import java.math.BigInteger;
import java.util.Arrays;
import java.util.Observable;
import java.util.Observer;

/**
 * the observer to check the data received from Raspberry
 *
 * @author Jørgen
 */
public class RovReceiveDataObservable extends Observable {

    float temperature=0;
    float descTemp=0;
    float pressure=0;
    float descPressure=0;
    float depth=0;
    float descDepth=0;
    float oxygen=0;
    float descOxygen=0;
    int waterlevel=0;
    float roll=0;
    float descRoll;
    float pitch;
    float descPitch;
    int heading=0;
    float tempInROV=0;
    float descTempInROV;
    float xAccMeter;
    float descxAccmeter;
    float yAccMeter;
    float descYAccmeter;
    float zAccMeter;
    float desczAccmeter;
    float tensiometer;
```

```
    float descTensiometer;

    //private final Datahandler datahandler;
    public RovReceiveDataObservable() {

    }

    /**
     * adds the observer to the object
     *
     * @param o
     */
    @Override
    public synchronized void addObserver(Observer o) {
        super.addObserver(o); //To change body of generated methods, choose
Tools | Templates.
    }

    /**
     * Sets the receiving data to the fields, and notifies observers.
     *
     * @param data
     */
    public void setData(byte[] data) {
        // check if the array is of the same length and the requestcode has
changed
        if (data.length == 25) {

            descTemp = data[2];
            temperature = (data[1] + (descTemp / 100));
            descPressure = data[4];
            pressure = (data[3] + (descPressure / 100));
            descDepth = data[6];
            depth = (data[5] + (descDepth / 100));
            descOxygen = data[8];
            oxygen = (data[7] + (descOxygen / 100));
            waterlevel = data[9];
            descRoll = data[11];
            roll = (data[10] + (descRoll / 100));
            descPitch = data[13];
            pitch = (data[12] + (descPitch / 100));
            heading = data[14];
            descTempInROV = data[16];
            tempInROV = (data[15] + (descTempInROV / 100));
```

```
            descxAccmeter = data[18];
            xAccMeter = (data[17] + (descxAccmeter / 100));
            descYAccmeter = data[20];
            yAccMeter = (data[19] + (descYAccmeter / 100));
            desczAccmeter = data[22];
            zAccMeter = (data[21] + (desczAccmeter / 100));
            descTensiometer = data[24];
            tensiometer = (data[23] + (descTensiometer / 100));

            setChanged();
            notifyObservers();
        }
    }

    // Gett all values of the sensores
    public float getDepth() {
        return depth;
    }

    public float getPressure() {
        return pressure;
    }

    public float getTemperature() {
        return temperature;
    }

    public float getOxygen() {
        return oxygen;
    }

    public int getWaterlevel() {
        return waterlevel;
    }

    public float getRoll() {
        return roll;
    }

    public float getPitch() {
        return pitch;
    }

    public int getHeading() {
```

```
        return heading;
    }

    public float getTempInROV() {
        return tempInROV;
    }

    public float getxAccMeter() {
        return xAccMeter;
    }

    public float getyAccMeter() {
        return yAccMeter;
    }

    public float getzAccMeter() {
        return zAccMeter;
    }

    public float getTensiometer() {
        return tensiometer;
    }

    public boolean shouldChildOfThisRun() {
        //return datahandler.shouldThreadRun();
        return true;
    }

}
```

Auto Mode

```
/*
 * 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 seafarm;

/**
 *
 * @author Platform
 */
public class autoMode {

private double currentLat=0;
private double currentLong=0;
private double acceptedLongOffset=0.0002;
private double acceptedLatOffset=0.0002;

public autoMode(){

}

public boolean shouldGetNewPos(float currentPlLat,float currentPlLong,
        double currentSetLong,double currentSetLat ){

    double longdiff= Math.abs(currentPlLong-currentSetLong);
    double latdiff= Math.abs(currentPlLat-currentSetLat);
    boolean longOk=false;
    boolean latOk=false;
    System.out.println("difflong: "+longdiff);
    System.out.println("difflat: "+latdiff);
    if (latdiff<acceptedLatOffset){
        latOk=true;
    }

    if (longdiff<acceptedLongOffset){
        longOk=true;
    }

    if (longOk && latOk){
        return true;
    }
    else {
```

```
        return false;
    }

}


}
```

M PLC Source Code

Project Documentation

File:	Platform_Application_app.ecp
Date:	5/20/2019
Profile:	e!COCKPIT

Table of Contents

1 Device: Generic_MODBUS_Master

Information

Name:	ThirdParty Modbus Master
Vendor:	WAGO
Categories:	
Type:	32808
ID:	1007 8211
Version:	1.0.0.1
Order number:	GenericModbusMaster
Description:	ThirdParty Modbus Master

2 Device: Generic_MODBUS_Master_1

Information

Name:	ThirdParty Modbus Master
Vendor:	WAGO
Categories:	
Type:	32808
ID:	1007 8211
Version:	1.0.0.1
Order number:	GenericModbusMaster
Description:	ThirdParty Modbus Master

3 Device: PFC100_2ETH_ECO

Users and Groups

Users:

Groups:

Access Rights

View
Modify
Execute
Add/remove children

Symbol Rights

Parameters

Parameters:

Name:	Type:	Value:	Default Value:	Unit:	Description:
Processor Load Lower Limit	DWORD	80	80		
Processor Load Upper Limit	DWORD	90	90		
Processor Load Processor Share	DWORD	90	90		
Processor Load Should Throw ProcessorLoadWatchdog_Exception	bool	FALSE	FALSE		

Information

Name:	750-8100 PFC100 CS 2ETH ECO
Vendor:	WAGO
Categories:	PLCs
Type:	4096
ID:	1006 1101
Version:	5.11.1.16
Order number:	0750-8100
Description:	Programmable Ethernet controller

3.1 PLC Logic: Plc Logic

3.1.1 Application: Application

3.1.1.1 Folder: FunctionBlocks

3.1.1.1.1 POU: FB_CalcAngle

```
FUNCTION_BLOCK  FB_CalcAngle
VAR_INPUT
    LatA :  LREAL ;
    LatB :  LREAL ;
    LonA :  LREAL ;
    LonB :  LREAL ;
END_VAR
VAR_OUTPUT

    Angle :  LREAL ;

```

```
END_VAR
VAR
    y :  LREAL ;
    x :  LREAL ;

    bearing :  LREAL ;

END_VAR

```

```
y  :=  (WagoAppMath . sin_L ( phi :=  LonB  -  LonA ) )  *  WagoAppMath . cos_L ( phi :=
LatB ) ;
x  :=  WagoAppMath . cos_L ( phi :=  LatA )  *  WagoAppMath . sin_L ( phi :=  LatB )  -
WagoAppMath . sin_L ( phi :=  LatA )  *  WagoAppMath . cos_L ( phi :=  LatB )  *
WagoAppMath . cos_L ( phi :=  LonB  -  LonA ) ;
bearing  :=  WagoAppMath . arcTan2 ( y := y  ,  x :=  x ) ;
//Angle2  := WagoAppMath.radiantToAngle(lrRadiant:=  bearing);
Angle  :=  WagoAppMath . angleToDegree_L ( phi :=  bearing ) ;


```

3.1.1.1.2 POU: FB_CalcDistance

```
FUNCTION_BLOCK  FB_CalcDistance
VAR_INPUT
    LatA : LREAL ;
    LatB : LREAL ;
    LonA : LREAL ;
    LonB : LREAL ;
END_VAR
VAR_OUTPUT
    Distance : LREAL ;
END_VAR
VAR
    dLat :  LREAL ;
    dLon :  LREAL ;
    lat1 :  LREAL ;
    lat2 :  LREAL ;
    a :  LREAL ;
    c :  LREAL ;
    dist2 :  LREAL ;

    x : LREAL := 0 ;
    Radius  :  LREAL  := 6372.795477598 ;

END_VAR

```

```
x :=  ( ( WagoAppMath . sin_L ( phi :=  LatA )  *  WagoAppMath . sin_L ( phi :=  LatB ) )
+  ( WagoAppMath . cos_L ( phi :=  LatA )  *  WagoAppMath . cos_L ( phi :=  LatB )  *
```

```
WagoAppMath . cos_L ( phi :=  ( LonA - LonB ) ) ) ) ;
distance  :=  Radius  *  WagoAppMath . arcCos ( WagoAppMath . angleToRadiant_L ( phi
:=  x ) ) ;

//dLat  := WagoAppMath.angleToRadiant_L(phi:=  LatB - LatA);
//dLon  := WagoAppMath.angleToRadiant_L(phi:=  LonB - LonA);
//lat1  := WagoAppMath.angleToRadiant_L(phi:=  LatA);
//lat2  := WagoAppMath.angleToRadiant_L(phi:=  LatB);
//a  := (WagoAppMath.sin_L(phi:=  dLat)/2)*(WagoAppMath.sin_L(phi:=  dLat)/2)  +
(WagoAppMath.sin_L(phi:=  dLon)/2)*(WagoAppMath.sin_L(phi:=  dLon)/2)  *
(WagoAppMath.cos_L(phi:=  lat1))*  (WagoAppMath.cos_L(phi:=  lat2));
//c  := 2  *  (WagoAppMath.sin_L(phi:=  WagoAppMath.sqrt_L(x:=  a)));
//dist2  := Radius  *  c;

```

3.1.1.1.3 POU: FB_CheckPitchRoll

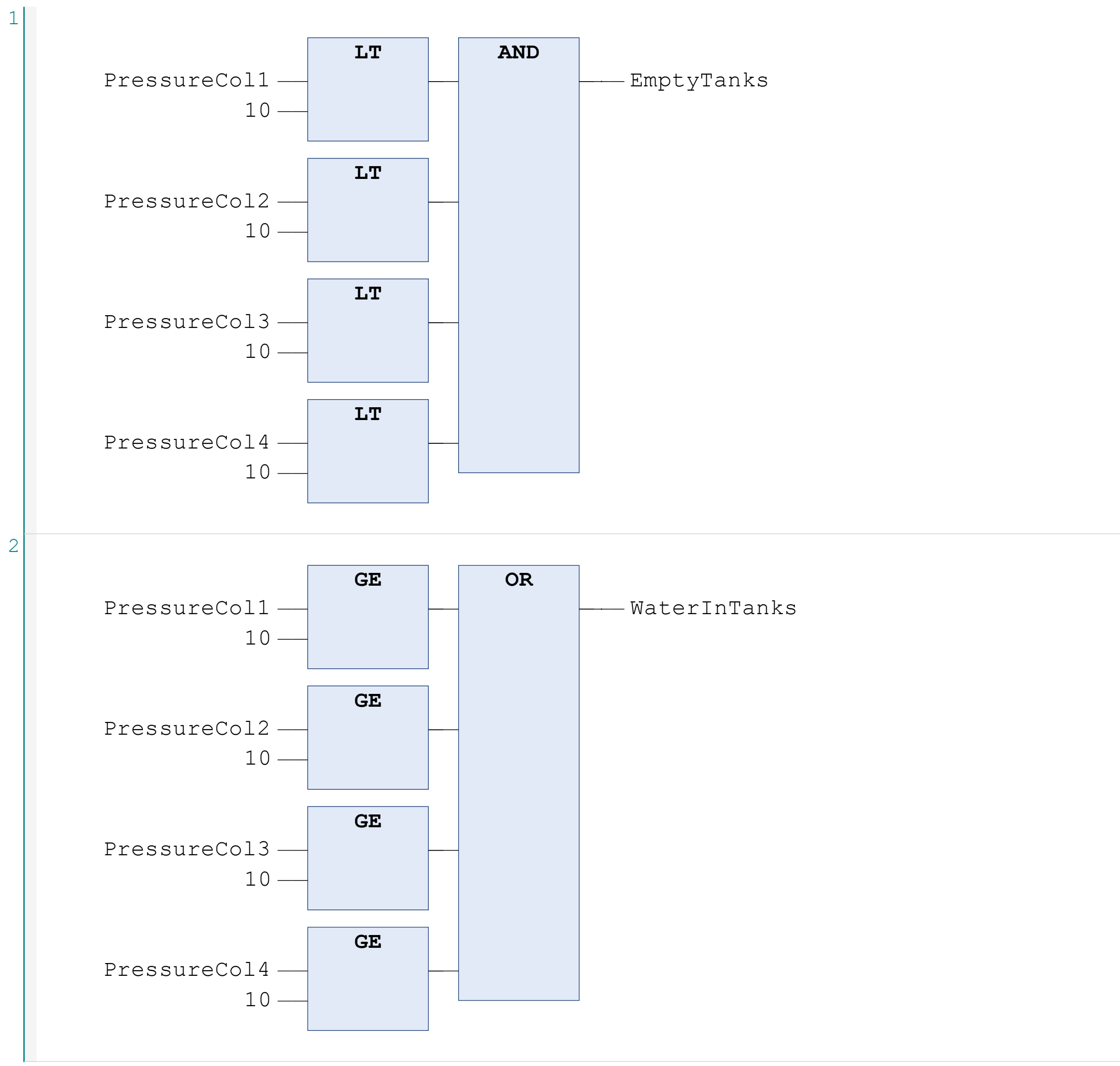
```
FUNCTION_BLOCK  FB_CheckPitchRoll
VAR_INPUT
Draft : REAL ;
offsetPitch : REAL ;
offsetRoll : REAL ;
offSetDraft : REAL ;
gyroRoll : REAL ;
gyroPitch : REAL ;
pitchSetPoint : REAL ;
rollSetpoint : REAL ;
draftSetpoint : REAL ;

END_VAR
VAR_OUTPUT
    corrDraft : BOOL ;
    corrPitch : BOOL ;
    corrRoll : BOOL ;

END_VAR
VAR
    lcCorrRoll : BOOL ;
    lcCorrPitch : BOOL ;
    lcCorrDraft : BOOL ;
END_VAR

```

```
lcCorrRoll := FALSE ;
lcCorrPitch := FALSE ;
lcCorrDraft := FALSE ;
IF  ( ( ( Draft  >  draftSetpoint  +  offSetDraft )  OR  ( Draft  <  draftSetpoint  -
offSetDraft ) )  )  THEN
        lcCorrDraft  :=  TRUE ;
END_IF
```

```
IF ( ( ( gyroRoll > simGVL . simSetPoint + offsetRoll ) OR ( gyroRoll < rollSetpoint - offsetRoll ) ) AND NOT ( lcCorrDraft OR lcCorrPitch ) ) THEN
        lcCorrRoll := TRUE ;
END_IF

IF ( ( ( gyroPitch > pitchSetPoint + offsetPitch ) OR ( gyroPitch < pitchSetPoint - offsetPitch ) ) AND NOT ( lcCorrDraft OR lcCorrRoll ) ) THEN
        lcCorrPitch := TRUE ;
END_IF
corrRoll := lcCorrRoll ;
corrPitch := lcCorrPitch ;
corrDraft := lcCorrDraft ;

//IF (((simGVL.simDraftAvg > simGVL.simDraftSetPoint + 5) OR (simGVL.simDraftAvg < simGVL.simDraftSetPoint -5)) AND NOT (corrRoll OR corrPitch)) THEN
//              corrDraft := TRUE;
//END_IF

//IF (((simGVL.simRollAvg > simGVL.simSetPoint + 5) OR (simGVL.simRollAvg < simGVL.simDraftSetPoint -5)) AND NOT (corrDraft OR corrPitch)) THEN
//              corrRoll := TRUE;
//END_IF

//IF (((simGVL.simPitchAvg > simGVL.simSetPoint + 5) OR (simGVL.simPitchAvg < simGVL.simDraftSetPoint -5))AND NOT (corrDraft OR corrRoll)) THEN
//              corrPitch := TRUE;
//END_IF

```

3.1.1.1.4 POU: FB_CheckWatertanks

```
FUNCTION_BLOCK FB_CheckWatertanks
VAR_INPUT
    PressureCol1 : REAL ;
    PressureCol2 : REAL ;
    PressureCol3 : REAL ;
    PressureCol4 : REAL ;
END_VAR

VAR_OUTPUT
    WaterInTanks : BOOL ;
    EmptyTanks : BOOL ;

END_VAR

VAR
END_VAR

```


1
LT
AND
PressureCol1
10
EmptyTanks
LT
PressureCol2
10
LT
PressureCol3
10
LT
PressureCol4
10
2
GE
OR
PressureCol1
10
WaterInTanks
GE
PressureCol2
10
GE
PressureCol3
10
GE
PressureCol4
10

3.1.1.1.5 POU: FB_EmptyTanks

```
FUNCTION_BLOCK  FB_EmptyTanks
VAR_INPUT
    Enable :  BOOL ;
    PressureCol1 :  REAL ;
    PressureCol2 :  REAL ;
    PressureCol3 :  REAL ;
    PressureCol4 :  REAL ;
END_VAR
VAR_OUTPUT
END_VAR
VAR
END_VAR

```

```
IF  ( Enable )  THEN
    IF  ( PressureCol1  >=  10 )  THEN
        IoConfig_Globals_Mapping . Col1Out  :=  TRUE ;
        ELSE
            IoConfig_Globals_Mapping . Col1Out  :=  FALSE ;
        END_IF

        IF  ( PressureCol2  >=  10 )  THEN
        IoConfig_Globals_Mapping . Col2Out  :=  TRUE ;
        ELSE
            IoConfig_Globals_Mapping . Col2Out  :=  FALSE ;
        END_IF

        IF  ( PressureCol3  >=  10 )  THEN
        IoConfig_Globals_Mapping . Col3Out  :=  TRUE ;
        ELSE
            IoConfig_Globals_Mapping . Col3Out  :=  FALSE ;
        END_IF

        IF  ( PressureCol4  >=  10 )  THEN
        IoConfig_Globals_Mapping . Col4Out  :=  TRUE ;
        ELSE
            IoConfig_Globals_Mapping . Col4Out  :=  FALSE ;
        END_IF

END_IF

```

3.1.1.1.6 POU: FB_Encoder

```
FUNCTION_BLOCK FB_Encoder
VAR_INPUT
    Inngang1 : BYTE ;
    Inngang2 : BYTE ;
    InngangZ : BOOL ;
    underflow  : BOOL ;
    overflow  : BOOL ;
END_VAR
VAR_OUTPUT
    counter : UINT ;
    RoundCounter  : int ;
END_VAR
VAR
    encoderValue : UINT ;
    encoderValue0 : UINT ;
    ///forskjell mellom encoder value og value 0
    dx : INT ;
    zeroIsFound : BOOL ;


    //positiv flanke over og underflow
    overflowPF  : BOOL ;
    underflowPF  : BOOL ;
    //roundcounter var
    counterVariable : REAL ;
    compareVar : REAL ;
END_VAR

```

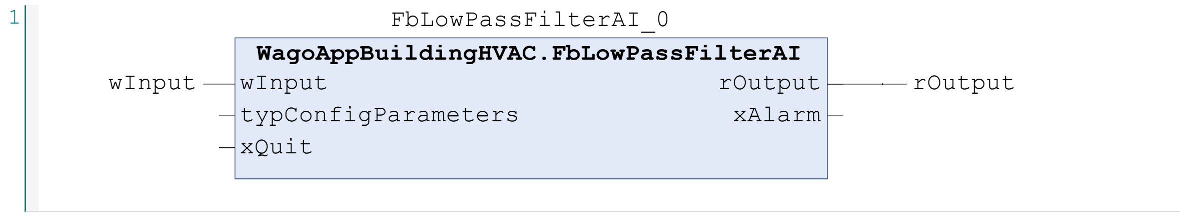
```
//Digital inputs
(* må deklarere inngangene fra enkoderen*)

//settup for zero at round counter



encoderValue  := Inngang1 + 256 * Inngang2 ;
zeroIsFound  := InngangZ OR zeroIsFound ;

//test 4096
//Assumption: 8000 steps for 1 full rotation (1-8000)
IF zeroIsFound THEN
    IF InngangZ THEN
        counter  := 2048 ;
    END_IF

    IF underflow AND NOT underflowPF THEN
```

```
        dx  :=  - ( encoderValue0  +  ( 65535 - encoderValue ) + 1 ) ;

        ELSIF  overflow  AND  NOT  overflowPF  THEN
            dx  :=   + ( encoderValue  +  ( 65535 - encoderValue0 ) + 1 ) ; ;
        ELSE
        dx  :=  encoderValue0  -  encoderValue  ;
        END_IF


        //if counter is going over the max lim or bellow 3 then count up and down the rotation var
        counterVariable := counter  +  dx  +  4096 ;
        compareVar  :=  counterVariable / 4096 ;


        IF  ( compareVar  >=  2 )  THEN
            RoundCounter  :=  RoundCounter  +  1 ;
        END_IF


        IF  ( ( counter  >  0 )  AND  ( compareVar  <=  1 ) )  THEN
                RoundCounter  :=  RoundCounter  -  1 ;


        END_IF

        //conter 1-4096
        counter  :=  ( counter  +  dx  +  4096 )  MOD  4096 ;

    END_IF

    encoderValue0  :=  encoderValue ;
    overflowPF := overflow ;
    underflowPF := underflow ;

```

3.1.1.1.7 POU: FB_Filter

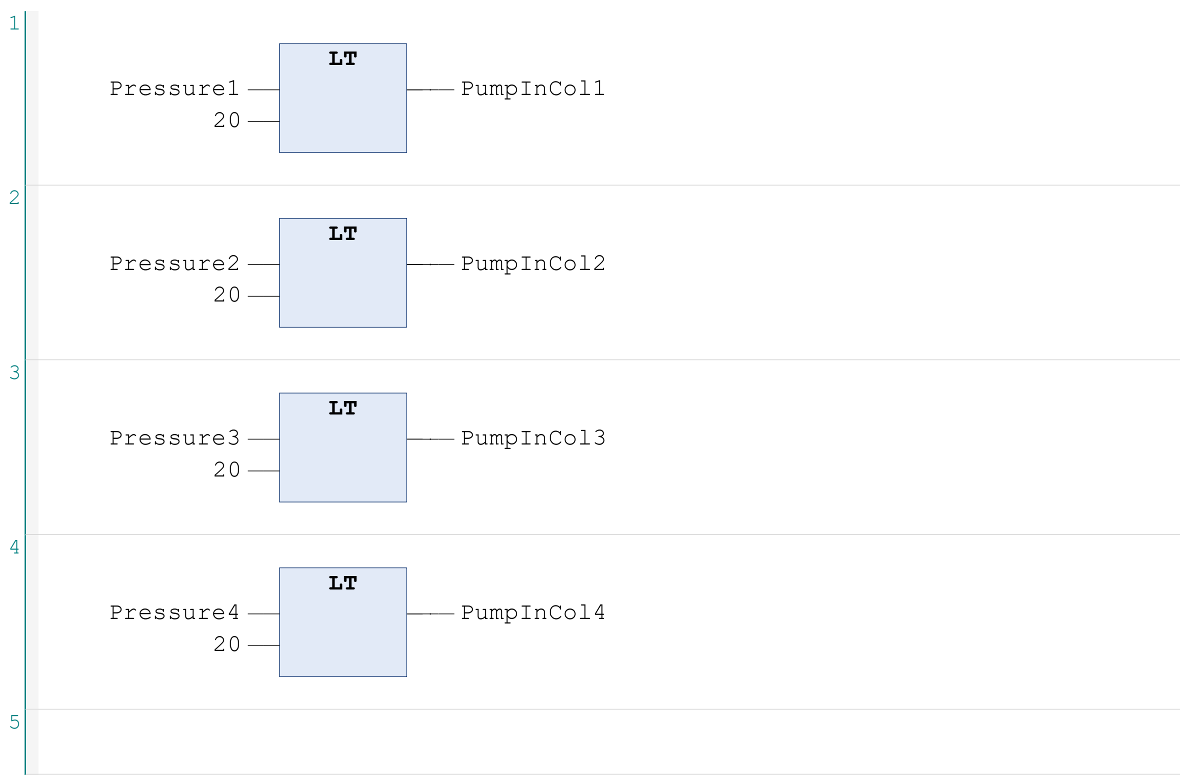
```
    FUNCTION_BLOCK  FB_Filter
    VAR_INPUT
        wInput :  WORD ;
    END_VAR
    VAR_OUTPUT
        rOutput :  REAL ;
    END_VAR
    VAR
        FbLowPassFilterAI_0  :  WagoAppBuildingHVAC . FbLowPassFilterAI ;
        FbLowPassFilter_0  :  WagoAppBuildingHVAC . FbLowPassFilter ;

    END_VAR

```

3.1.1.1.7 POU: FB_Filter



3.1.1.1.8 POU: FB_PumpInWater

```
FUNCTION_BLOCK  FB_PumpInWater
VAR_INPUT
    Pressure1 :  REAL ;
    Pressure2 :  REAL ;
    Pressure3 :  REAL ;
    Pressure4 :  REAL ;
END_VAR
VAR_OUTPUT
    PumpInCol1 :  BOOL ;
    PumpInCol2 :  BOOL ;
    PumpInCol3 :  BOOL ;
    PumpInCol4 :  BOOL ;

END_VAR
VAR
END_VAR
```



3.1.1.1.9 POU: FB_Simulate_ThrusterDir

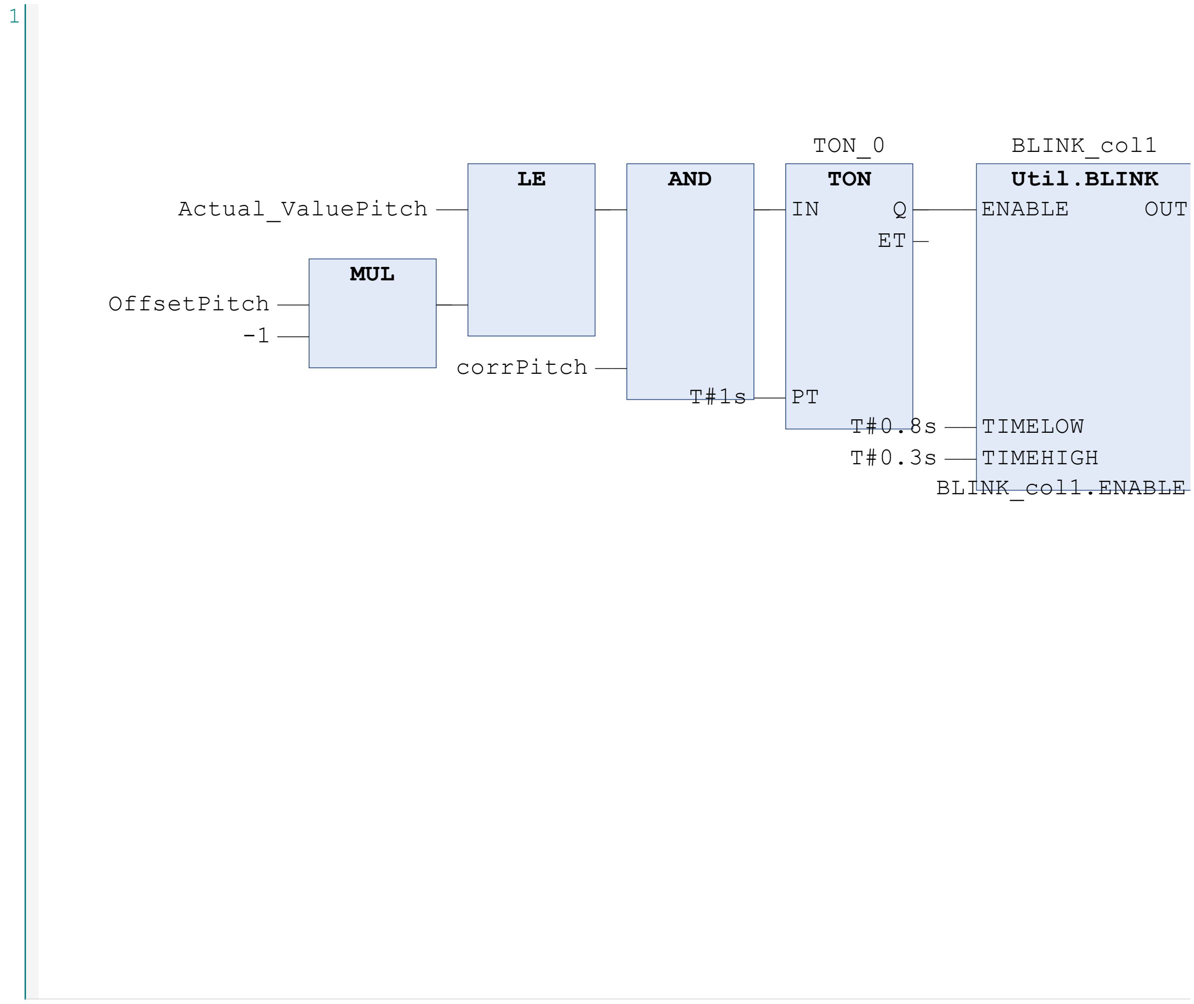
```
FUNCTION_BLOCK  FB_Simulate_ThrusterDir
VAR_INPUT
    Thruster_Value  :  REAL ;
END_VAR
VAR_OUTPUT
END_VAR
VAR
END_VAR

```

```
IF  Thruster_Value  <  5450  THEN
    simGVL . simThrustRight  :=  TRUE ;
    ELSE
            simGVL . simThrustRight  :=  FALSE ;
END_IF

IF  Thruster_Value  >  5550  THEN
    simGVL . simThrustLeft  :=  TRUE ;
    ELSE
            simGVL . simThrustLeft  :=  FALSE ;
END_IF

```

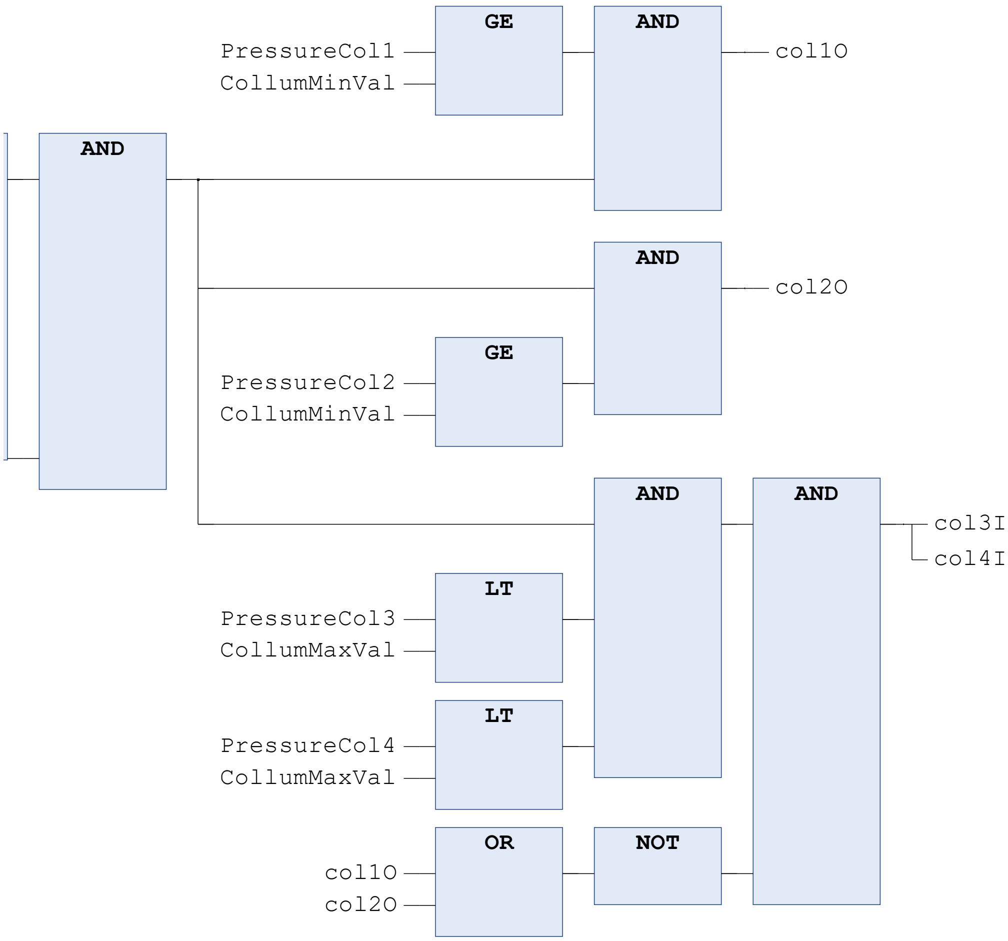
3.1.1.1.10 POU: FB_Stabilization_Pitch

```
FUNCTION_BLOCK  FB_Stabilization_Pitch
VAR_INPUT
    corrPitch : BOOL ;
    Actual_ValuePitch :  REAL ;
    CollumMaxVal :  REAL ;
    CollumMinVal :  REAL ;
    OffsetPitch :  REAL ;
    PressureCol1 :  REAL ;
    PressureCol2 :  REAL ;
    PressureCol3 :  REAL ;
    PressureCol4 :  REAL ;

END_VAR


VAR_OUTPUT
col1O : BOOL ;
col1I : BOOL ;
col2O : BOOL ;
col2I : BOOL ;
col3O : BOOL ;
col3I : BOOL ;
col4O : BOOL ;
```

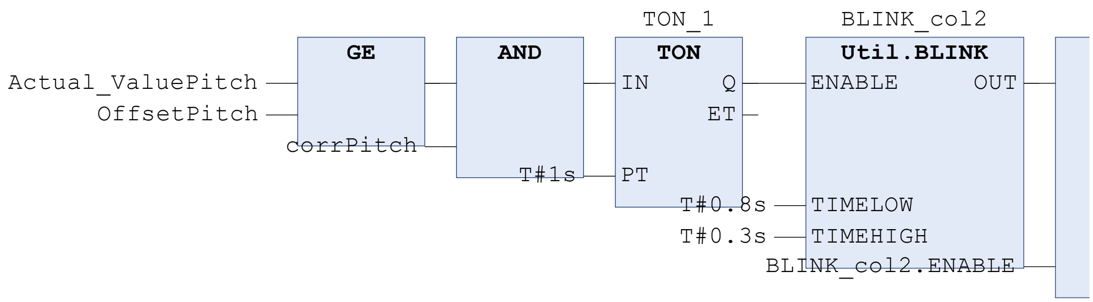
```
    col4I : BOOL ;
    END_VAR


    VAR
        OffsetMinus :  REAL ;
        OffsetPlus :  REAL ;
        TON_0 :  TON ;
        TON_1 :  TON ;
        JobDone_1 :  BOOL ;
        JobDone_2 :  BOOL ;
        BLINK_col1 : Util . BLINK ;
        BLINK_col2 : Util . BLINK ;
    END_VAR

```


1
TON_0
BLINK_col1
LE
AND
TON
Util.BLINK
Actual_ValuePitch
IN
Q
ENABLE
OUT
ET
MUL
OffsetPitch
-1
corrPitch
T#1s
PT
T#0.8s
TIMELOW
T#0.3s
TIMEHIGH
BLINK_col1.ENABLE

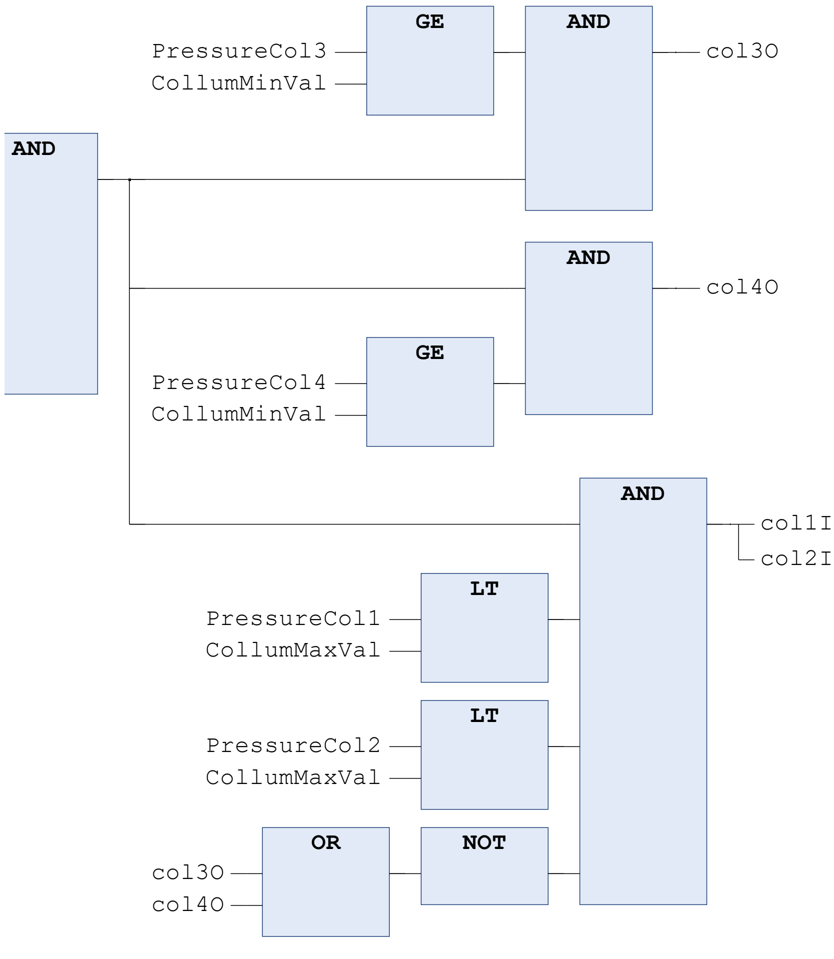
3.1.1.1.10 POU: FB_Stabilization_Pitch



2

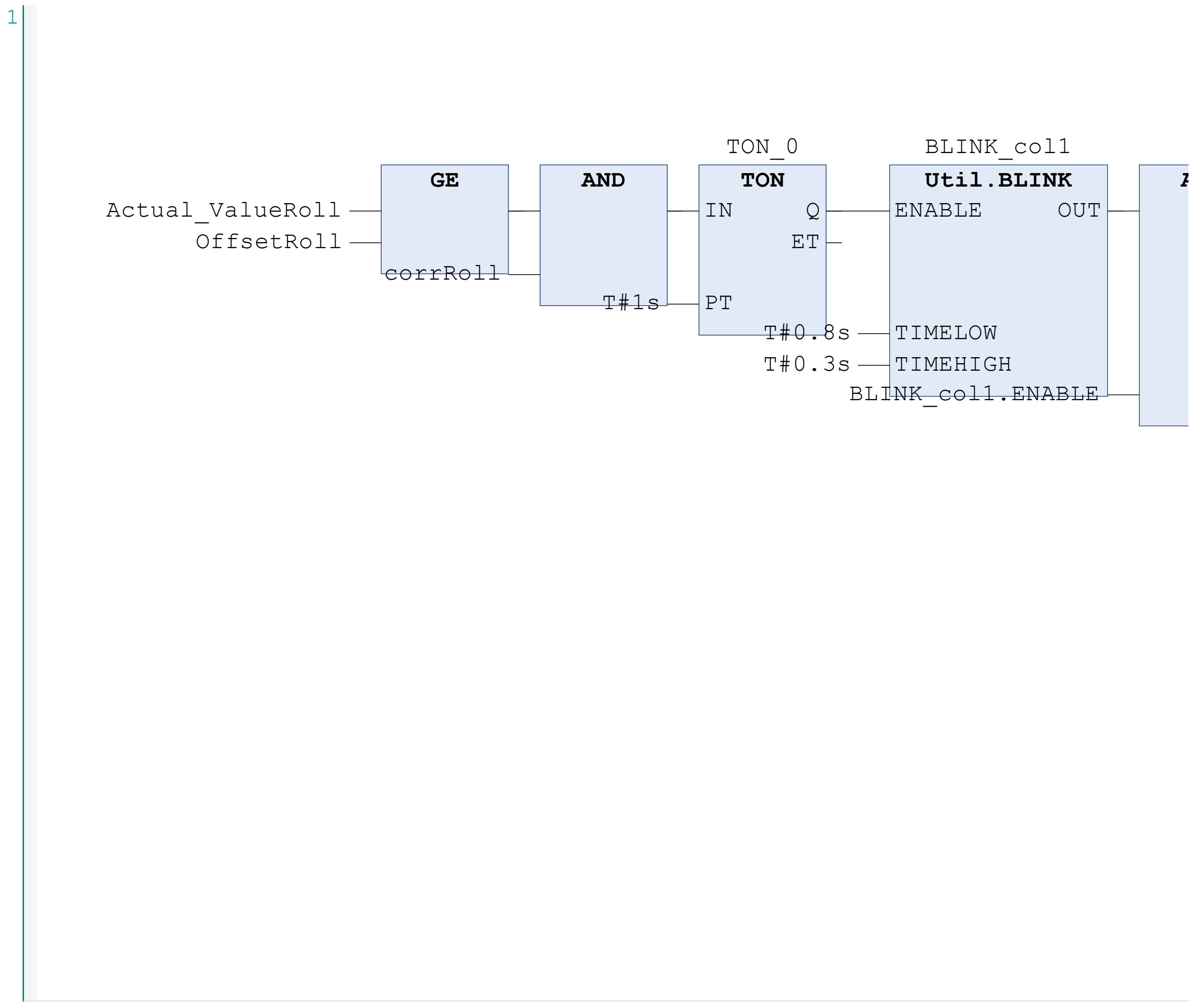


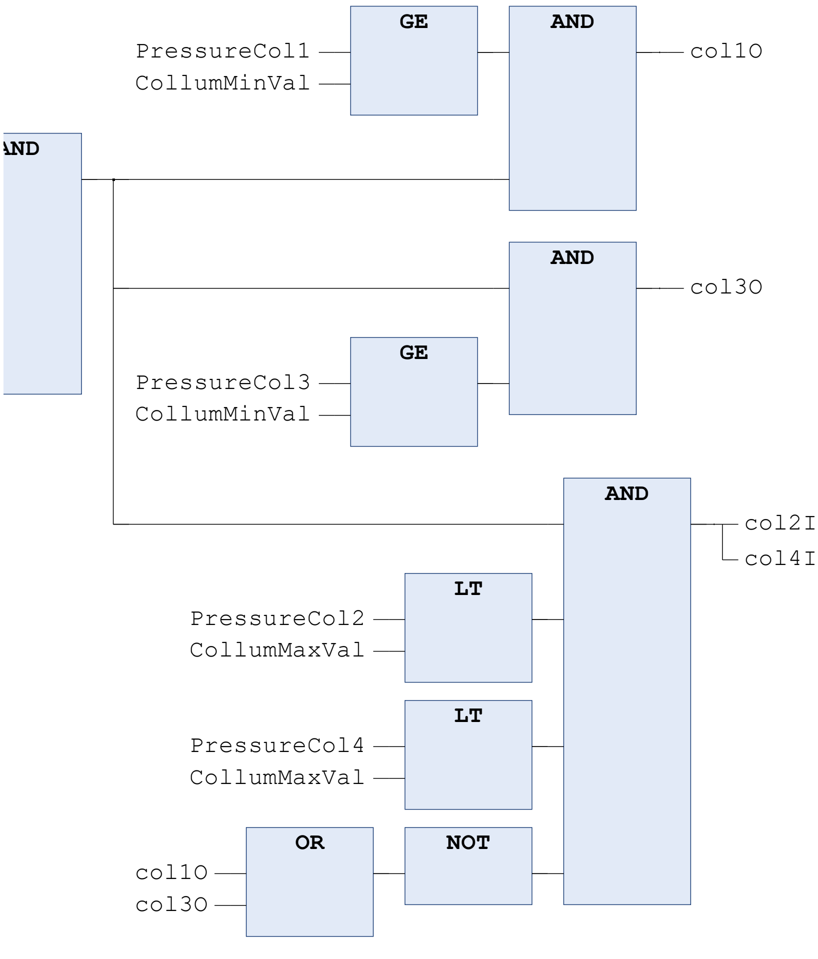
TON_1
BLINK_col2
GE
AND
TON
Util.BLINK
Actual_ValuePitch
OffsetPitch
corrPitch
T#1s
IN
Q
ET
PT
ENABLE
OUT
T#0.8s
T#0.3s
TIMELOW
TIMEHIGH
BLINK_col2.ENABLE


GE
AND
PressureCol3
CollumMinVal
col3O
AND
AND
col4O
GE
PressureCol4
CollumMinVal
AND
col1I
col2I
LT
PressureCol1
CollumMaxVal
LT
PressureCol2
CollumMaxVal
OR
NOT
col3O
col4O

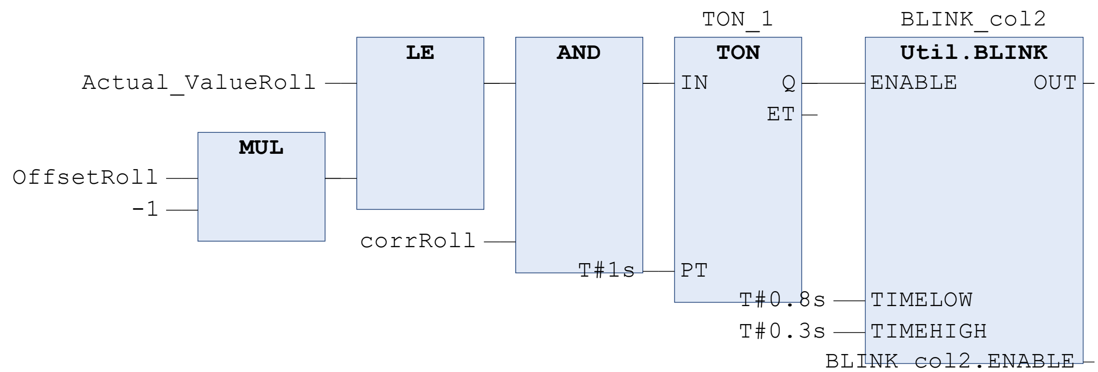
3.1.1.1.11 POU: FB_Stabilization_Roll_1

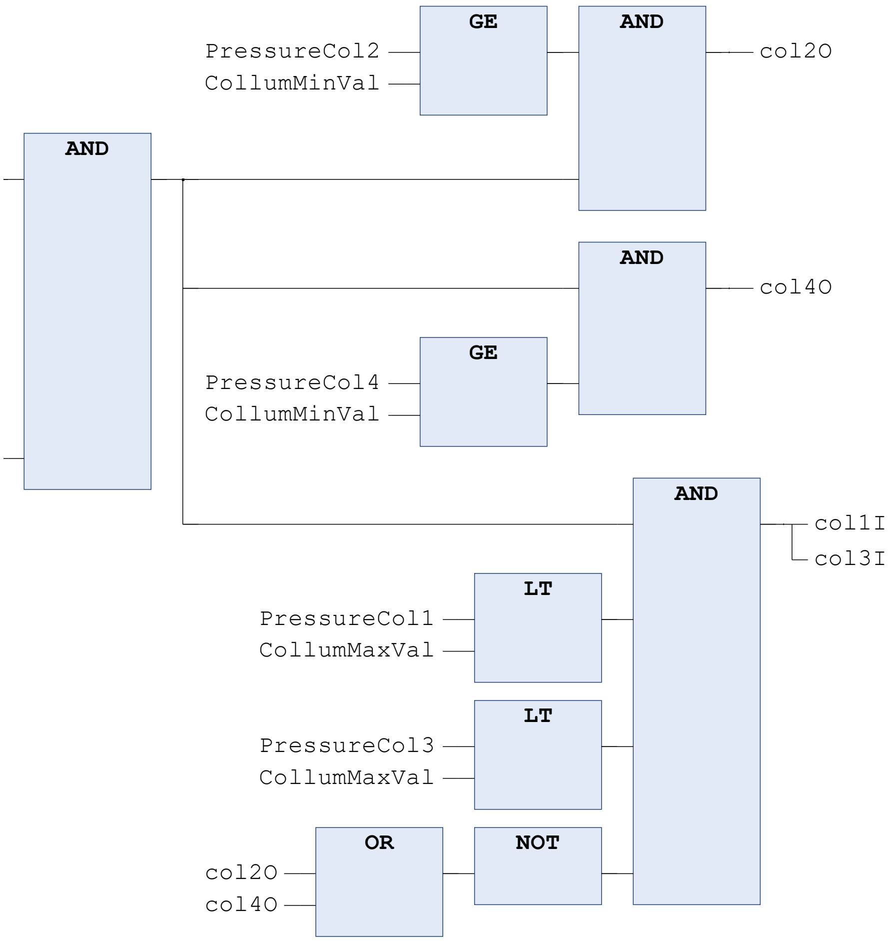
```
FUNCTION_BLOCK  FB_Stabilization_Roll_1
VAR_INPUT
    corrRoll : BOOL ;
    Actual_ValueRoll :  REAL ;
    CollumMaxVal :  REAL ;
    CollumMinVal :  REAL ;
    OffsetRoll :  REAL ;
    PressureCol1 :  REAL ;
    PressureCol2 :  REAL ;
    PressureCol3 :  REAL ;
    PressureCol4 :  REAL ;
END_VAR


VAR_OUTPUT
col1O : BOOL ;
col1I : BOOL ;
col2O : BOOL ;
col2I : BOOL ;
col3O : BOOL ;
col3I : BOOL ;
col4O : BOOL ;
col4I : BOOL ;
END_VAR


VAR
    OffsetMinus :  REAL ;
    OffsetPlus :  REAL ;
    TON_0 :  TON ;
    TON_1 :  TON ;
    JobDone_1 :  BOOL ;
    blink1En :  BOOL ;
    blink3En :  BOOL ;
    blink2En :  BOOL ;
    blink4En :  BOOL ;
    BLINK_col1 :  Util . BLINK ;
    BLINK_col2 :  Util . BLINK ;

END_VAR

```


1
TON_0
BLINK_col1
GE
AND
TON
Util.BLINK
Actual_ValueRoll
OffsetRoll
corrRoll
T#1s
IN
Q
ET
PT
ENABLE
OUT
T#0.8s
TIMELOW
T#0.3s
TIMEHIGH
BLINK_col1.ENABLE


GE
AND
PressureCol1
CollumMinVal
col1O
AND
AND
col3O
GE
PressureCol3
CollumMinVal
AND
col2I
col4I
LT
PressureCol2
CollumMaxVal
LT
PressureCol4
CollumMaxVal
OR
NOT
col1O
col3O

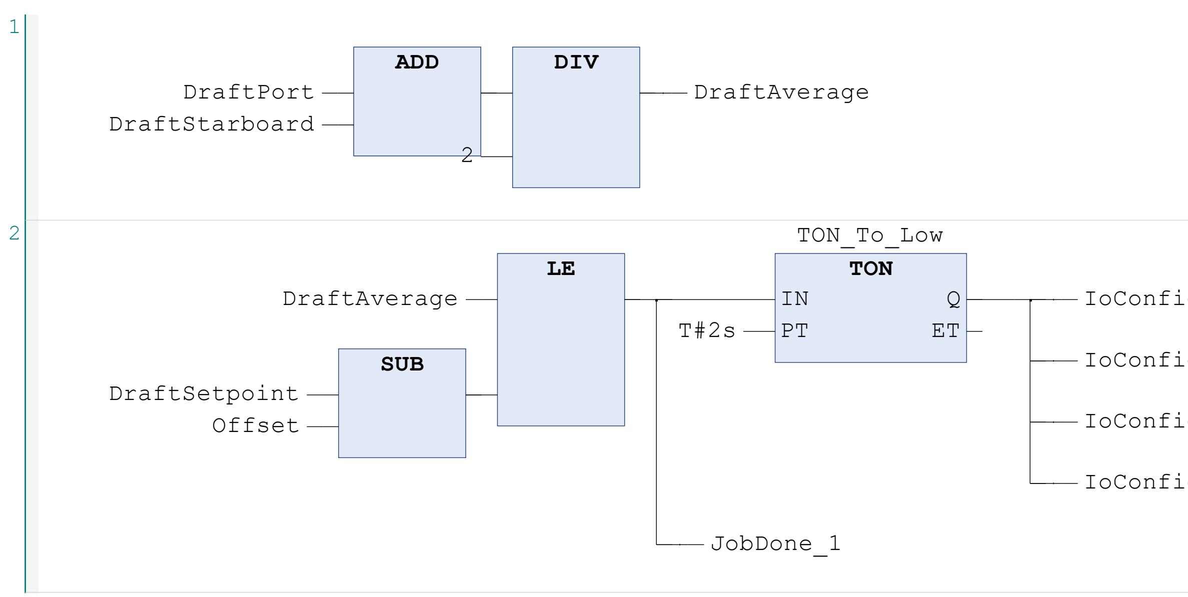
2


TON_1
BLINK_col2
LE
AND
TON
Util.BLINK
Actual_ValueRoll
IN
Q
ENABLE
OUT
ET
MUL
OffsetRoll
-1
corrRoll
T#1s
PT
T#0.8s
TIMELOW
T#0.3s
TIMEHIGH
BLINK_col2.ENABLE

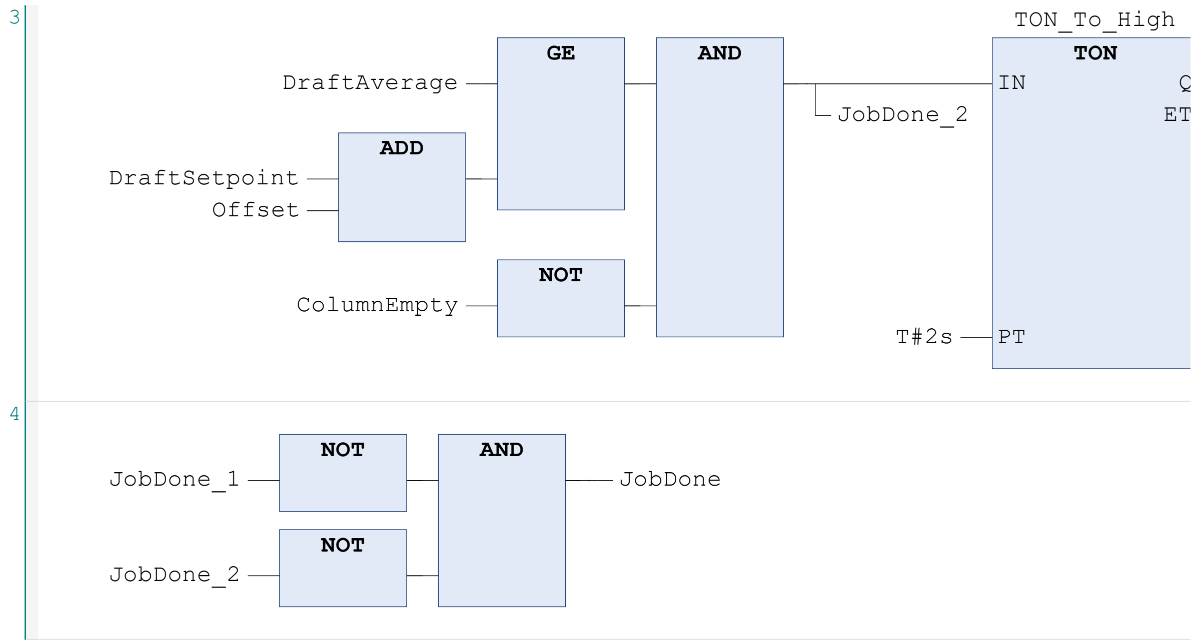

GE
AND
PressureCol2
CollumMinVal
col2O
AND
AND
col4O
GE
PressureCol4
CollumMinVal
AND
col1I
col3I
LT
PressureCol1
CollumMaxVal
LT
PressureCol3
CollumMaxVal
OR
NOT
col2O
col4O

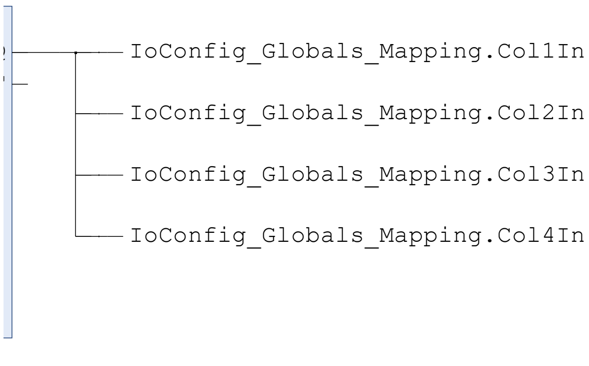
3.1.1.1.12 POU: FB_Stablilization_Draft

```
FUNCTION_BLOCK  FB_Stablilization_Draft
VAR_INPUT
    DraftPort :  REAL ;
    DraftStarboard :  REAL ;
    DraftSetpoint :  REAL ;
    Offset :  REAL ;
END_VAR


VAR_OUTPUT

    JobDone : BOOL ;

END_VAR

VAR

    DraftAverage :  REAL ;
    TON_To_Low :  TON ;
    TON_To_High :  TON ;
    ColumnEmpty :  BOOL ;
    JobDone_1 :  BOOL ;
    JobDone_2 :  BOOL ;
END_VAR

```



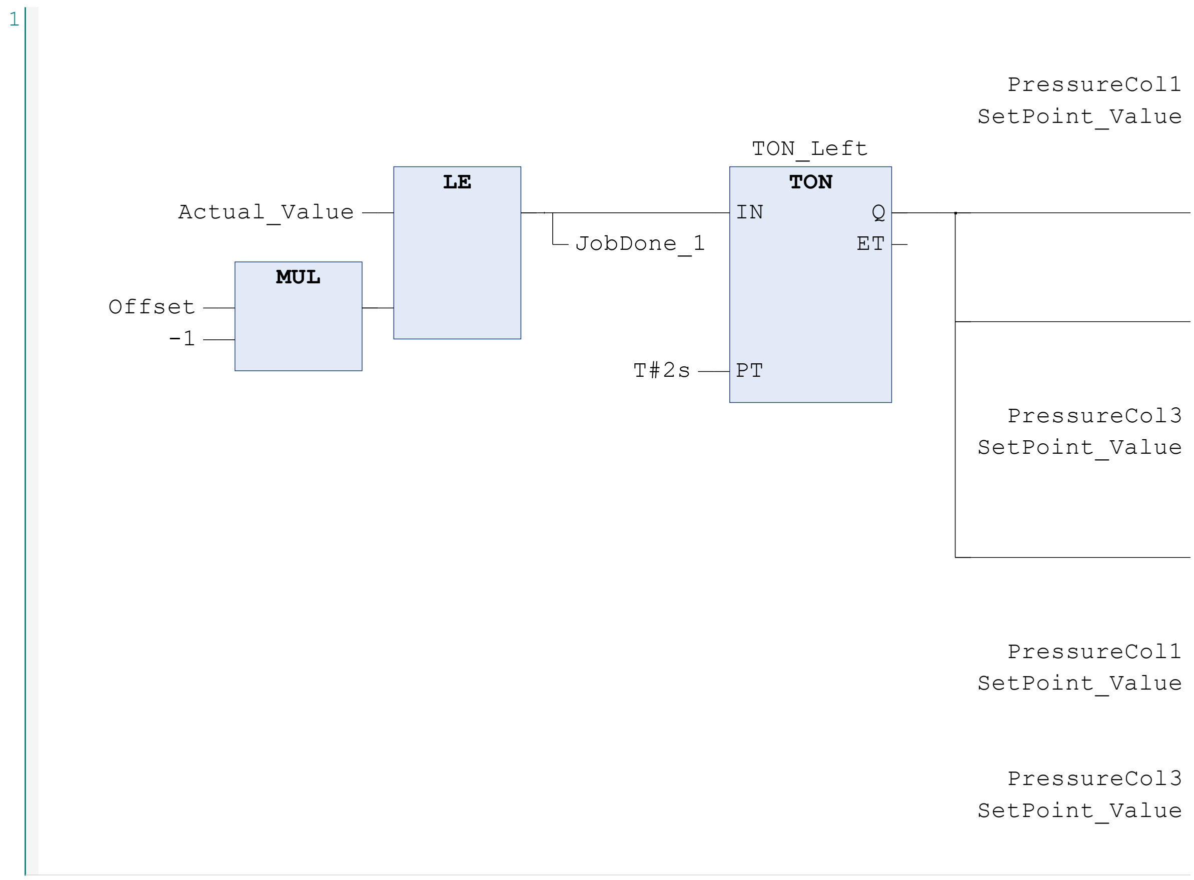
```
g_Globals_Mapping.Col1Out

g_Globals_Mapping.Col2Out

g_Globals_Mapping.Col3Out

g_Globals_Mapping.Col4Out
```

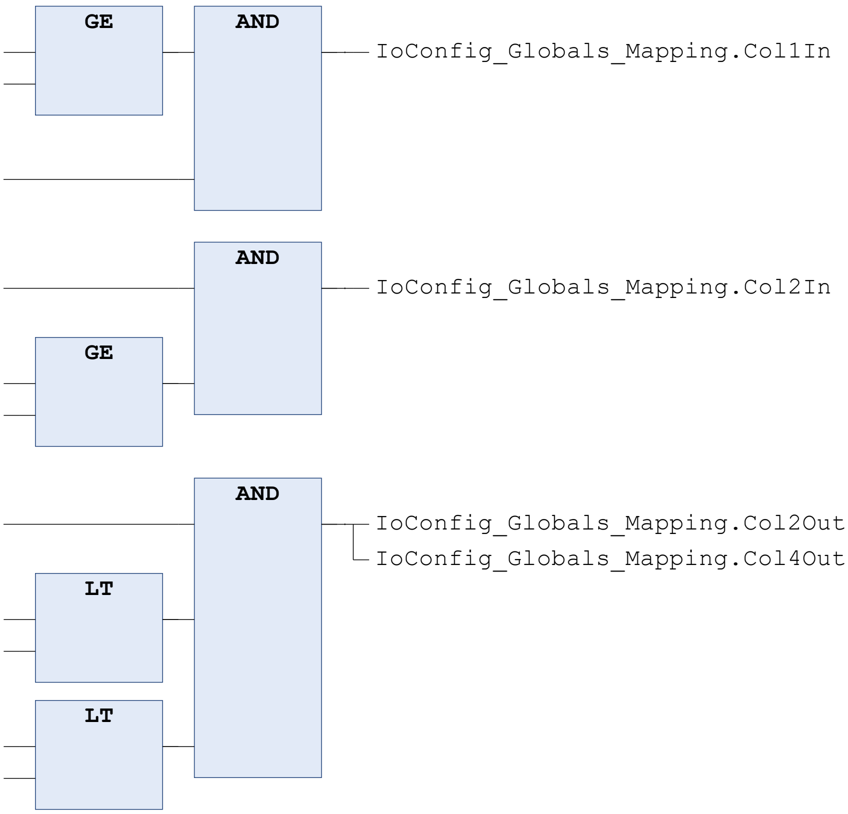

3
TON_To_High
GE
AND
TON
DraftAverage
IN
Q
JobDone_2
ET
ADD
DraftSetpoint
Offset
NOT
ColumnEmpty
T#2s
PT
4
NOT
AND
JobDone_1
JobDone
NOT
JobDone_2

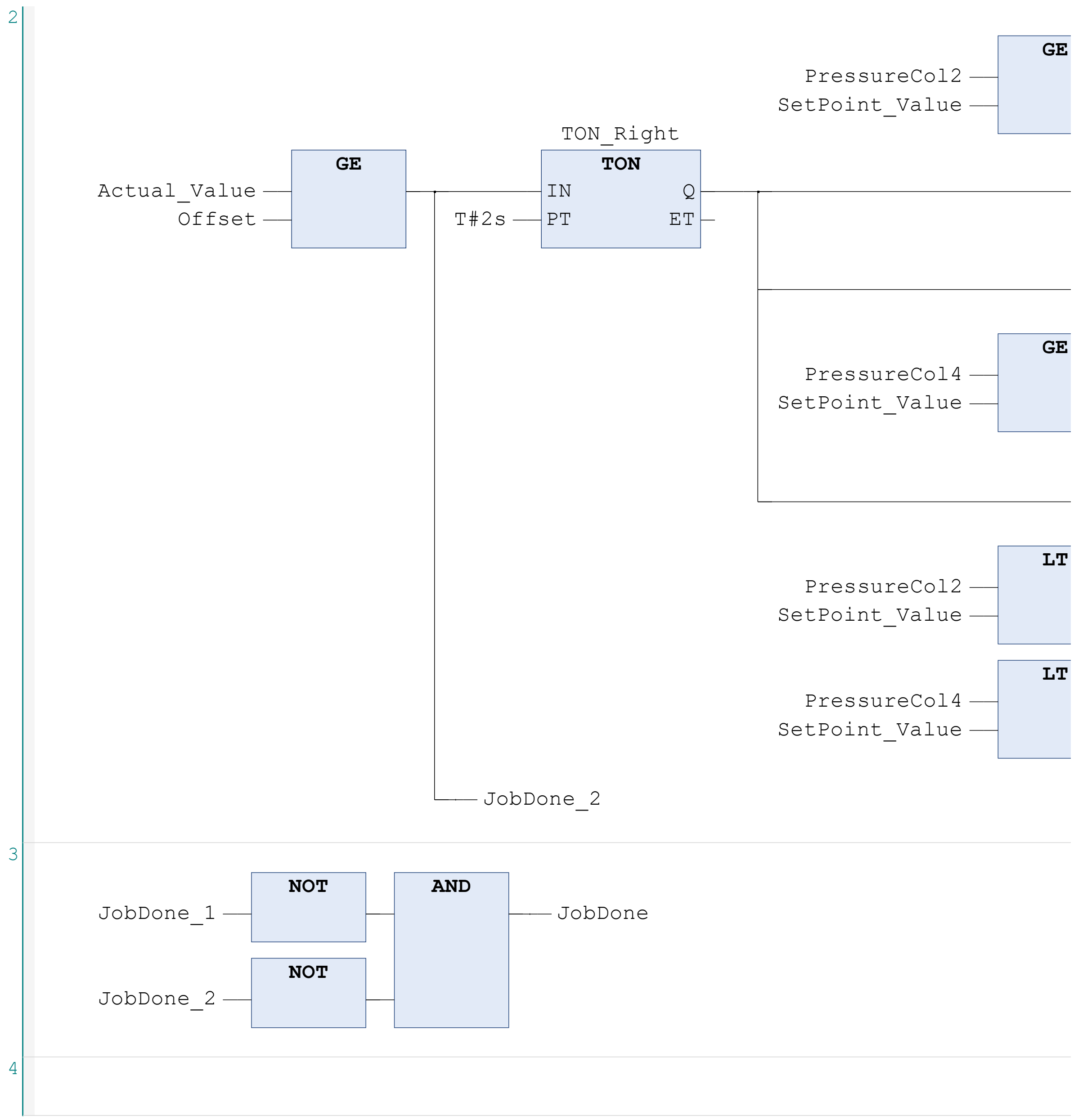

IoConfig_Globals_Mapping.Col1In
IoConfig_Globals_Mapping.Col2In
IoConfig_Globals_Mapping.Col3In
IoConfig_Globals_Mapping.Col4In

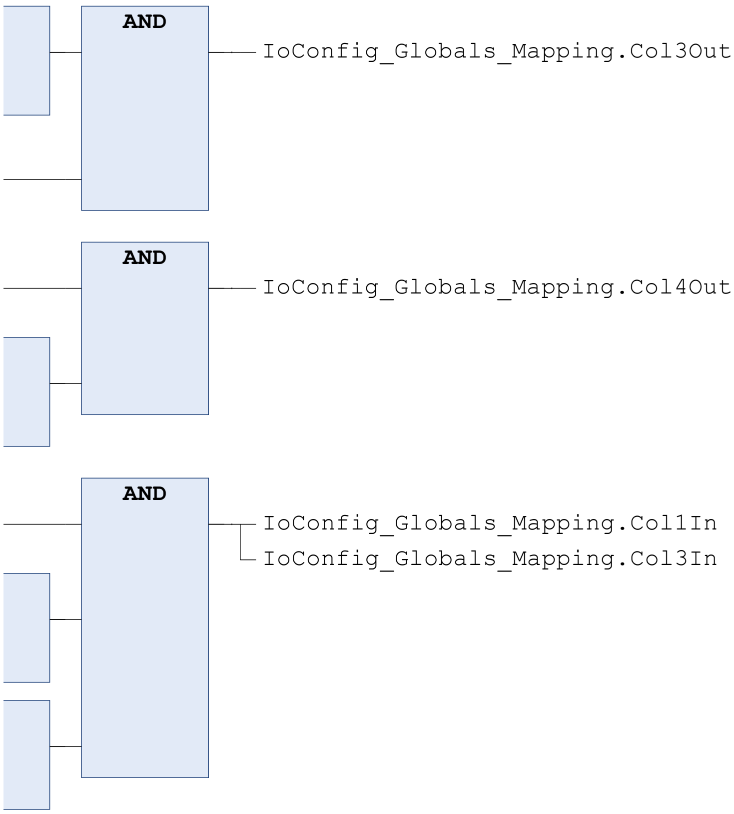
3.1.1.1.13 POU: FB_Stablilization_Roll

```
FUNCTION_BLOCK  FB_Stablilization_Roll
VAR_INPUT
        Actual_Value :  REAL ;
    SetPoint_Value :  REAL ;
    Offset :  REAL ;
    PressureCol1 :  REAL ;
    PressureCol2 :  REAL ;
    PressureCol3 :  REAL ;
    PressureCol4 :  REAL ;
END_VAR
VAR_OUTPUT
    JobDone : BOOL ;
END_VAR
VAR
        OffsetMinus :  REAL ;
    OffsetPlus :  REAL ;
    JobDone_1 :  BOOL ;
    JobDone_2 :  BOOL ;
    TON_Left :  TON ;
    TON_Right :  TON ;
END_VAR

```


1
PressureCol1
SetPoint_Value
TON_Left
LE
TON
Actual_Value
IN
Q
JobDone_1
ET
MUL
Offset
-1
T#2s
PT
PressureCol3
SetPoint_Value
PressureCol1
SetPoint_Value
PressureCol3
SetPoint_Value


GE
AND
IoConfig_Globals_Mapping.Col1In
AND
IoConfig_Globals_Mapping.Col2In
GE
AND
IoConfig_Globals_Mapping.Col2Out
IoConfig_Globals_Mapping.Col4Out
LT
LT


2
GE
PressureCol2
SetPoint_Value
TON_Right
GE
TON
Actual_Value
Offset
IN
Q
T#2s
PT
ET
GE
PressureCol4
SetPoint_Value
LT
PressureCol2
SetPoint_Value
LT
PressureCol4
SetPoint_Value
JobDone_2
3
NOT
AND
JobDone_1
JobDone
NOT
JobDone_2
4


AND
IoConfig_Globals_Mapping.Col3Out
AND
IoConfig_Globals_Mapping.Col4Out
AND
IoConfig_Globals_Mapping.Col1In
IoConfig_Globals_Mapping.Col3In

3.1.1.1.14 POU: FB_ThrusterScaling

```
FUNCTION_BLOCK  FB_ThrusterScaling
VAR_INPUT
    Enable :  BOOL ;
    Forward :  BOOL ;
    Backward :  BOOL ;
    LeftPivot :  BOOL ;
    RightPivot :  BOOL ;
    Left :  BOOL ;
    Right :  BOOL ;
    ThrusterValue :  REAL ;

END_VAR
VAR_OUTPUT
    Forward_Out :  REAL ;
    Backward_Out :  REAL ;
    LeftPivot_Out :  REAL ;
    RightPivot_Out :  REAL ;
    Left_Out :  REAL ;
    Right_Out :  REAL ;


END_VAR
VAR
    Forward_Lin_1 :  Util . LIN_TRAFO ;
    Forward_Lin_2 :  Util . LIN_TRAFO ;
    Backward_Lin_1 :  Util . LIN_TRAFO ;
    Backward_Lin_2 :  Util . LIN_TRAFO ;

    LeftPivot_Lin_1 :  Util . LIN_TRAFO ;
    LeftPivot_Lin_2 :  Util . LIN_TRAFO ;
    RightPivot_Lin_1 :  Util . LIN_TRAFO ;
    RightPivot_Lin_2 :  Util . LIN_TRAFO ;

    Left_Lin_1 :  Util . LIN_TRAFO ;
    Left_Lin_2 :  Util . LIN_TRAFO ;
    Right_Lin_1 :  Util . LIN_TRAFO ;
    Right_Lin_2 :  Util . LIN_TRAFO ;


END_VAR

```

```
IF  ( Enable )  THEN

    IF  ( Forward )  THEN
        Forward_Lin_1 (
        IN :=  ThrusterValue ,
```

```
            IN_MIN :=  0 ,
            IN_MAX :=  100 ,
            OUT_MIN :=  5500 ,
            OUT_MAX :=  0 ,
            OUT =>  Forward_Out ,
            ERROR =>  ) ;

            Backward_Lin_2 (
            IN :=  ThrusterValue ,
            IN_MIN :=  0 ,
            IN_MAX :=  100 ,
            OUT_MIN :=  5500 ,
            OUT_MAX :=  0 ,
            OUT =>  Backward_Out ,
            ERROR =>  ) ;
        END_IF


        IF  ( Backward )  THEN
            Backward_Lin_1 (
            IN :=  ThrusterValue ,
            IN_MIN :=  0 ,
            IN_MAX :=  100 ,
            OUT_MIN :=  5500 ,
            OUT_MAX :=  10000 ,
            OUT =>  Backward_Out ,
            ERROR =>  ) ;

            Forward_Lin_2 (
            IN :=  ThrusterValue ,
            IN_MIN :=  0 ,
            IN_MAX :=  100 ,
            OUT_MIN :=  5500 ,
            OUT_MAX :=  10000 ,
            OUT =>  Forward_Out ,
            ERROR =>  ) ;
        END_IF


        IF  ( LeftPivot )  THEN
            LeftPivot_Lin_1 (
            IN :=  ThrusterValue ,
            IN_MIN :=  0 ,
            IN_MAX :=  100 ,
            OUT_MIN :=  5500 ,
            OUT_MAX :=  0 ,
            OUT =>  LeftPivot_Out ,
            ERROR =>  ) ;

            RightPivot_Lin_2 (
            IN :=  ThrusterValue ,
```

```
            IN_MIN := 0 ,
            IN_MAX := 100 ,
            OUT_MIN := 5500 ,
            OUT_MAX := 0 ,
            OUT => RightPivot_Out ,
            ERROR => ) ;
        END_IF


        IF ( RightPivot ) THEN
            RightPivot_Lin_1 (
            IN := ThrusterValue ,
            IN_MIN := 0 ,
            IN_MAX := 100 ,
            OUT_MIN := 5500 ,
            OUT_MAX := 10000 ,
            OUT => RightPivot_Out ,
            ERROR => ) ;

            LeftPivot_Lin_2 (
            IN := ThrusterValue ,
            IN_MIN := 0 ,
            IN_MAX := 100 ,
            OUT_MIN := 5500 ,
            OUT_MAX := 10000 ,
            OUT => LeftPivot_Out ,
            ERROR => ) ;
        END_IF

        IF ( Left ) THEN
            Left_Lin_1 (
            IN := ThrusterValue ,
            IN_MIN := 0 ,
            IN_MAX := 100 ,
            OUT_MIN := 5500 ,
            OUT_MAX := 0 ,
            OUT => Left_Out ,
            ERROR => ) ;

            Right_Lin_2 (
            IN := ThrusterValue ,
            IN_MIN := 0 ,
            IN_MAX := 100 ,
            OUT_MIN := 5500 ,
            OUT_MAX := 10000 ,
            OUT => Right_Out ,
            ERROR => ) ;
        END_IF


        IF ( Right ) THEN
```

```
            Right_Lin_1 (
            IN := ThrusterValue ,
            IN_MIN := 0 ,
            IN_MAX := 100 ,
            OUT_MIN := 5500 ,
            OUT_MAX := 0 ,
            OUT => Right_Out ,
            ERROR => ) ;

            Left_Lin_2 (
            IN := ThrusterValue ,
            IN_MIN := 0 ,
            IN_MAX := 100 ,
            OUT_MIN := 5500 ,
            OUT_MAX := 10000 ,
            OUT => Left_Out ,
            ERROR => ) ;
        END_IF

    END_IF

```

3.1.1.1.15 POU: FB_Thruster_Values

```
FUNCTION_BLOCK FB_Thruster_Values
VAR_INPUT
    Enable : BOOL ;
    ThrusterValue : REAL ;
END_VAR
VAR_OUTPUT
    OutputThrust : REAL ;
END_VAR
VAR
END_VAR

```

```
IF ( Enable ) THEN
    OutputThrust := ThrusterValue ;
    ELSE
        OutputThrust := 5500 ;
END_IF

```

3.1.1.1.16 POU: F_RotationError

```
FUNCTION F_RotationError : INT
VAR_INPUT
    Setpoint_Angle : INT ;
    Current_Angle : INT ;
END_VAR
VAR
    max_Value : INT := 360 ;
    rev : INT ;
    fw : INT ;
    angle : INT ;
END_VAR

```

```
angle := Current_Angle ;

IF Setpoint_Angle > angle THEN
    rev := angle + max_Value - Setpoint_Angle ;
    ELSE
        rev := angle - Setpoint_Angle ;
    END_IF

IF  Setpoint_Angle < angle THEN
    fw := max_Value - angle + Setpoint_Angle ;
    ELSE
        fw := Setpoint_Angle - angle ;
END_IF

IF fw <= rev THEN

    F_RotationError := - fw ;
    ELSE
        F_RotationError := rev ;
    END_IF
RETURN ;

```

3.1.1.2 Folder: Global Variables

3.1.1.2.1 Global Variable List: Global_Variables

```
{attribute 'qualified_only'}
VAR_GLOBAL


    GUIisDisconnected :  BOOL ;
    RPIisDisconnected :  BOOL ;


    PressureCol1 :  REAL ;
    PressureCol2 :  REAL ;
    PressureCol3 :  REAL ;
    PressureCol4 :  REAL ;
    PressurePS :  REAL ;
    PressureSB :  REAL ;


    // declared by MODBUS-Configurator
    GPS_Speed :  REAL ;

    // declared by MODBUS-Configurator
    GPS_NumbersOfSatelites :  INT ;

    // declared by MODBUS-Configurator
    GPS_Enabled :  BOOL ;

    // declared by MODBUS-Configurator
    GPS_Latitude :  LREAL ;

    // declared by MODBUS-Configurator
    GPS_Longitude :  LREAL ;

    // declared by MODBUS-Configurator
    GPS_Heading :  LREAL ;

    // declared by MODBUS-Configurator
    ConnectionCheck :  BOOL ;

    // declared by MODBUS-Configurator
    Gyro_Pitch :  REAL ;

    // declared by MODBUS-Configurator
    Gyro_Roll :  REAL ;

    // declared by MODBUS-Configurator
    Gyro_Yaw :  REAL ;

    // declared by MODBUS-Configurator
    FwdMotion :  BOOL ;
```

```

    // declared by MODBUS-Configurator
    BackMotion :  BOOL ;

    // declared by MODBUS-Configurator
    RightMotion :  BOOL ;

    // declared by MODBUS-Configurator
    LeftMotion :  BOOL ;

    // declared by MODBUS-Configurator
    ClockWMotion :  BOOL ;

    // declared by MODBUS-Configurator
    CounterClockMotion :  BOOL ;

    // declared by MODBUS-Configurator
    EnableLight :  BOOL ;

    // declared by MODBUS-Configurator
    EnableFlute :  BOOL ;

    // declared by MODBUS-Configurator
    PlatformEnable :  BOOL ;

    // declared by MODBUS-Configurator
    EnableAuto :  BOOL ;

    // declared by MODBUS-Configurator
    EnableManual :  BOOL ;

    // declared by MODBUS-Configurator
    Enable_DP :  BOOL ;

    // declared by MODBUS-Configurator
    WinchUp :  BOOL ;

    // declared by MODBUS-Configurator
    WinchDown :  BOOL ;

    // declared by MODBUS-Configurator
    Winch_Lock_On :  BOOL ;

    // declared by MODBUS-Configurator
    Winch_Lock_Off :  BOOL ;

    // declared by MODBUS-Configurator
    Start_Pump :  BOOL ;

    // declared by MODBUS-Configurator
    ThrusterSpeed :  INT ;
```

```
        // declared by MODBUS-Configurator
        WinchSpeed :  INT ;

        // declared by MODBUS-Configurator
        GUI_Latitude :  REAL ;

        // declared by MODBUS-Configurator
        GUI_Longitude :  REAL ;

        // declared by MODBUS-Configurator
        GUI_ConCheck :  BOOL ;


        // declared by MODBUS-Configurator
        platLat :  REAL ;

        // declared by MODBUS-Configurator
        PlatLong :  REAL ;

        // declared by MODBUS-Configurator
        platYaw :  REAL ;

        // declared by MODBUS-Configurator
        platRoll :  REAL ;

        // declared by MODBUS-Configurator
        PLC_Run :  BOOL ;

        // declared by MODBUS-Configurator  Pitch
        platHeading :  REAL ;

        // declared by MODBUS-Configurator
        platSpeed :  REAL ;

        // declared by MODBUS-Configurator
        platROVLocked :  BOOL ;

        // declared by MODBUS-Configurator
        platROVUpperPos :  BOOL ;

        // declared by MODBUS-Configurator
        platDP_ModeEnabled :  BOOL ;

        // declared by MODBUS-Configurator
        platAutopilot_Enabled :  BOOL ;

        // declared by MODBUS-Configurator
        platManual_ModeEnabled :  BOOL ;
```

```
        // declared by MODBUS-Configurator
        ROVTemp :  REAL ;

        // declared by MODBUS-Configurator
        ROVDepth :  REAL ;

        // declared by MODBUS-Configurator
        ROVWaterTemp :  REAL ;

        // declared by MODBUS-Configurator
        ROVOxygenWater :  REAL ;

        // declared by MODBUS-Configurator
        ROVHeading :  REAL ;

    END_VAR

```

3.1.1.2.2 Global Variable List: simGVL

```
    {attribute 'qualified_only'}
    VAR_GLOBAL
        // Thruster Variables
        simThrustValue :  REAL ;
        simForward :  BOOL ;
        simBackward :  BOOL ;
        simLeftPivot :  BOOL ;
        simRightPivot :  BOOL ;

        simLeft :  BOOL ;
        simRight :  BOOL ;

        // Stabilization Variables
        simActual :  REAL ;
        simSetPoint :  REAL ;
        simOffset :  REAL  := 5 ;

        // Pitch/Roll
        simPitchAvg :   REAL ;
        simRollAvg :    REAL ;

        // Tanks
        simBtnEmptyTanks :   BOOL ;
        simEmptyTanksLamp :  BOOL ;

        // Pressure
        simCol1Pressure :    REAL ;
        simCol2Pressure :    REAL ;
        simPortPressure :    REAL ;
        simStarboardPressure :  REAL ;
        simCol3Pressure :    REAL ;
```

```
        simCol4Pressure :    REAL ;

        // Draft
        simDraftPort :  REAL ;
        simDraftStarboard :  REAL ;
        simDraftAvg :  REAL ;
        simDraftSetPoint :  REAL  := 0 ;


        // Plaform
        simLanterns :  BOOL ;
        simHorn :  BOOL ;
        simKillSwitch_SMC :  BOOL ;

        // simBtn
        simBtnBool :  BOOL ;
        simSimPressure :  BOOL ;
        simBtnNext :  BOOL ;

        // Dockinghead
        simEnableDH :  BOOL ;
        simDH_InPosFeedback :  BOOL ;
        simDH_OpenFeedback :  BOOL ;
        simDH_ClosedFeedback :  BOOL ;
        simDH_Open : BOOL ;
        simDH_Close :  BOOL ;

        // Winch
        simWinchIn :  BOOL ;
        simWinchOut :  BOOL ;
        simWinchSpeed :  INT ;
        simEnableWinch :  BOOL ;
        simMOut : DINT ;

        // AutoPilot
        simAutopilotON :  BOOL ;
        simAngle :  REAL ;

        simGPSLatitude :  LREAL  :=  62.458642 ;  //Y
        simGPSLongitude :  LREAL  :=  6.204729 ;  //X
        simGyroHeading :  LREAL ;
        simWaypointXLon :  LREAL  :=  6.211937 ;  //Long
        simWaypointYLat :  LREAL  :=  62.459837 ;  // Lat


        simAngleCalc  :  INT ;

        simActualValue :  INT ;

        simThrusterPIDTurn :  REAL ;

```

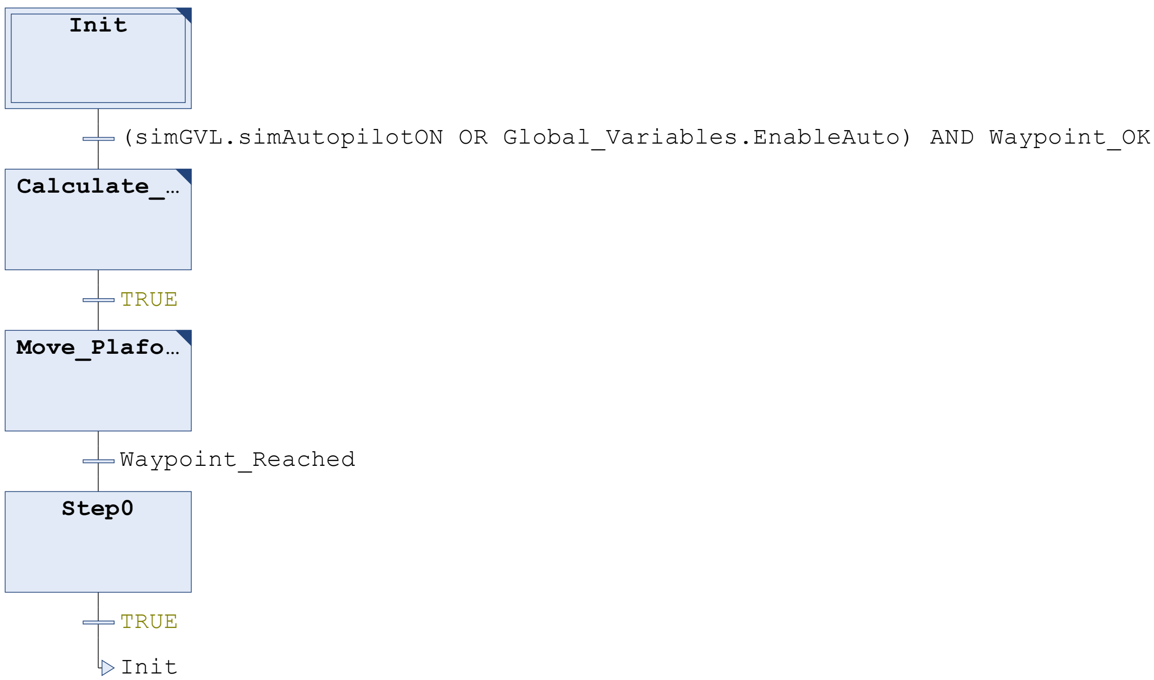
```
        simThrusterPIDDistance :  REAL ;

        simThrustRight :  BOOL ;
        simThrustLeft :  BOOL ;
        simActualValueDist : REAL ;





    END_VAR

```

3.1.1.3 Folder: POUs

3.1.1.3.1 POU: POU_Autopilot

```
    PROGRAM  POU_Autopilot
    VAR
        // Init
        Autopilot_On  :  BOOL ;
        Waypoint_OK  : BOOL ;

        // Calculate Variables
        y  :  LREAL ;
        x  :  LREAL ;
        bearing  :  LREAL ;
        angle  :  LREAL ;

    xEnable
        //ThrusterValueCorr: REAL;

        // Move Plaform Variables
          :  BOOL ;
        PID_Direction  :  WagoAppBuildingHVAC . FbPIDController ;
        deadZone  :  REAL ;
        P  :  REAL  := 10 ;
        I  :  REAL ;
        D  :  REAL ;

        presentON :  BOOL  :=  TRUE ;
        presentOFF :  BOOL  :=  TRUE ;
        rY  :  REAL ;

        Waypoint_Reached  :  BOOL ;



        //Simulator variables
```

```
        FB_Simulate_ThrusterDir_0  :  FB_Simulate_ThrusterDir ;


    END_VAR

```

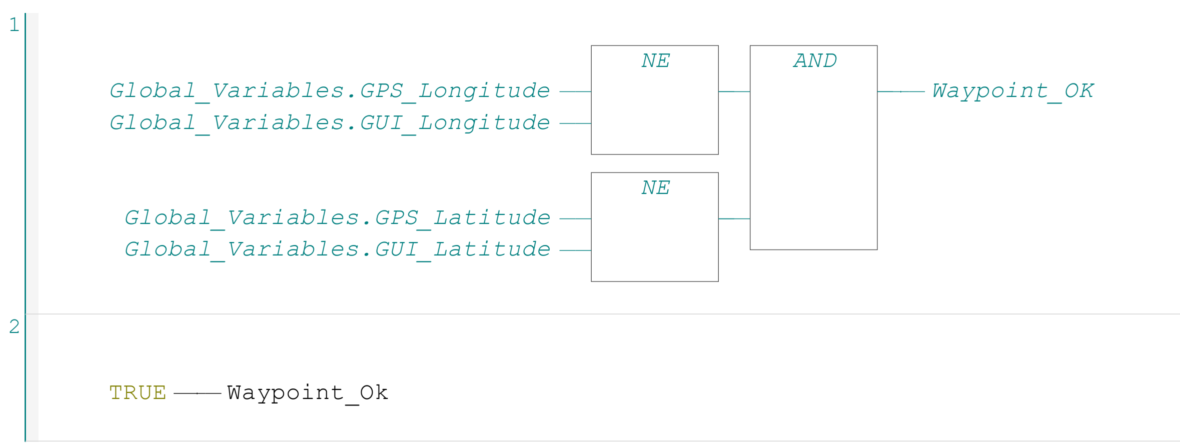


3.1.1.3.1.1 Action: Calculate_Angle_active

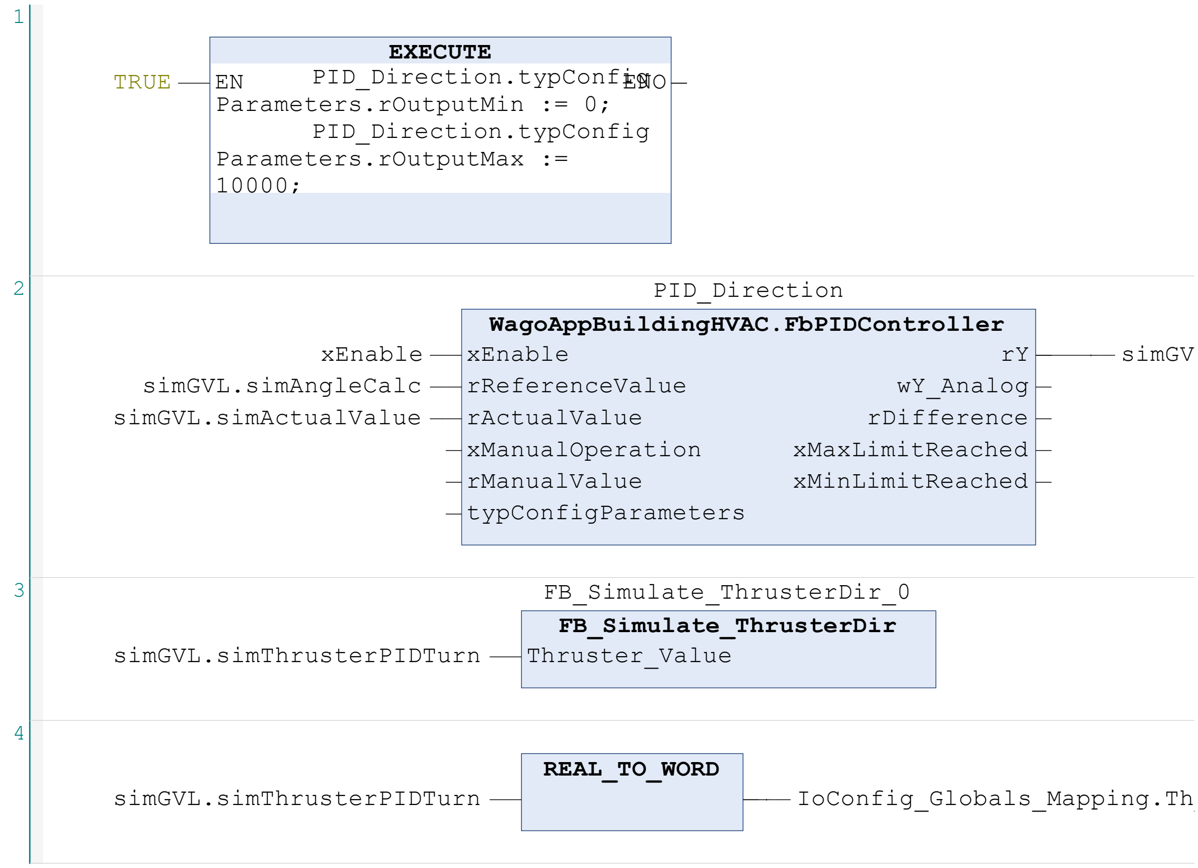
```
y := ( WagoAppMath . sin_L ( phi := simGVL . simWaypointXLon - simGVL . simGPSLongitude ) ) * WagoAppMath . cos_L ( phi := simGVL . simWaypointYLat ) ;
x := WagoAppMath . cos_L ( phi := Global_Variables . GPS_Latitude ) * WagoAppMath . sin_L ( phi := simGVL . simWaypointYLat ) - WagoAppMath . sin_L ( phi := Global_Variables . GPS_Latitude ) * WagoAppMath . cos_L ( phi := simGVL . simWaypointYLat ) * WagoAppMath . cos_L ( phi := simGVL . simWaypointXLon - Global_Variables . GPS_Longitude ) ;
bearing := WagoAppMath . arcTan2 ( y := y , x := x ) ;
//Angle2 := WagoAppMath.radiantToAngle(lrRadiant:= bearing);
Angle := WagoAppMath . angleToDegree_L ( phi := bearing ) ;




























//GPS_Latitude y
//GPS_Longitude x
//Gyro_Yaw
// Waypoint_Latitude
// Waypoint_Longitude


//dy := GPS_Latitude - Waypoint_Latitude;
//dx := GPS_Longitude - Waypoint_Longitude;
```

```
    //Longitude := (simGVL.simWaypointX - Global_Variables.GPS_Longitude);
    //Latitude := (simGVL.simWaypointY - Global_Variables.GPS_Latitude);


    ////////////////////////
    //Longitude := (simGVL.simWaypointX - simGVL.simGPSLongitude);
    //Latitude := (simGVL.simWaypointY - simGVL.simGPSLatitude);
    //absLongitude := WagoAppMath.abs_L(x:= Longitude);
    //absLatitude := WagoAppMath.abs_L(x:= Latitude);
    //angle_to_Waypoint := WagoAppMath.arcTan2(y:= absLongitude, x:=absLatitude );
    //angle := WagoAppMath.angleToDegree_L(phi:= angle_to_Waypoint);
    //absAngle := WagoAppMath.abs_L(x:= angle);
    // Check where waypoint is located:
    //IF (Latitude >= 0 AND Longitude >= 0) THEN
        //Final_Angle := absAngle + 0;
    //        simGVL.simAngle := absAngle + 0;
    //END_IF
    //IF (Latitude < 0 AND Longitude >= 0) THEN
        //Final_Angle := absAngle + 90;
    //        simGVL.simAngle := absAngle + 90;
    //END_IF
    //IF (Latitude < 0 AND Longitude < 0) THEN
        //Final_Angle := absAngle + 180;
        //simGVL.simAngle := absAngle + 180;
    //END_IF
    //IF (Latitude >= 0 AND Longitude < 0) THEN
        //Final_Angle := absAngle + 270;
        //simGVL.simAngle := absAngle + 270;
    //END_IF
    ////////////////////////

```

3.1.1.3.1.2 Action: Init_active



3.1.1.3.1.3 Action: Move_Plaform_active

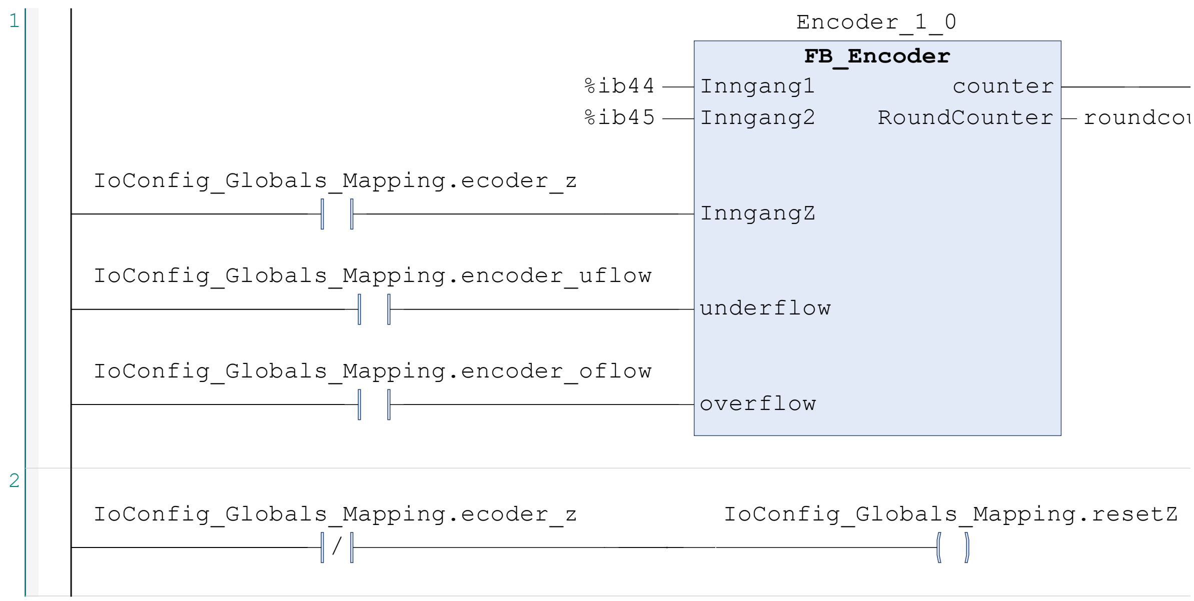


L.simThrusterPIDTurn

_PS

3.1.1.3.2 POU: POU_Encoder_Winch

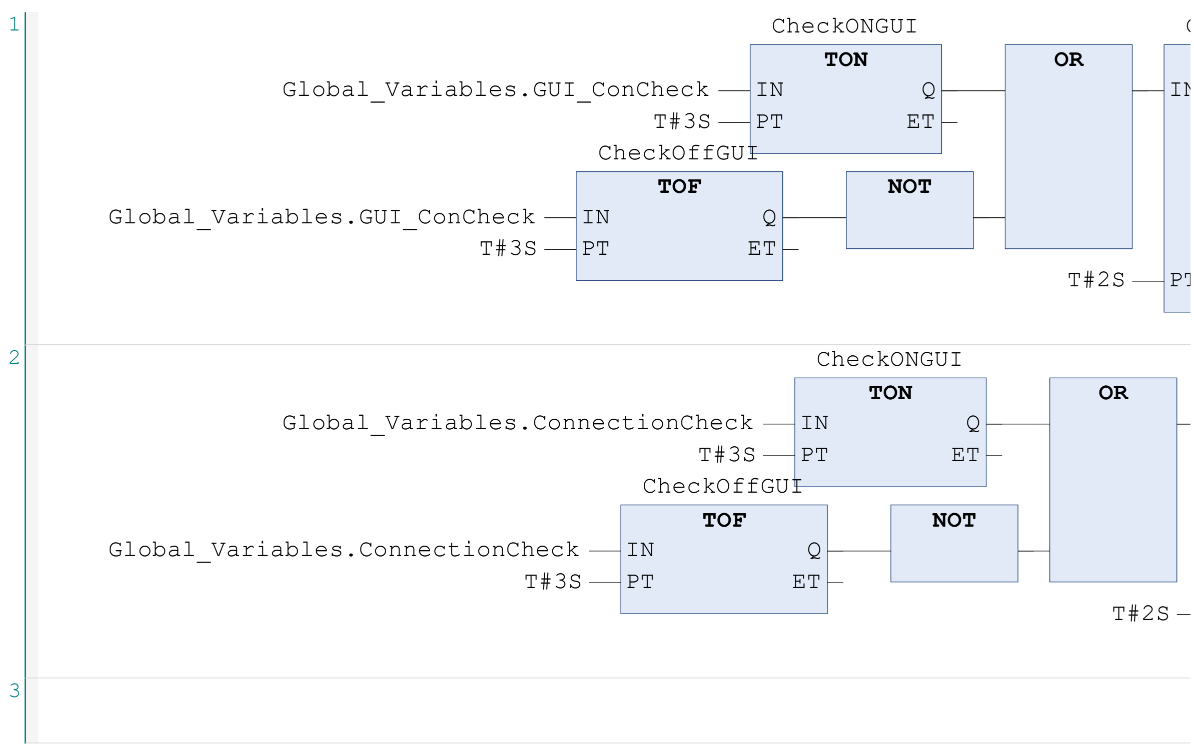
```
PROGRAM  POU_Encoder_Winch
VAR
    Encoder_1_0  :  FB_Encoder ;
    roundcounter  :  INT ;
    counter :  UINT ;

END_VAR

```

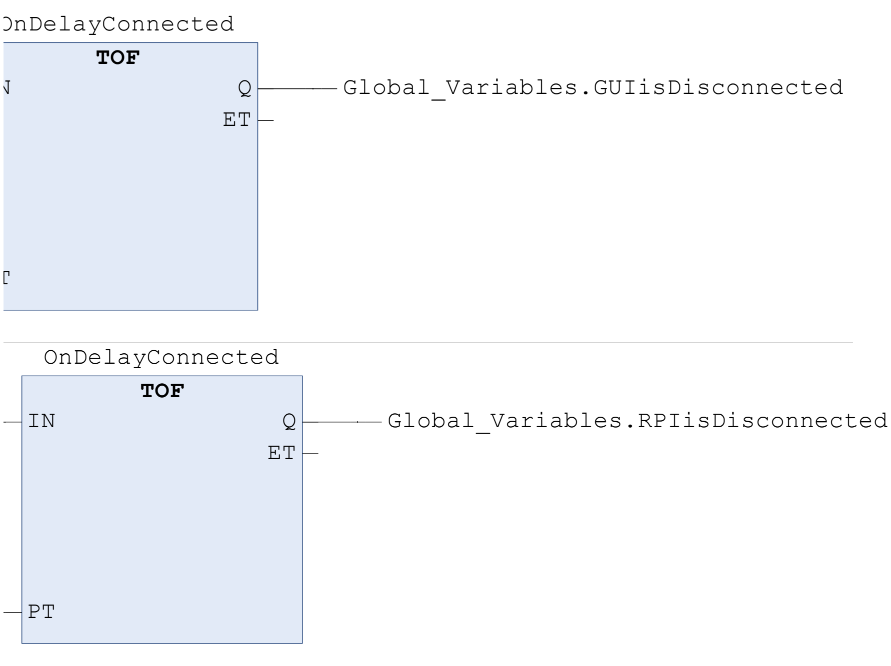


counter
unter

3.1.1.3.3 POU: POU_Platform_General

```
PROGRAM  POU_Platform_General
VAR
    CheckONGUI :  TON ;
    CheckOffGUI :  TOF ;
    OnDelayConnected :  TOF ;

    Lat :  REAL ;
    Long :  REAL ;

    current :  WORD ;

    ROVDepth :  REAL ;
    ROVOxygenWater :  REAL ;
    ROVTemp  :  REAL ;
    ROVWaterTemp :  REAL ;
    ROVHeading :  REAL ;
END_VAR

```




OnDelayConnected
TOF
Q
ET
Global_Variables.GUIisDisconnected
OnDelayConnected
TOF
IN
Q
ET
PT
Global_Variables.RPIisDisconnected

4 FALSE —— IoConfig_Globals_Mapping.Lanterns

5 TRUE —— IoConfig_Globals_Mapping.KillSwitch_SMC

6 Global_Variables.EnableFlute —, simGVL.simHorn — **OR** —— IoConfig_Globals_Mapping.Horn

7

8 Global_Variables.GPS_Latitude —— Global_Variables.platLat

9 Global_Variables.GPS_Longitude —— Global_Variables.platLong

10 Global_Variables.Enable_DP —— Global_Variables.platDP_ModeEnabled

11 Global_Variables.EnableAuto —— Global_Variables.platAutopilot_Enabled

12 Global_Variables.EnableManual —— Global_Variables.platManual_ModeEnabled

13

14 IoConfig_Globals_Mapping.DH_LatchClosed —— Global_Variables.platROVLocked

15 IoConfig_Globals_Mapping.DH_InPos —— Global_Variables.platROVUpperPos

16 *Global_Variables.GUI_Latitude —— Lat*

17 *Global_Variables.GUI_Longitude —— Long*

18 Global_Variables.Gyro_Pitch —— Global_Variables.platHeading

19 Global_Variables.Gyro_Roll —— Global_Variables.platRoll

20 Global_Variables.Gyro_Yaw —— Global_Variables.platYaw

21 Global_Variables.GPS_Speed —— Global_Variables.platSpeed

22 IoConfig_Globals_Mapping.Current_Logg —— current

23 Global_Variables.ROVDepth —— ROVDepth

24 Global_Variables.ROVOxygenWater —— ROVOxygenWater

25 Global_Variables.ROVTemp —— ROVTemp

26 Global_Variables.ROVWaterTemp —— ROVWaterTemp

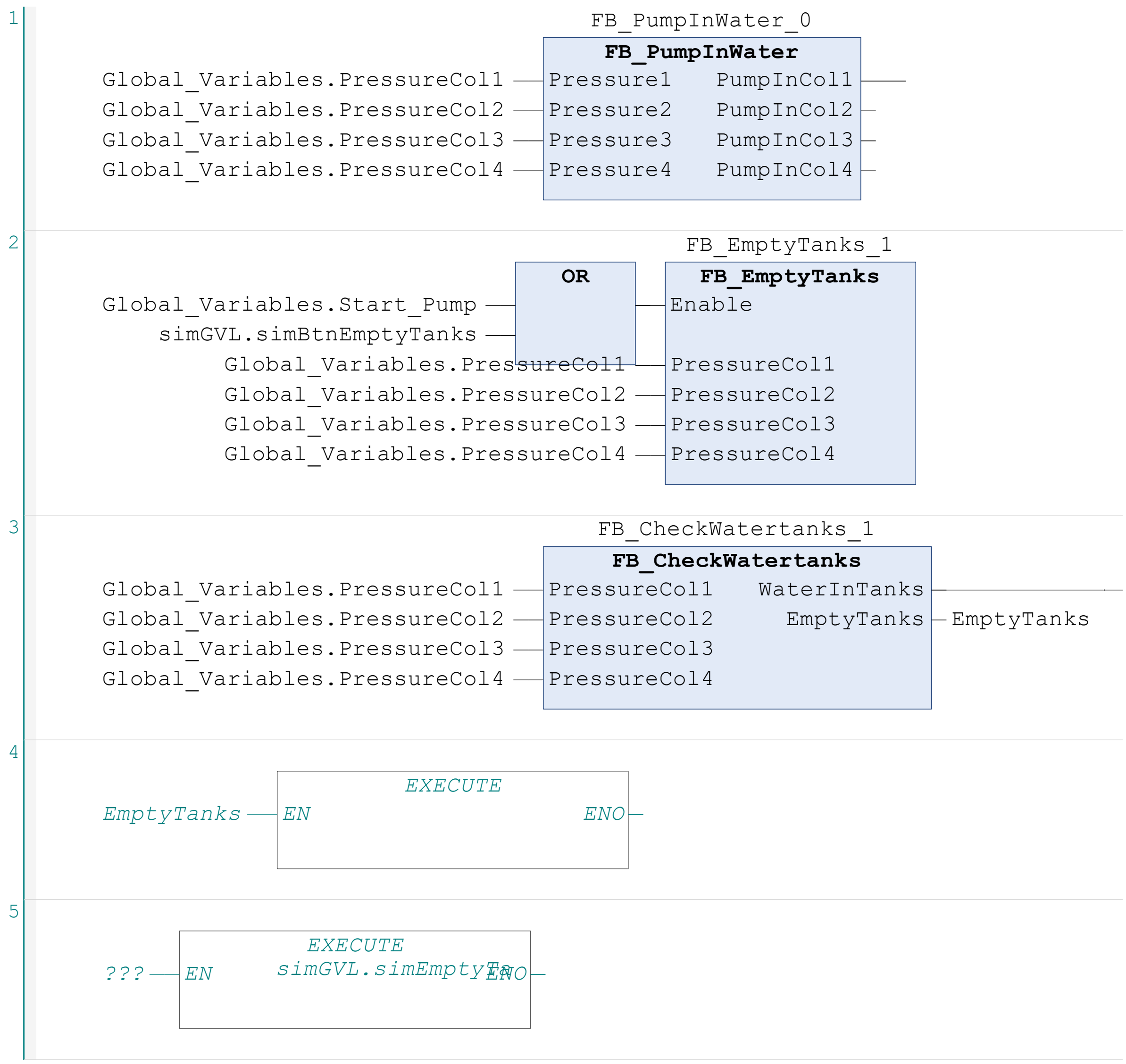
27

Global_Variables.ROVHeading —— ROVHeading

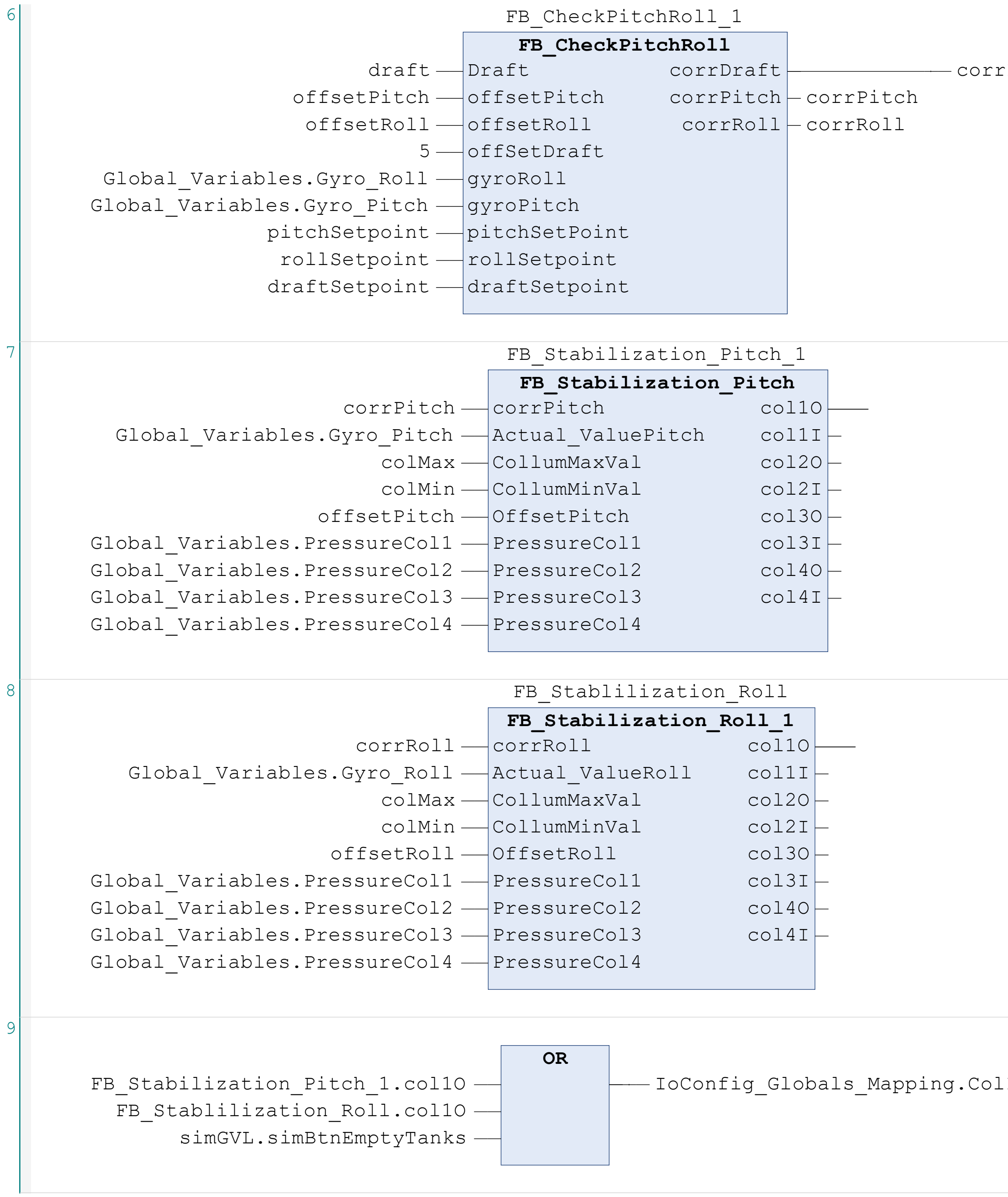
3.1.1.3.4 POU: POU_Stablilization

```
PROGRAM  POU_Stablilization
VAR

    draft :  REAL ;
    xEnable :  BOOL ;
    xEnablePitch :  BOOL ;
    xEnableRoll :  BOOL ;
    xEnableDraft :  BOOL ;

    WaterInTank :  BOOL ;
    EmptyTanks :  BOOL ;

    corrDraft :  BOOL ;
    corrRoll : BOOL ;
    corrPitch : BOOL ;

    JobDonePitch :  BOOL ;
    JobDoneRoll :  BOOL ;
    JobDoneDraft :  BOOL ;

    FB_CheckPitchRoll_1 : FB_CheckPitchRoll ;
    FB_EmptyTanks_1 :  FB_EmptyTanks ;
    FB_CheckWatertanks_1 :  FB_CheckWatertanks ;
    FB_Stabilization_Pitch_1 :  FB_Stabilization_Pitch ;
    FB_Stablilization_Roll :  FB_Stabilization_Roll_1 ;
    FB_Stablilization_Draft_1 :  FB_Stablilization_Draft ;
    FB_PumpInWater_0 :  FB_PumpInWater ;



    offset_Setpoint :  INT ;
    offset_SetpointMinus :  INT ;
    pitchSetpoint : REAL  :=  0 ;
    rollSetpoint :  REAL  :=  0 ;
    draftSetpoint :  REAL  :=  0 ;

    offsetRoll :  REAL  :=  5 ;
    offsetPitch :  REAL  :=  5 ;

    colMin :  REAL  :=  5 ;
    colMax :  REAL  :=  40 ;

    timeSet :  INT ;
END_VAR

```

3.1.1.3.4 POU: POU_Stablilization



-WaterInTank


6
FB_CheckPitchRoll_1
FB_CheckPitchRoll
draft
Draft
corrDraft
corrI
offsetPitch
offsetPitch
corrPitch
corrPitch
offsetRoll
offsetRoll
corrRoll
corrRoll
5
offSetDraft
Global_Variables.Gyro_Roll
gyroRoll
Global_Variables.Gyro_Pitch
gyroPitch
pitchSetpoint
pitchSetPoint
rollSetpoint
rollSetpoint
draftSetpoint
draftSetpoint
7
FB_Stabilization_Pitch_1
FB_Stabilization_Pitch
corrPitch
corrPitch
col1O
Global_Variables.Gyro_Pitch
Actual_ValuePitch
col1I
colMax
CollumMaxVal
col2O
colMin
CollumMinVal
col2I
offsetPitch
OffsetPitch
col3O
Global_Variables.PressureCol1
PressureCol1
col3I
Global_Variables.PressureCol2
PressureCol2
col4O
Global_Variables.PressureCol3
PressureCol3
col4I
Global_Variables.PressureCol4
PressureCol4
8
FB_Stablilization_Roll
FB_Stabilization_Roll_1
corrRoll
corrRoll
col1O
Global_Variables.Gyro_Roll
Actual_ValueRoll
col1I
colMax
CollumMaxVal
col2O
colMin
CollumMinVal
col2I
offsetRoll
OffsetRoll
col3O
Global_Variables.PressureCol1
PressureCol1
col3I
Global_Variables.PressureCol2
PressureCol2
col4O
Global_Variables.PressureCol3
PressureCol3
col4I
Global_Variables.PressureCol4
PressureCol4
9
OR
FB_Stabilization_Pitch_1.col1O
FB_Stablilization_Roll.col1O
simGVL.simBtnEmptyTanks
IoConfig_Globals_Mapping.Col

Draft

lOut

10

Inputs	Block	Output
FB_Stabilization_Pitch_1.col1I	OR	IoConfig_Globals_Mapping.Col
FB_Stablilization_Roll.col1I		
FB_PumpInWater_0.PumpInCol1		

11

Inputs	Block	Output
FB_Stabilization_Pitch_1.col2O	OR	IoConfig_Globals_Mapping.Col
FB_Stablilization_Roll.col2O		
simGVL.simBtnEmptyTanks		

12

Inputs	Block	Output
FB_Stabilization_Pitch_1.col2I	OR	IoConfig_Globals_Mapping.Col
FB_Stablilization_Roll.col2I		
FB_PumpInWater_0.PumpInCol2		

13

Inputs	Block	Output
FB_Stabilization_Pitch_1.col3O	OR	IoConfig_Globals_Mapping.Col
FB_Stablilization_Roll.col3O		
simGVL.simBtnEmptyTanks		

14

Inputs	Block	Output
FB_Stabilization_Pitch_1.col3I	OR	IoConfig_Globals_Mapping.Col
FB_Stablilization_Roll.col3I		
FB_PumpInWater_0.PumpInCol3		

15

Inputs	Block	Output
FB_Stabilization_Pitch_1.col4O	OR	IoConfig_Globals_Mapping.Col
FB_Stablilization_Roll.col4O		
simGVL.simBtnEmptyTanks		

16

Inputs	Block	Output
FB_Stabilization_Pitch_1.col4I	OR	IoConfig_Globals_Mapping.Col
FB_Stablilization_Roll.col4I		
FB_PumpInWater_0.PumpInCol4		

1In

2Out

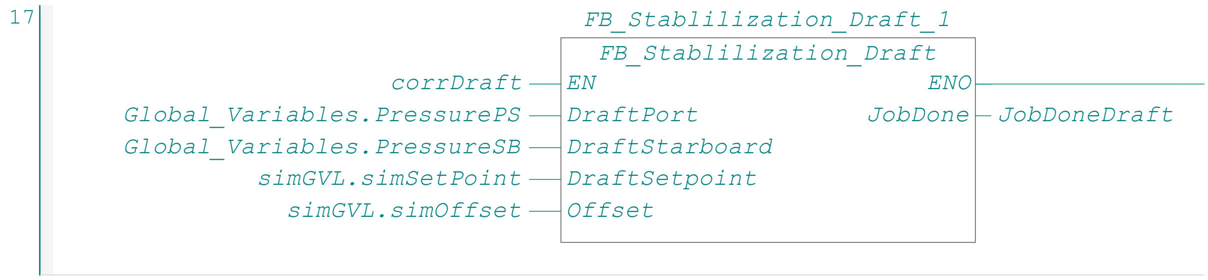
2In

3Out

3In

4Out

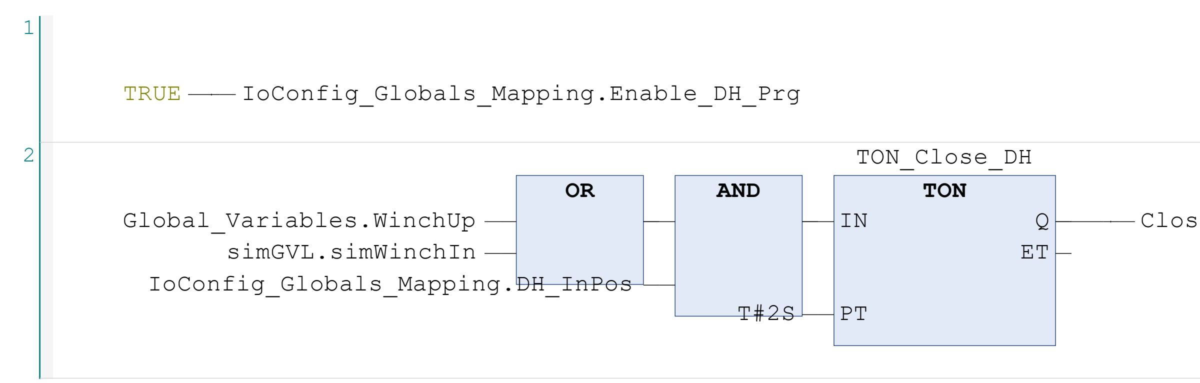
4In


17
FB_Stablilization_Draft_1
FB_Stablilization_Draft
corrDraft
EN
ENO
Global_Variables.PressurePS
DraftPort
JobDone
JobDoneDraft
Global_Variables.PressureSB
DraftStarboard
simGVL.simSetPoint
DraftSetpoint
simGVL.simOffset
Offset

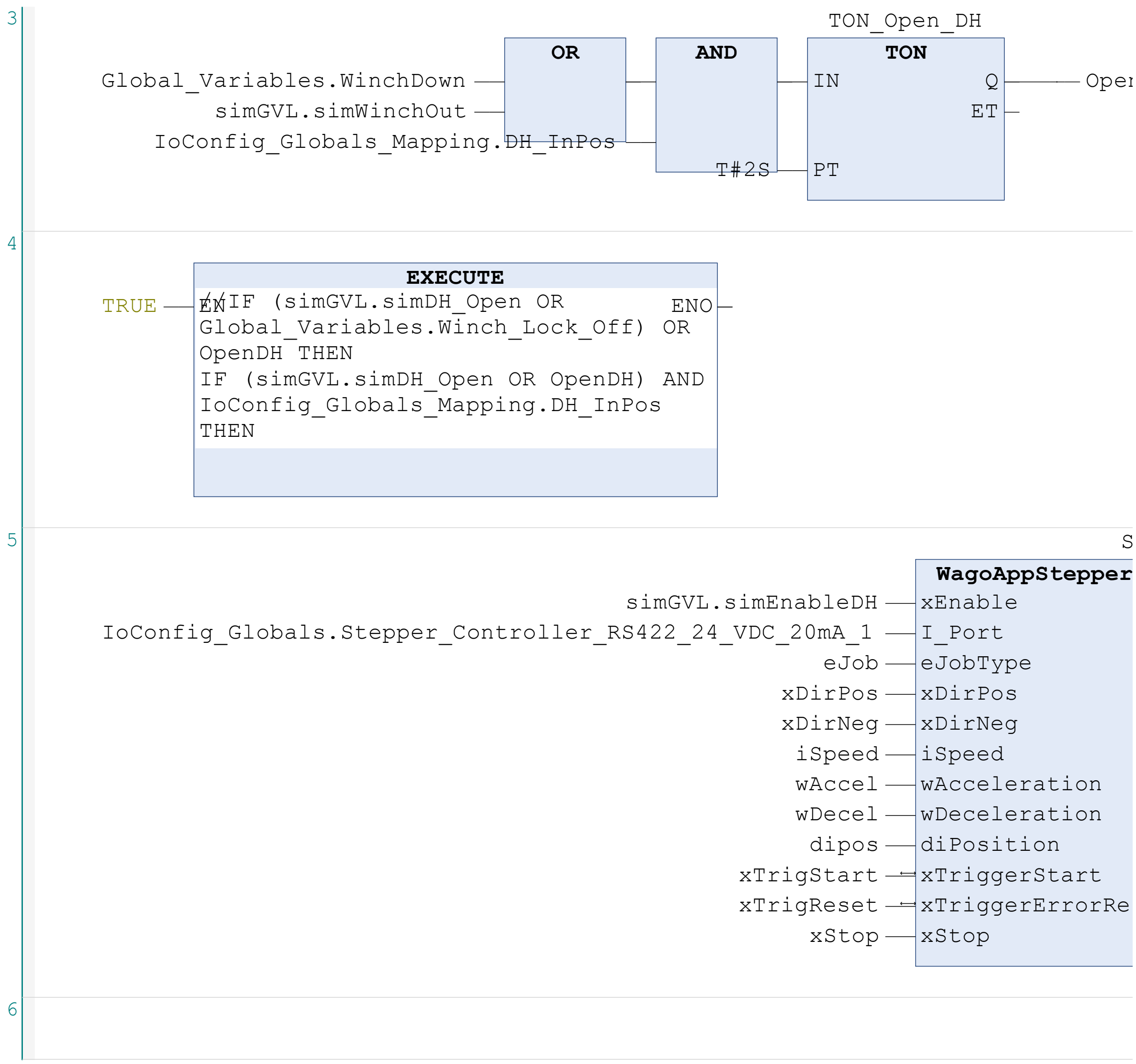

EXECUTE
EN
ENO

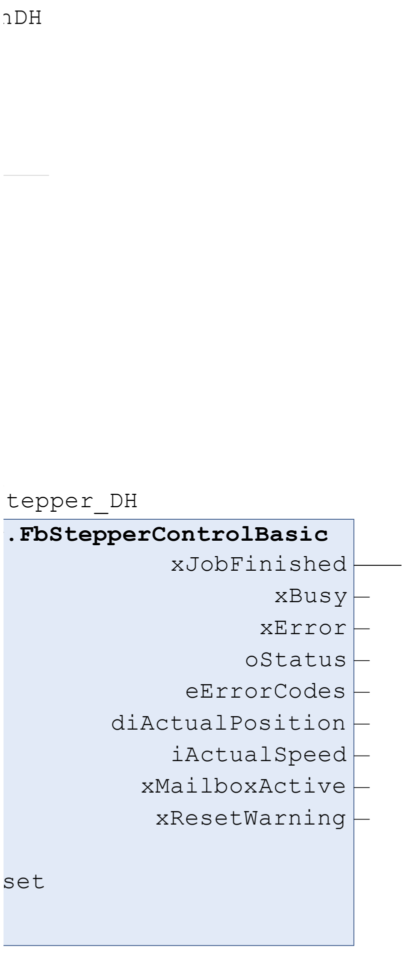
3.1.1.3.5 POU: POU_Stepper_DH

```
PROGRAM  POU_Stepper_DH
VAR
    Stepper_DH :  WagoAppStepper . FbStepperControlBasic ;
    FbMoveVelocity_Test  :  WagoAppStepper . FbMoveVelocity ;
    eJobType :  WagoAppStepper . FbMoveAbsolute ;
    eJob :  eMode  :=  EMODE . MoveAbsolute ;
    TON_Close_DH :  TON ;
    TON_Open_DH :  TON ;


    xEnable  :  BOOL ;
    xDirPos  : BOOL ;
    xDirNeg :  BOOL ;
    iSpeed :  INT  :=  10000 ;
    wAccel :  WORD  :=  32767 ;
    wDecel :  WORD  :=  32767 ;
    dipos  :  DINT ;
    xTrigStart  :  BOOL ;
    xTrigReset  :  BOOL ;
    xStop  :  BOOL ;

    CloseDH :  BOOL ;
    OpenDH :  BOOL ;
    test : BOOL ;

    pppp :  DINT ;
    openVar :  DINT ;
END_VAR

```



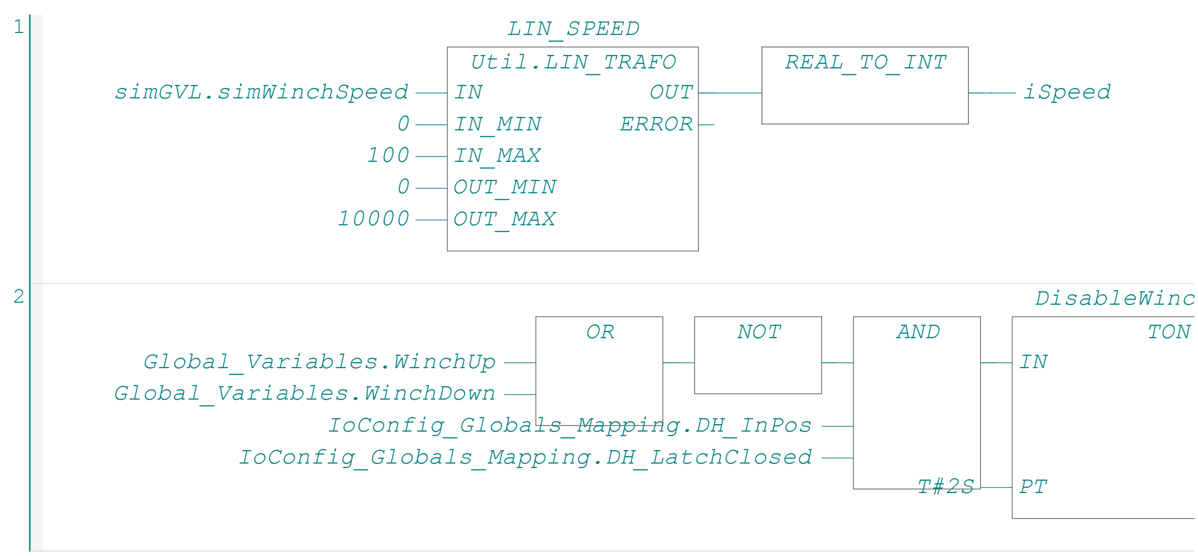
eDH

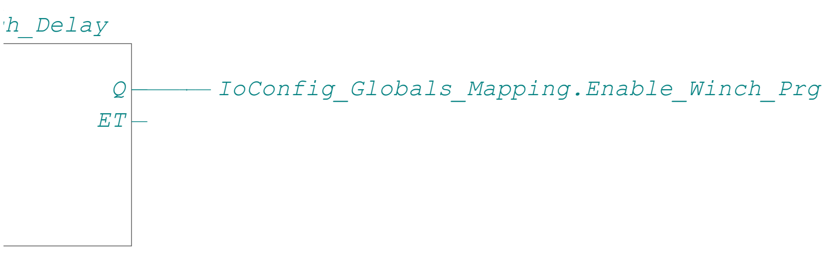

3
TON_Open_DH
OR
AND
TON
Global_Variables.WinchDown
simGVL.simWinchOut
IoConfig_Globals_Mapping.DH_InPos
T#2S
IN
Q
ET
PT
Ope
4
EXECUTE
TRUE
EN
ENO
//IF (simGVL.simDH_Open OR
Global_Variables.Winch_Lock_Off) OR
OpenDH THEN
IF (simGVL.simDH_Open OR OpenDH) AND
IoConfig_Globals_Mapping.DH_InPos
THEN
5
S
WagoAppStepper
simGVL.simEnableDH
IoConfig_Globals.Stepper_Controller_RS422_24_VDC_20mA_1
eJob
xDirPos
xDirNeg
iSpeed
wAccel
wDecel
dipos
xTrigStart
xTrigReset
xStop
xEnable
I_Port
eJobType
xDirPos
xDirNeg
iSpeed
wAcceleration
wDeceleration
diPosition
xTriggerStart
xTriggerErrorRe
xStop
6

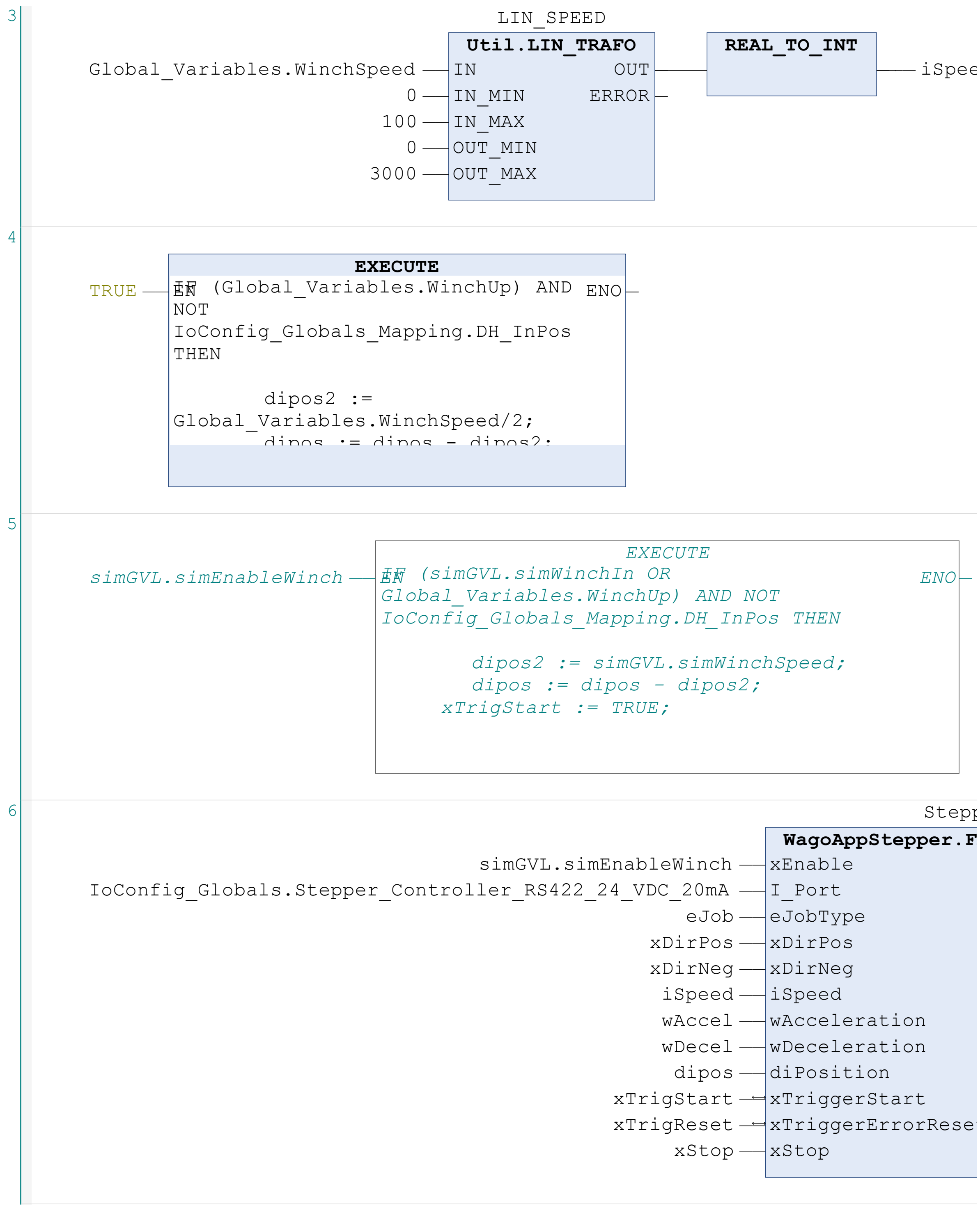

nDH
tepper_DH
.FbStepperControlBasic
xJobFinished
xBusy
xError
oStatus
eErrorCodes
diActualPosition
iActualSpeed
xMailboxActive
xResetWarning
set

3.1.1.3.6 POU: POU_Stepper_Winch

```
PROGRAM  POU_Stepper_Winch
VAR
    Stepper_Winch :  WagoAppStepper . FbStepperControlBasic ;
    eJobType :  WagoAppStepper . FbMoveAbsolute ;
    eJob :  eMode  :=  EMODE . MoveAbsolute ;
    LIN_SPEED  :  Util . LIN_TRAFO ;

    DisableWinch_Delay :  TON ;
    test :  BOOL ;

    xDirPos  : BOOL ;
    xDirNeg :  BOOL ;
    iSpeed :  INT ;
    wAccel :  WORD  :=  32767 ;
    wDecel :  WORD  :=  32767 ;
    dipos  :  DINT ;
    dipos2  :  DINT ;
    minValue  :  DINT ;
    xTrigStart  :  BOOL ;
    xTrigReset  :  BOOL ;
    xStop  :  BOOL ;
    xLampPayIn :  BOOL ;
    xLampPayOut : BOOL ;
END_VAR

```




h_Delay
Q
ET
IoConfig_Globals_Mapping.Enable_Winch_Prg


3
LIN_SPEED
Util.LIN_TRAFO
Global_Variables.WinchSpeed
IN
OUT
0
IN_MIN
ERROR
100
IN_MAX
0
OUT_MIN
3000
OUT_MAX
REAL_TO_INT
iSpee
4
EXECUTE
TRUE
EN
ENO
IF (Global_Variables.WinchUp) AND
NOT
IoConfig_Globals_Mapping.DH_InPos
THEN
dipos2 :=
Global_Variables.WinchSpeed/2;
5
EXECUTE
simGVL.simEnableWinch
EN
ENO
IF (simGVL.simWinchIn OR
Global_Variables.WinchUp) AND NOT
IoConfig_Globals_Mapping.DH_InPos THEN
dipos2 := simGVL.simWinchSpeed;
dipos := dipos - dipos2;
xTrigStart := TRUE;
6
Stepp
WagoAppStepper.F
simGVL.simEnableWinch
xEnable
IoConfig_Globals.Stepper_Controller_RS422_24_VDC_20mA
I_Port
eJob
eJobType
xDirPos
xDirPos
xDirNeg
xDirNeg
iSpeed
iSpeed
wAccel
wAcceleration
wDecel
wDeceleration
dipos
diPosition
xTrigStart
xTriggerStart
xTrigReset
xTriggerErrorRese
xStop
xStop

ed


per_Winch
bStepperControlBasic
xJobFinished
xBusy
xError
oStatus
eErrorCodes
diActualPosition
iActualSpeed
xMailboxActive
xResetWarning
t

3.1.1.3.7 POU: POU_TestPumps

```
PROGRAM  POU_TestPumps
VAR
    Col1In_Sim :  BOOL ;
    Col1Out_Sim :  BOOL ;
    Col2In_Sim  : BOOL ;
    Col2Out_Sim :  BOOL ;
    Col3In_Sim :  BOOL ;
    Col3Out_Sim :  BOOL ;
    Col4In_Sim :  BOOL ;
    Col4Out_Sim :  BOOL ;
END_VAR

```

```
1
Col1In_Sim —— IoConfig_Globals_Mapping.Col1In
2
Col1Out_Sim —— IoConfig_Globals_Mapping.Col1Out
3
Col2In_Sim —— IoConfig_Globals_Mapping.Col2In
4
Col2Out_Sim —— IoConfig_Globals_Mapping.Col2Out
5
Col3In_Sim —— IoConfig_Globals_Mapping.Col3In
6
Col3Out_Sim —— IoConfig_Globals_Mapping.Col3Out
7
Col4In_Sim —— IoConfig_Globals_Mapping.Col4In
```

8

```
Col4Out_Sim —— IoConfig_Globals_Mapping.Col4Out
```

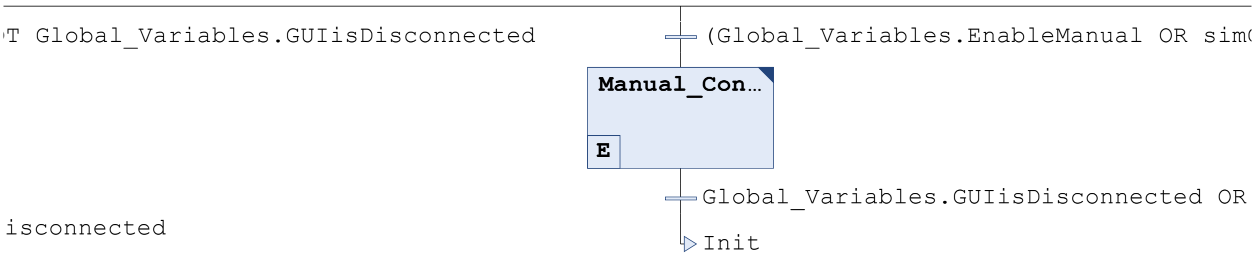
3.1.1.3.8 POU: POU_ThrusterControl

```
PROGRAM POU_ThrusterControl
VAR
    X2_Dir : REAL ;

P_distX : REAL := 100 ;
I_distX : REAL ;
D_distX : REAL ;

P_distX2 : REAL := 100 ;
I_distX2 : REAL ;
D_distX2 : REAL ;

P_distY : REAL := 100 ;
I_distY : REAL ;
D_distY : REAL ;
ThPS_DP : Util . LIN_TRAFO ;

DirInvertX2 : BOOL := TRUE ;
    DirInvertX : BOOL ;
    DirInvertY : BOOL ;
    DirInvert : BOOL ;
    // Init
    Autopilot_On : BOOL ;
    Waypoint_OK : BOOL ;

    // Calculate Angle Variables
    angle_AutopilotCalculated : LREAL ;
    Setpoint_Angle_Autopilot : INT ;
    RotationPID : INT ;

    // Autopilot Variables Angle
    PID_Angle_Autopilot : WagoAppBuildingHVAC . FbPIDController ;
    LIN_TRAFO_Map_Angle : LIN_TRAFO ;
    TimerCalcNewAngle_Autopilot : TON ;
    P_Ang_Autopilot : REAL := 100 ;
    I_Ang_Autopilot : REAL ;
    D_Ang_Autopilot : REAL ;
    presentON : BOOL := TRUE ;
    presentOFF : BOOL := TRUE ;
    SetTimerCalcNewAngle_Autopilot : BOOL ;
    CalcNewAng : BOOL ;
    TurnForce_Autopilot : REAL ;


    // Move Plaform Variables Distance Autopilot
    PID_Distance_Autopilot : WagoAppBuildingHVAC . FbPIDController ;
    P_dist : REAL := 2000 ;
    I_dist : REAL ;
```

```
        D_dist  : REAL ;
        deadsone_dist :  REAL ;
        ref : REAL ;
        SpeedValue_Autopilot :  REAL ;

        //Simulator variables
        FB_Simulate_ThrusterDir_0  :  FB_Simulate_ThrusterDir ;

        //Manual Control
        Forward_Out :  REAL ;
        Backward_Out :  REAL ;
        LeftPivot_Out :  REAL ;
        RightPivot_Out :  REAL ;
        Left_Out :  REAL ;
        Right_Out :  REAL ;
        PS_Value :  REAL ;
        SB_Value :  REAL ;
        FB_ThrusterControl_1 :  FB_ThrusterScaling ;
        FB_Thruster_Values_Forward :  FB_Thruster_Values ;
        FB_Thruster_Values_Backward :  FB_Thruster_Values ;
        FB_Thruster_Values_LeftPivot :  FB_Thruster_Values ;
        FB_Thruster_Values_RightPivot :  FB_Thruster_Values ;
        FB_Thruster_Values_Left :  FB_Thruster_Values ;
        FB_Thruster_Values_Right :  FB_Thruster_Values ;
        Thruster_Speed  :  REAL ;

        // Distance Variables
        Radius  :  LREAL  :=  6372.795477598 ;
        distance  :  LREAL ;
        FB_CalcAng :  FB_CalcAngle ;
        FB_Dist  :  FB_CalcDistance ;

        // DP mode Variables
        X_Dir :  REAL ;
        Y_Dir :  REAL ;
        PID_Heading :  WagoAppBuildingHVAC . FbPIDController ;
        PID_X_Dir  :  WagoAppBuildingHVAC . FbPIDController ;
        PID_X2_Dir  :  WagoAppBuildingHVAC . FbPIDController ;
        PID_Y_Dir  :  WagoAppBuildingHVAC . FbPIDController ;
        TON_Heading_DP_OK :  TON ;
        TON_Heading_Check :  TON ;
        XLat_Coordinate :  REAL ;
        YLong_Coordinate :  REAL ;
        Heading_OK :  BOOL ;
        RotationPID_DP :  INT ;
        TurnValue_DP :  REAL ;
        Adjust_Heading :  BOOL ;
        RotationCheck_DP :  INT ;
        Longitude_DP  :  REAL ;
        Latitude_DP :  REAL ;

```

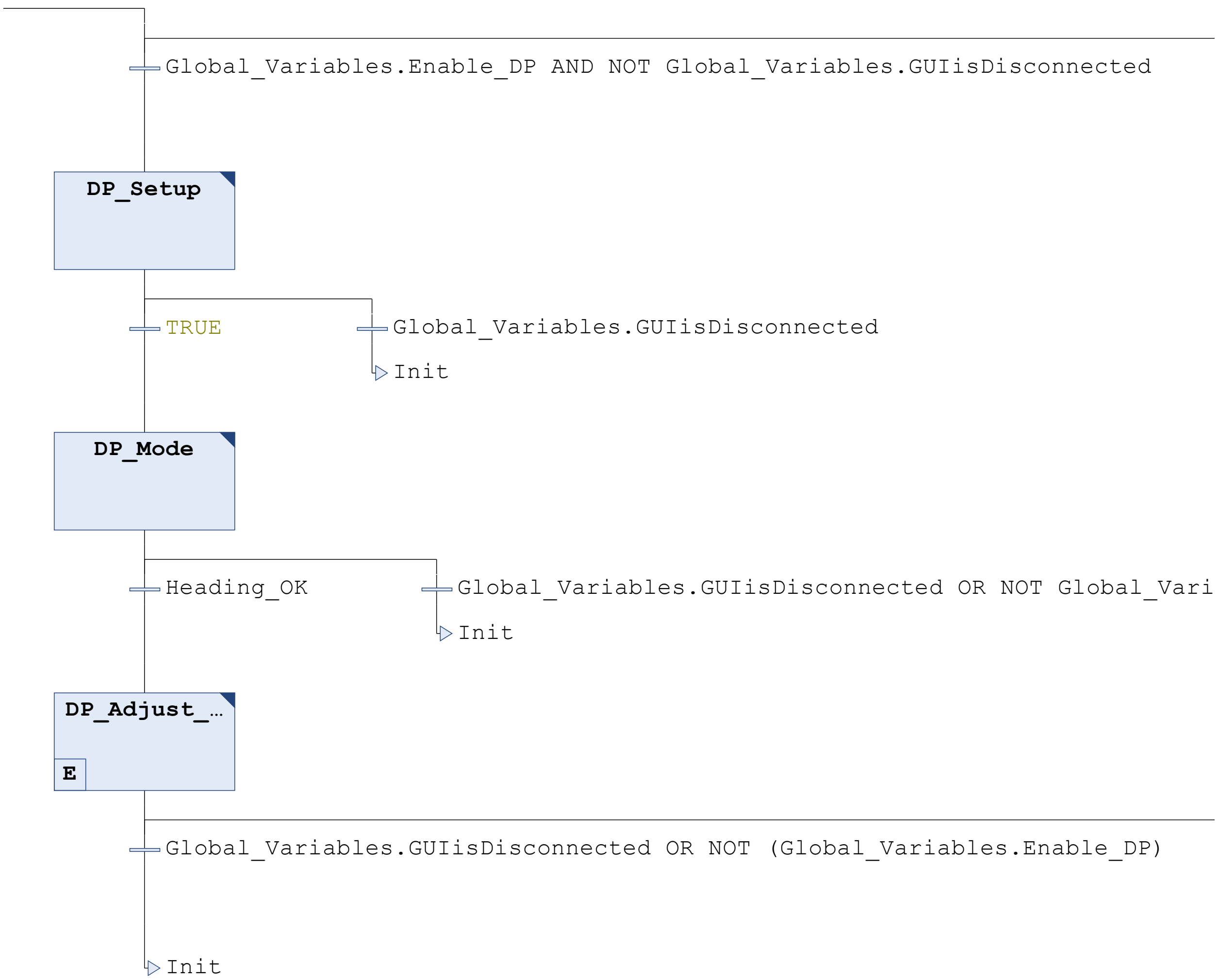
```
END_VAR

```


Init
(simGVL.simAutopilotON OR Global_Variables.EnableAuto) AND Waypoint_OK AND NO
Autopilot_…
TRUE
Global_Variables.GUIisDisconnected OR Global_Variables.RPIisD
Init
Calculate_…


'T Global_Variables.GUIisDisconnected
(Global_Variables.EnableManual OR sim(
Manual_Con…
E
Global_Variables.GUIisDisconnected OR
isconnected
Init

```
GVL.simBtnBool) AND NOT (simGVL.simAutopilotON OR Global_Variables.GUIisDisconnected)
```

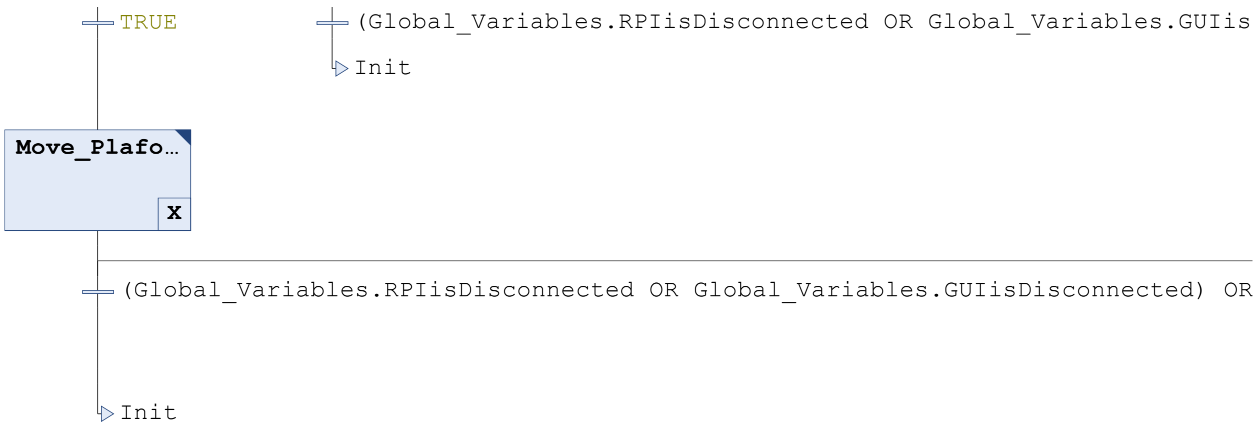
```
NOT (Global_Variables.EnableManual)
```


Global_Variables.Enable_DP AND NOT Global_Variables.GUIisDisconnected
DP_Setup
TRUE
Global_Variables.GUIisDisconnected
Init
DP_Mode
Heading_OK
Global_Variables.GUIisDisconnected OR NOT Global_Vari
Init
DP_Adjust_...
E
Global_Variables.GUIisDisconnected OR NOT (Global_Variables.Enable_DP)
Init

Global_Variables.GUIisDisconnected
Init

ables.Enable_DP

Adjust_Heading
DP_Mode


TRUE
(Global_Variables.RPIisDisconnected OR Global_Variables.GUIis
Init
Move_Plafo…
X
(Global_Variables.RPIisDisconnected OR Global_Variables.GUIisDisconnected) OR
Init

```
Disconnected) OR NOT (Global_Variables.EnableAuto)
```

```
NOT (Global_Variables.EnableAuto)          CalcNewAng
                                           Calculate_Angle
```

3.1.1.3.8 POU: POU_ThrusterControl

3.1.1.3.8.1 Action: Autopilot_Setup_active

1

TRUE — EN — **EXECUTE** — ENO

```
PID_Angle_Autopilot.typConfigParameters.xChangeInDirection := DirInvert;
PID_Angle_Autopilot.typConfigParameters.rOutputMin := 0;
```

2

TRUE — EN — **EXECUTE** — ENO

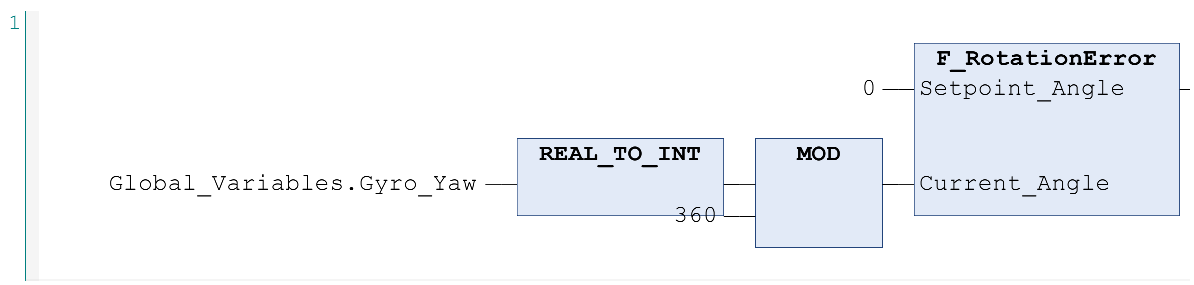
```
PID_Distance_Autopilot.typConfigParameters.rOutputMin := 0;
PID_Distance_Autopilot.typConfigParameters.rOutputMax := 10000;
PID_Distance_Autopilot.typCon
```

3

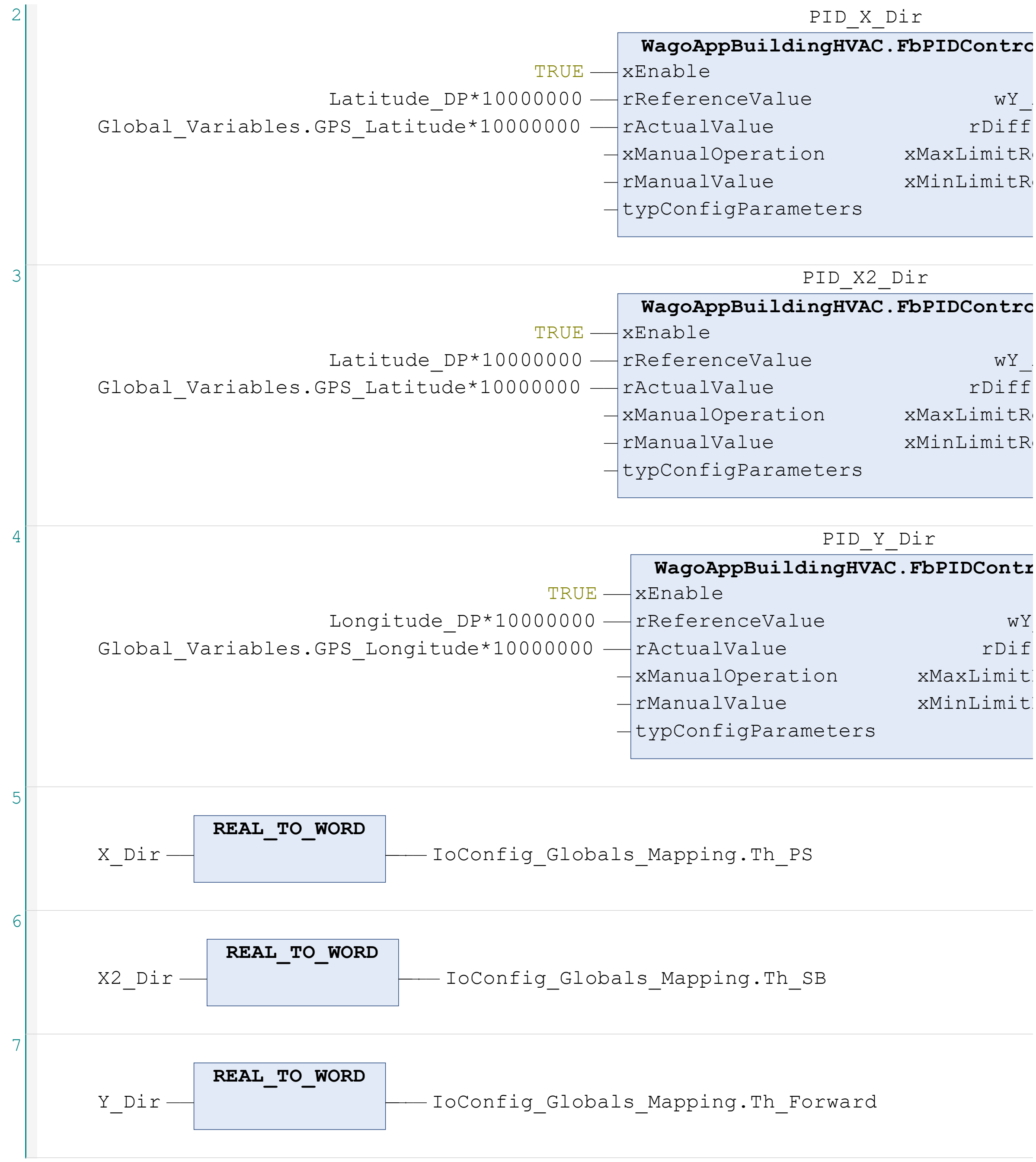
3.1.1.3.8.2 Action: Calculate_Angle_active

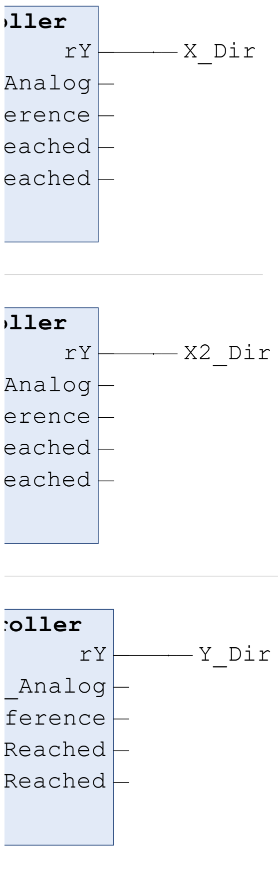
```
//y := (WagoAppMath.sin_L(phi:= simGVL.simWaypointXLon - simGVL.simGPSLongitude)) * WagoAppMath.cos_L(phi:= simGVL.simWaypointYLat);
//x := WagoAppMath.cos_L(phi:= Global_Variables.GPS_Latitude) * WagoAppMath.sin_L(phi:= simGVL.simWaypointYLat) - WagoAppMath.sin_L(phi:= Global_Variables.GPS_Latitude) * WagoAppMath.cos_L(phi:= simGVL.simWaypointYLat) * WagoAppMath.cos_L(phi:= simGVL.simWaypointXLon - Global_Variables.GPS_Longitude);
//bearing := WagoAppMath.arcTan2(y:=y , x:= x);
//Angle2 := WagoAppMath.radiantToAngle(lrRadiant:= bearing);
//Angle := WagoAppMath.angleToDegree_L(phi:= bearing);
SetTimerCalcNewAngle_Autopilot := FALSE ;
FB_CalcAng (
    LatA := Global_Variables . GPS_Latitude ,
    LatB := Global_Variables . GUI_Latitude ,
    LonA := Global_Variables . GPS_Longitude ,
    LonB := Global_Variables . GUI_Longitude ,
    Angle => angle_AutopilotCalculated ) ;

// Distance
FB_Dist (
    LatA := Global_Variables . GPS_Latitude ,
    LatB := Global_Variables . GUI_Latitude ,
    LonA := Global_Variables . GPS_Longitude ,
    LonB := Global_Variables . GUI_Longitude ,
    Distance => distance ) ;
```

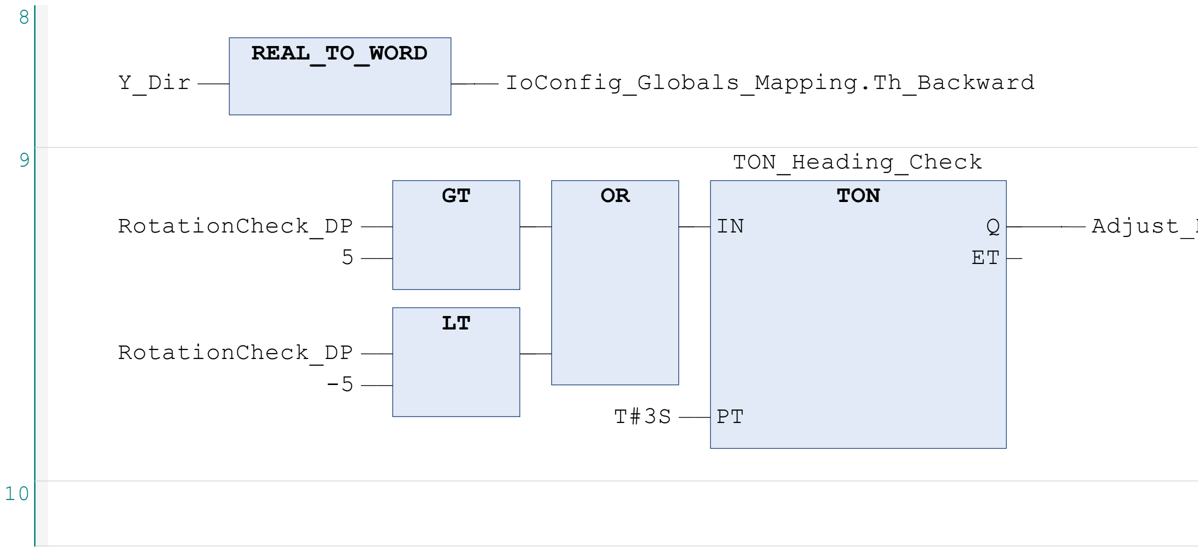
3.1.1.3.8.3 Action: DP_Adjust_XY_active



—— RotationCheck_DP


2
PID_X_Dir
WagoAppBuildingHVAC.FbPIDContro
TRUE
xEnable
Latitude_DP*10000000
rReferenceValue
wY_
Global_Variables.GPS_Latitude*10000000
rActualValue
rDiff
xManualOperation
xMaxLimitR
rManualValue
xMinLimitR
typConfigParameters
3
PID_X2_Dir
WagoAppBuildingHVAC.FbPIDContro
TRUE
xEnable
Latitude_DP*10000000
rReferenceValue
wY_
Global_Variables.GPS_Latitude*10000000
rActualValue
rDiff
xManualOperation
xMaxLimitR
rManualValue
xMinLimitR
typConfigParameters
4
PID_Y_Dir
WagoAppBuildingHVAC.FbPIDContr
TRUE
xEnable
Longitude_DP*10000000
rReferenceValue
wY
Global_Variables.GPS_Longitude*10000000
rActualValue
rDif
xManualOperation
xMaxLimit
rManualValue
xMinLimit
typConfigParameters
5
X_Dir
REAL_TO_WORD
IoConfig_Globals_Mapping.Th_PS
6
X2_Dir
REAL_TO_WORD
IoConfig_Globals_Mapping.Th_SB
7
Y_Dir
REAL_TO_WORD
IoConfig_Globals_Mapping.Th_Forward


oller
rY
X_Dir
Analog
erence
eached
eached
oller
rY
X2_Dir
Analog
erence
eached
eached
roller
rY
Y_Dir
_Analog
ference
Reached
Reached


8
Y_Dir
REAL_TO_WORD
IoConfig_Globals_Mapping.Th_Backward
9
TON_Heading_Check
RotationCheck_DP
5
GT
OR
TON
IN
Q
ET
Adjust_
RotationCheck_DP
-5
LT
T#3S
PT
10

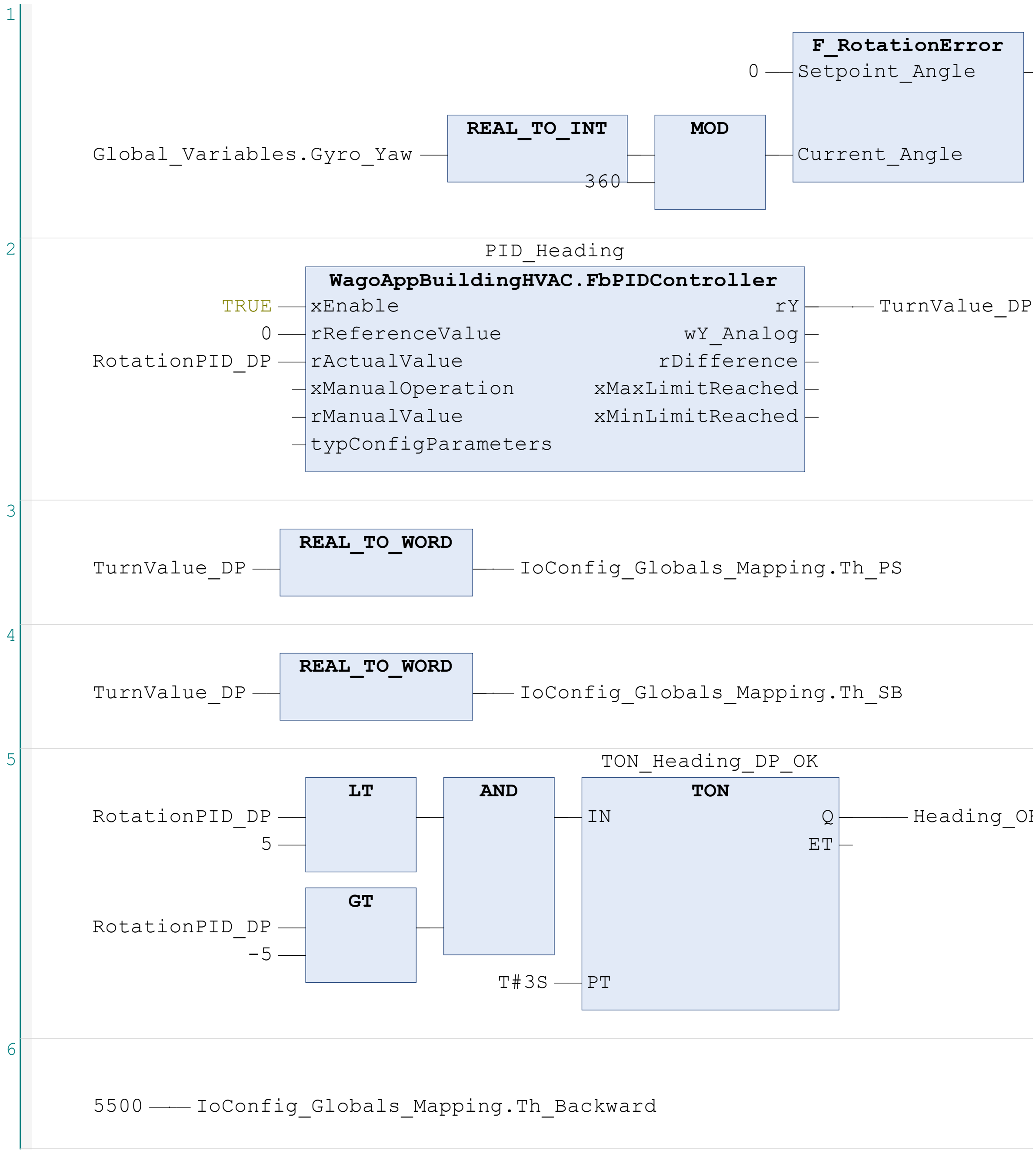
3.1.1.3.8.3 Action: DP_Adjust_XY_active

Heading

3.1.1.3.8.4 Action: DP_Adjust_XY_entry

```
Global_Variables.GPS_Longitude ← XLat_Coordinate
Global_Variables.GPS_Latitude ← YLong_Coordinate
```

3.1.1.3.8.5 Action: DP_Mode_AdjustHeadingactive



3.1.1.3.8.5 Action: DP_Mode_AdjustHeadingactive

—— RotationPID_DP

K

3.1.1.3.8.5 Action: DP_Mode_AdjustHeadingactive

```
5500 —— IoConfig_Globals_Mapping.Th_Forward
```

3.1.1.3.8.5 Action: DP_Mode_AdjustHeadingactive

3.1.1.3.8.6 Action: DP_Setup_active

1

2

TRUE — EN **EXECUTE** ENO —

```
        PID_Heading.typConfigPa
rameters.rOutputMin := 0;
        PID_Heading.typConfigPa
rameters.rOutputMax := 10000;
```

3

TRUE — EN **EXECUTE** ENO —

```
        PID_X_DIR.typConfigParamet
ers.xChangeInDirection :=
DirInvertX;
        PID_X_DIR.typConfigParamet
ers.rOutputMin := 0;
```

4

TRUE — EN **EXECUTE** ENO —

```
        PID_X2_DIR.typConfigParamet
ers.xChangeInDirection :=
DirInvertX2;
        PID_X2_DIR.typConfigParamet
ers.rOutputMin := 0;
```

5

6

7

TRUE — EN **EXECUTE** ENO —

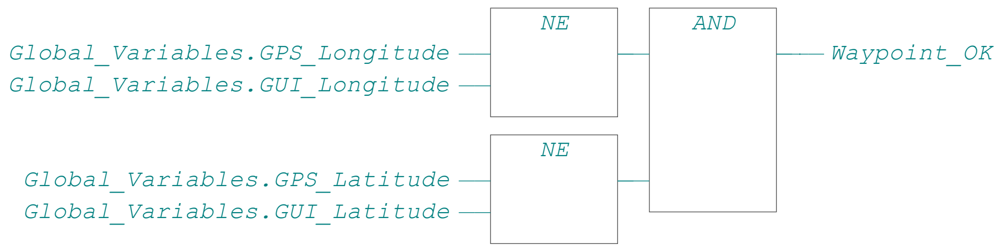
```
        PID_Y_DIR.typConfigParamet
ers.xChangeInDirection :=
DirInvertY;
        PID_Y_DIR.typConfigParamet
ers.rOutputMin := 0;
```

3.1.1.3.8.6 Action: DP_Setup_active

```
Global_Variables.GPS_Longitude —— Longitude_DP

Global_Variables.GPS_Latitude —— Latitude_DP
```

3.1.1.3.8.7 Action: Init_active

1



2

TRUE —— Waypoint_Ok

3

5500 —— IoConfig_Globals_Mapping.Th_Forward

4

5500 —— IoConfig_Globals_Mapping.Th_Backward

5

5500 —— IoConfig_Globals_Mapping.Th_PS

6

5500 —— IoConfig_Globals_Mapping.Th_SB

3.1.1.3.8.8 Action: Manual_Control_active

1

Global_Variables.ThrusterSpeed — INT_TO_REAL — Thruster_Speed

2

simGVL.simThrustValue —— Thruster_Speed

3

(Global_Variables.EnableManual OR simGVL.simBtnBool) AND NOT simGVL.simAut
Global_Variables.FwdMotion OR simGVL.s
Global_Variables.BackMotion OR simGVL.si
Global_Variables.CounterClockMotion OR simGVL.sim
Global_Variables.ClockWMotion OR simGVL.simR
Global_Variables.LeftMotion OR simGV
Global_Variables.RightMotion OR simGVL
Thrus

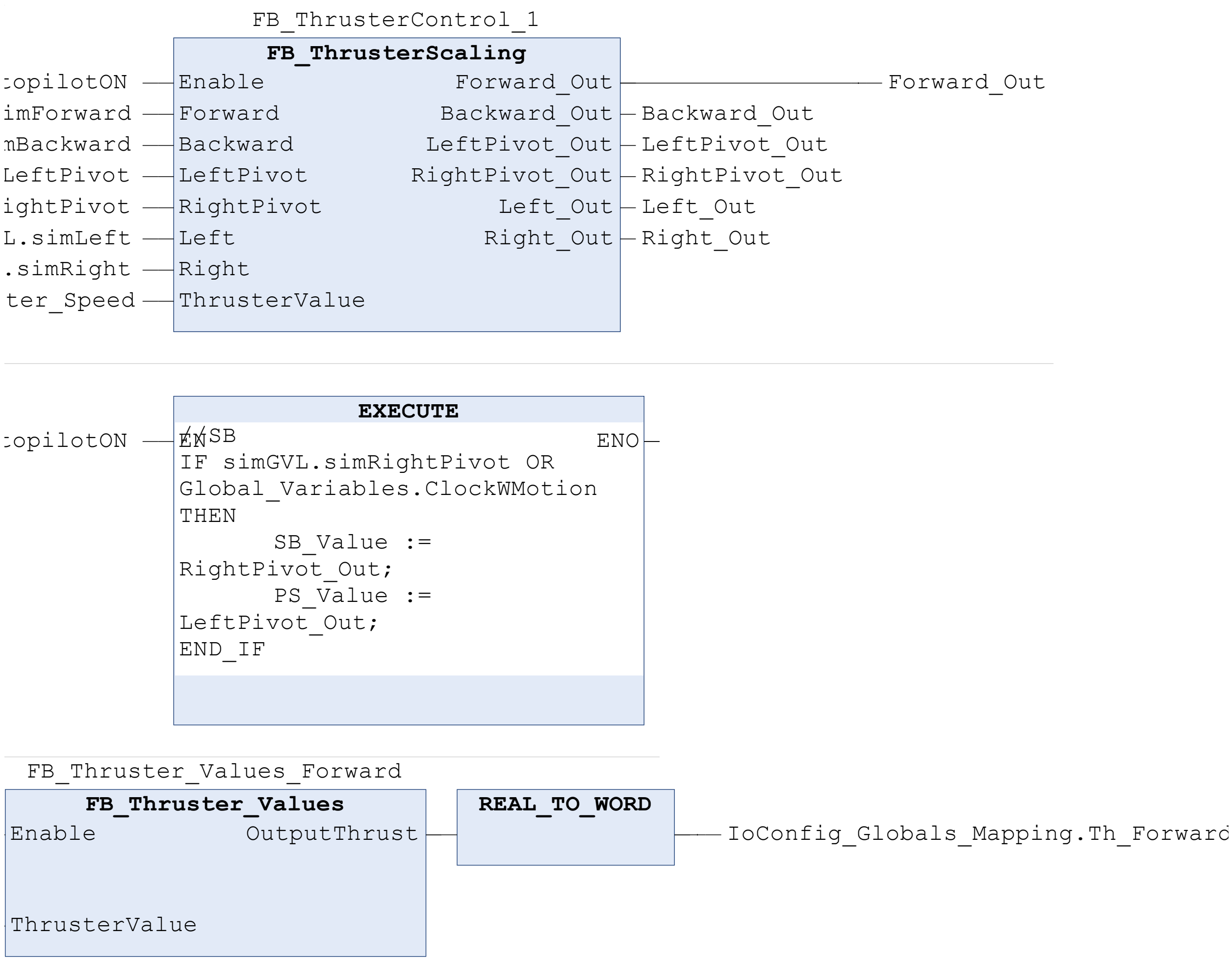
4

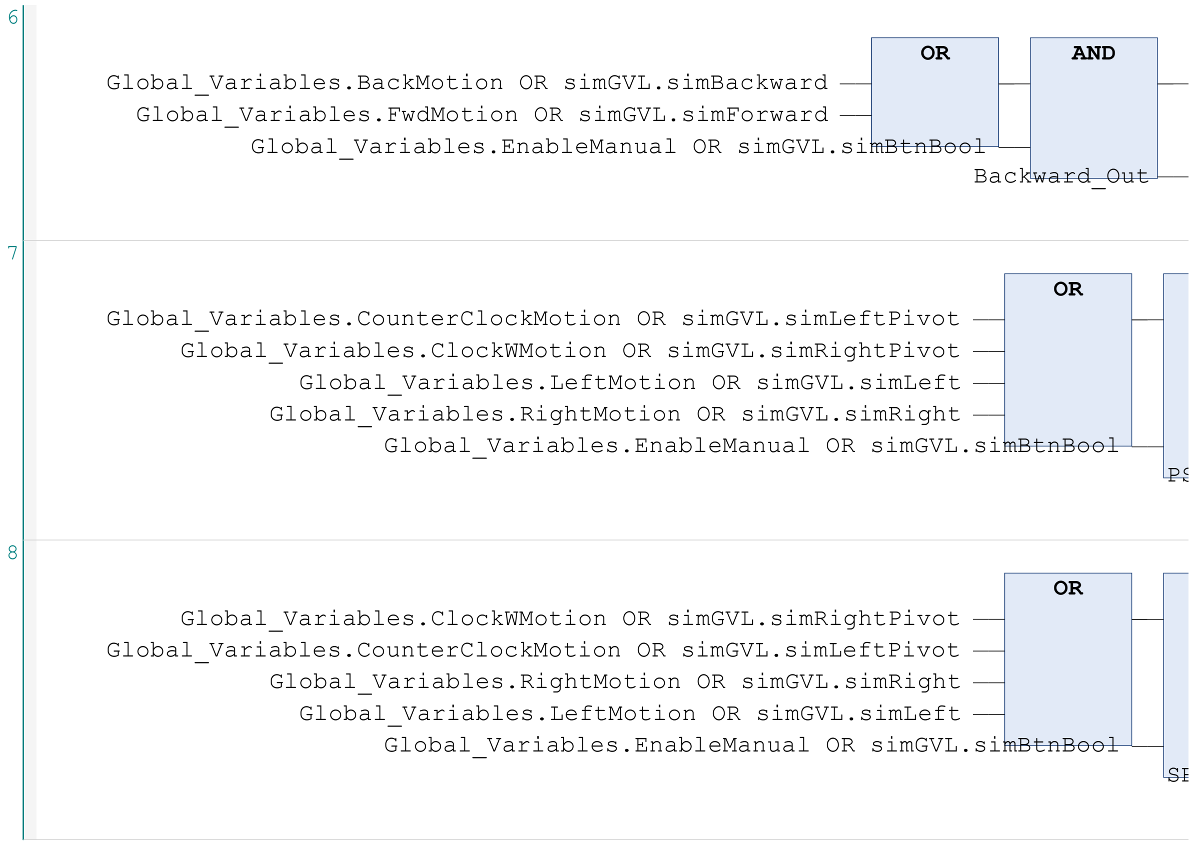
(Global_Variables.EnableManual OR simGVL.simBtnBool) AND NOT simGVL.simAut

5

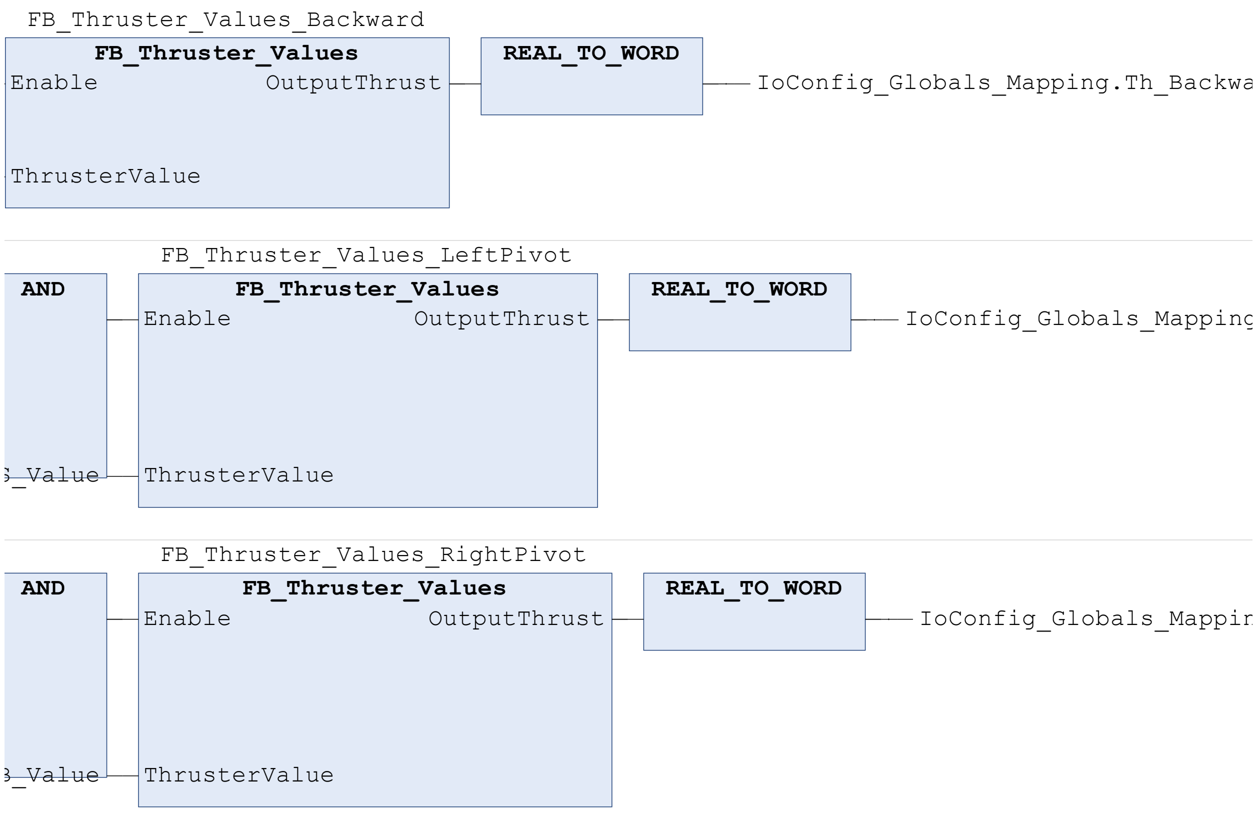
OR
AND
Global_Variables.FwdMotion OR simGVL.simForward
Global_Variables.BackMotion OR simGVL.simBackward
Global_Variables.EnableManual OR simGVL.simBtnBool
Forward_Out

3.1.1.3.8.8 Action: Manual_Control_active




6
OR
AND
Global_Variables.BackMotion OR simGVL.simBackward
Global_Variables.FwdMotion OR simGVL.simForward
Global_Variables.EnableManual OR simGVL.simBtnBool
Backward_Out
7
OR
Global_Variables.CounterClockMotion OR simGVL.simLeftPivot
Global_Variables.ClockWMotion OR simGVL.simRightPivot
Global_Variables.LeftMotion OR simGVL.simLeft
Global_Variables.RightMotion OR simGVL.simRight
Global_Variables.EnableManual OR simGVL.simBtnBool
8
OR
Global_Variables.ClockWMotion OR simGVL.simRightPivot
Global_Variables.CounterClockMotion OR simGVL.simLeftPivot
Global_Variables.RightMotion OR simGVL.simRight
Global_Variables.LeftMotion OR simGVL.simLeft
Global_Variables.EnableManual OR simGVL.simBtnBool

3.1.1.3.8.8 Action: Manual_Control_active



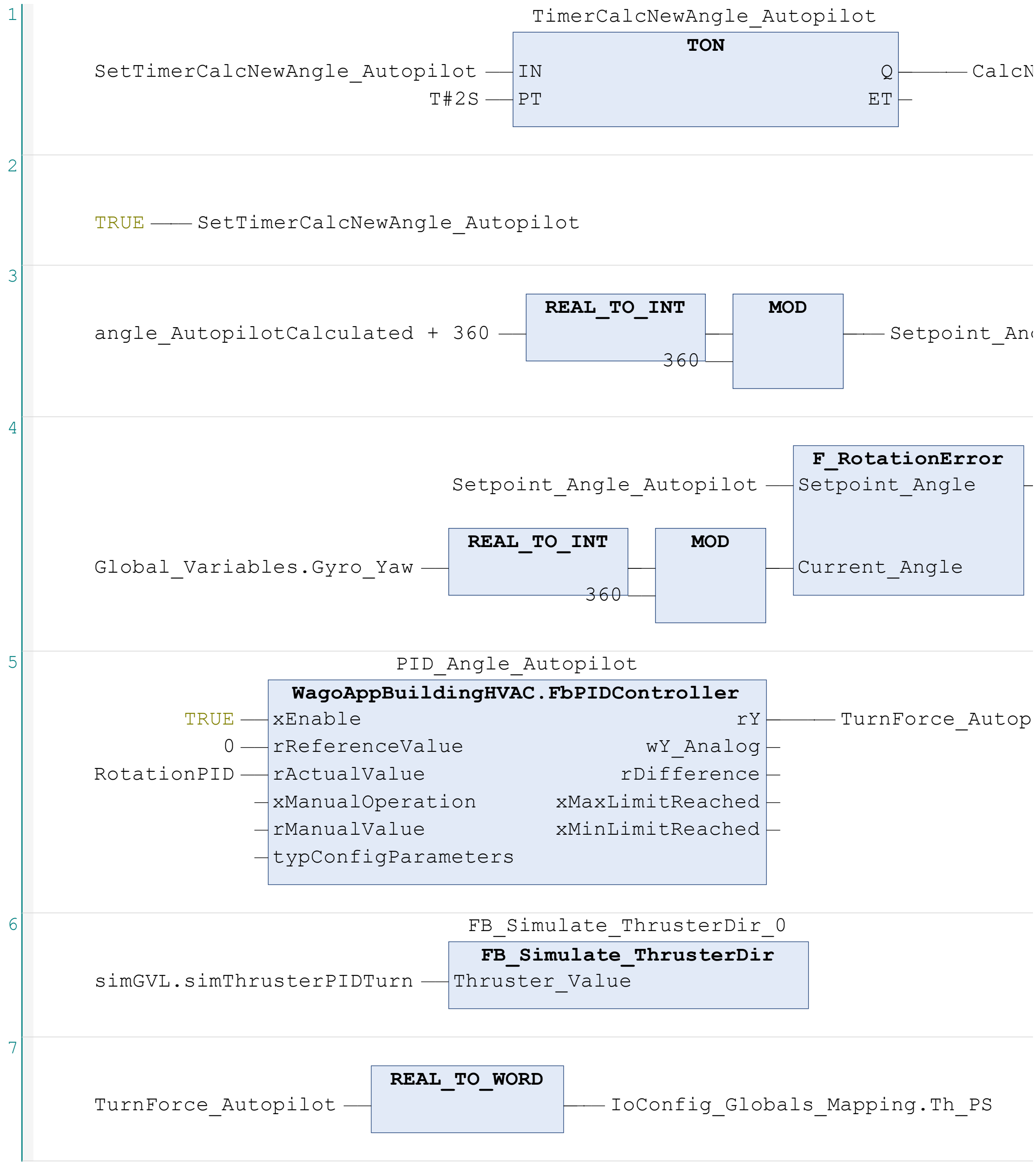
ırd

ȷ.Th_PS

ıg.Th_SB

3.1.1.3.8.9 Action: Manual_Control_entry

3.1.1.3.8.10 Action: Move_Plaform_active



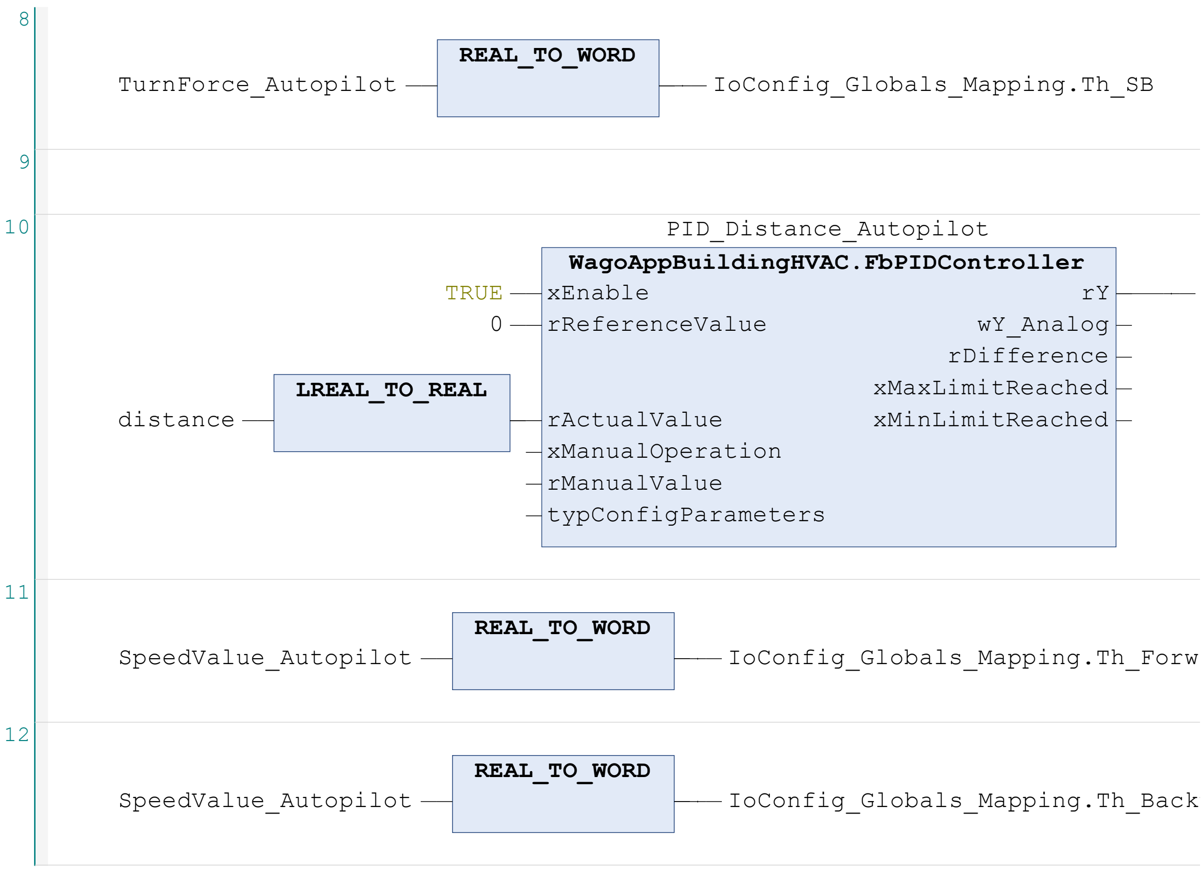
ewAng

gle_Autopilot

—— RotationPID

ilot

3.1.1.3.8.10 Action: Move_Plaform_active



SpeedValue_Autopilot

ard

ward

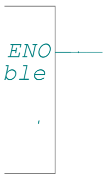
3.1.1.3.8.11 Action: Move_Plaform_exit

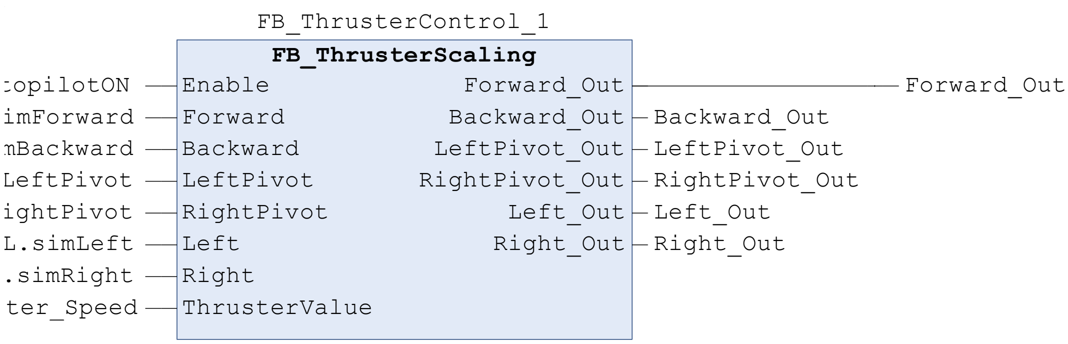
```
SetTimerCalcNewAngle_Autopilot := FALSE ;

```

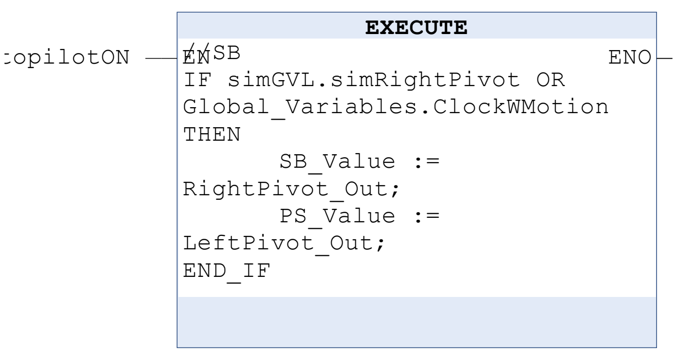
3.1.1.3.9 POU: POU_ThrusterController_Manual

```
PROGRAM POU_ThrusterController_Manual
VAR

    TestGraph : INT ;
    TestGraphSecond : INT ;


    Forward_Out : REAL ;
    Backward_Out : REAL ;
    LeftPivot_Out : REAL ;
    RightPivot_Out : REAL ;
    Left_Out : REAL ;
    Right_Out : REAL ;

    PS_Value : REAL ;
    SB_Value : REAL ;

    FB_ThrusterControl_1 : FB_ThrusterScaling ;
    FB_Thruster_Values_Forward : FB_Thruster_Values ;
    FB_Thruster_Values_Backward : FB_Thruster_Values ;

    FB_Thruster_Values_LeftPivot : FB_Thruster_Values ;
    FB_Thruster_Values_RightPivot : FB_Thruster_Values ;

    FB_Thruster_Values_Left : FB_Thruster_Values ;
    FB_Thruster_Values_Right : FB_Thruster_Values ;

    Thruster_Speed : REAL ;
END_VAR
//IF (Enable = TRUE) THEN
//        OutputThrust := ThrusterValue;
//        ELSE
//              OutputThrust := 0;
//END_IF

```

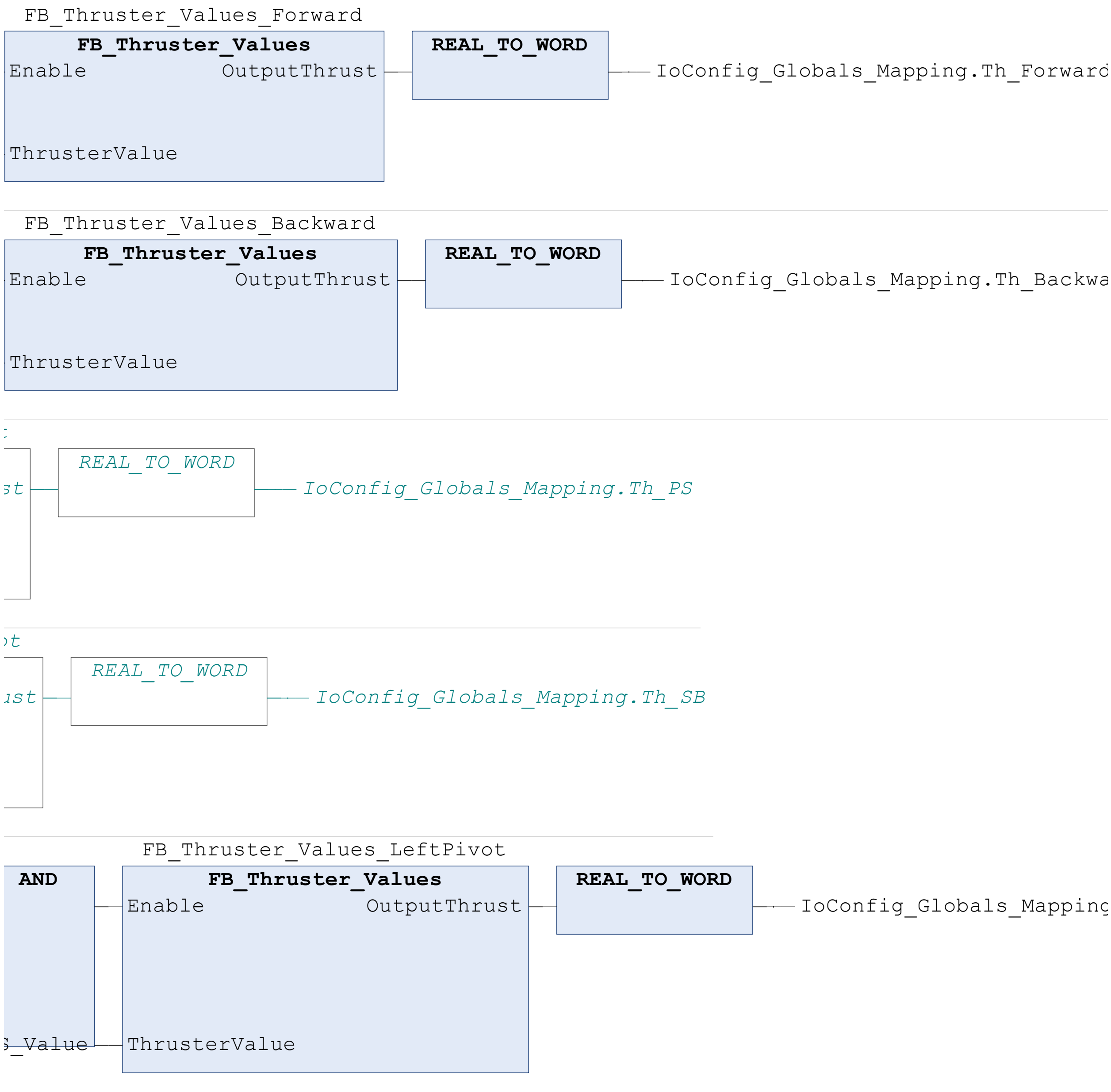
```
1
Global_Variables.EnableManual OR simGVL.simBtnBool —— EXECUTE
                                                      EN
                                                      IF
                                                      Global_Variables.Ena
                                                      Manual THEN

2
Global_Variables.ThrusterSpeed —— INT_TO_REAL —— Thruster_Speed

3
simGVL.simThrustValue —— Thruster_Speed

4
(Global_Variables.EnableManual OR simGVL.simBtnBool) AND NOT simGVL.simAut
                                     Global_Variables.FwdMotion OR simGVL.s
                                   Global_Variables.BackMotion OR simGVL.si
                          Global_Variables.CounterClockMotion OR simGVL.sim
                               Global_Variables.ClockWMotion OR simGVL.simR
                                       Global_Variables.LeftMotion OR simGV
                                     Global_Variables.RightMotion OR simGVL
                                                                      Thrus

5
(Global_Variables.EnableManual OR simGVL.simBtnBool) AND NOT simGVL.simAut
```


ENO
ble


FB_ThrusterControl_1
FB_ThrusterScaling
:opilotON
imForward
mBackward
LeftPivot
ightPivot
L.simLeft
.simRight
ter_Speed
Enable
Forward
Backward
LeftPivot
RightPivot
Left
Right
ThrusterValue
Forward_Out
Backward_Out
LeftPivot_Out
RightPivot_Out
Left_Out
Right_Out
Forward_Out
Backward_Out
LeftPivot_Out
RightPivot_Out
Left_Out
Right_Out


EXECUTE
:opilotON
EN
ENO
//SB
IF simGVL.simRightPivot OR
Global_Variables.ClockWMotion
THEN
SB_Value :=
RightPivot_Out;
PS_Value :=
LeftPivot_Out;
END_IF

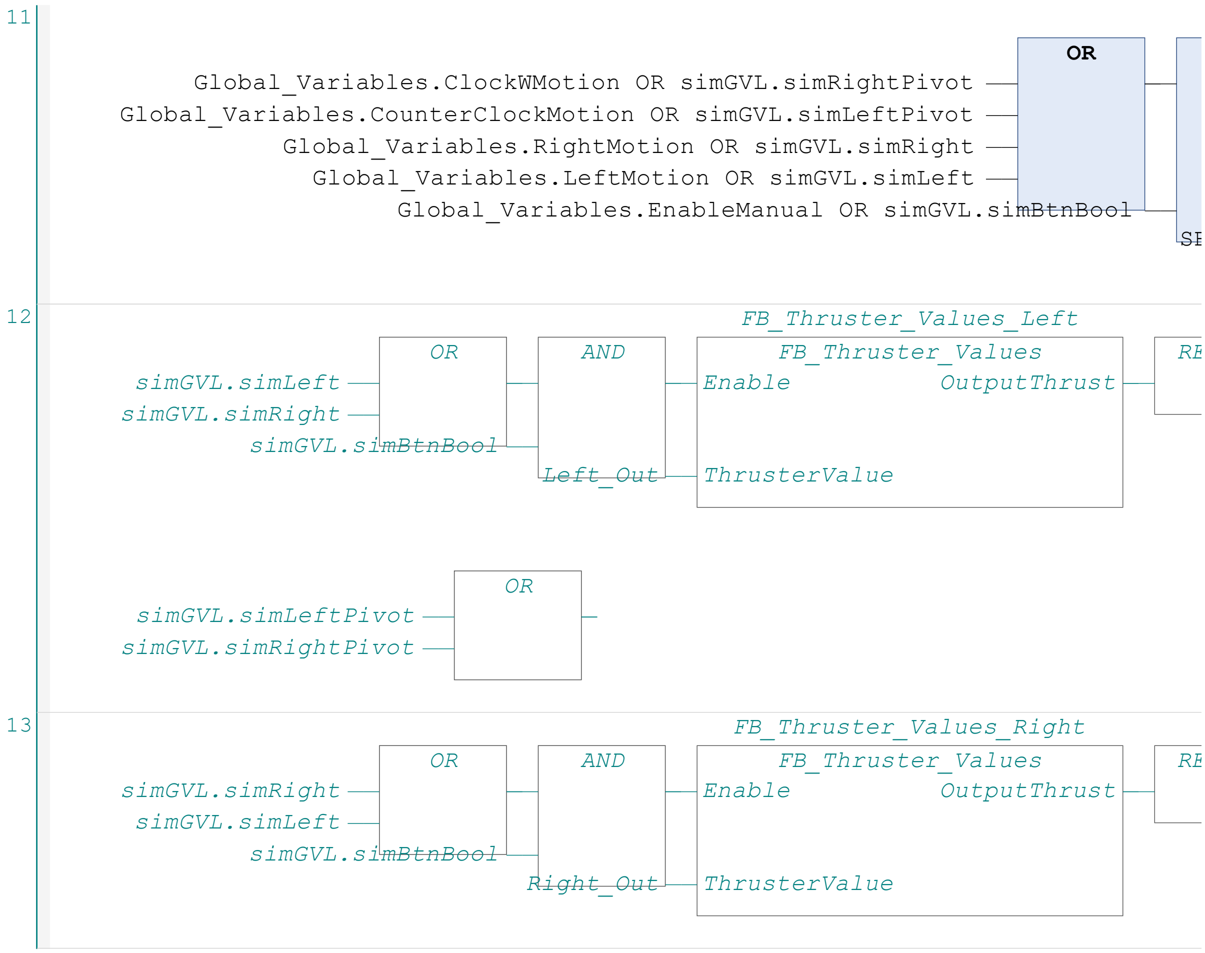

6
OR
AND
Global_Variables.FwdMotion OR simGVL.simForward
Global_Variables.BackMotion OR simGVL.simBackward
Global_Variables.EnableManual OR simGVL.simBtnBool
Forward_Out
7
OR
AND
Global_Variables.BackMotion OR simGVL.simBackward
Global_Variables.FwdMotion OR simGVL.simForward
Global_Variables.EnableManual OR simGVL.simBtnBool
Backward_Out
8
FB_Thruster_Values_LeftPivot
OR
AND
FB_Thruster_Values
simGVL.simLeftPivot
simGVL.simRightPivot
simGVL.simBtnBool
LeftPivot_Out
Enable
ThrusterValue
9
FB_Thruster_Values_RightPivo
OR
AND
FB_Thruster_Values
simGVL.simRightPivot
simGVL.simLeftPivot
simGVL.simBtnBool
RightPivot_Out
Enable
ThrusterValue
10
OR
Global_Variables.CounterClockMotion OR simGVL.simLeftPivot
Global_Variables.ClockWMotion OR simGVL.simRightPivot
Global_Variables.LeftMotion OR simGVL.simLeft
Global_Variables.RightMotion OR simGVL.simRight
Global_Variables.EnableManual OR simGVL.simBtnBool

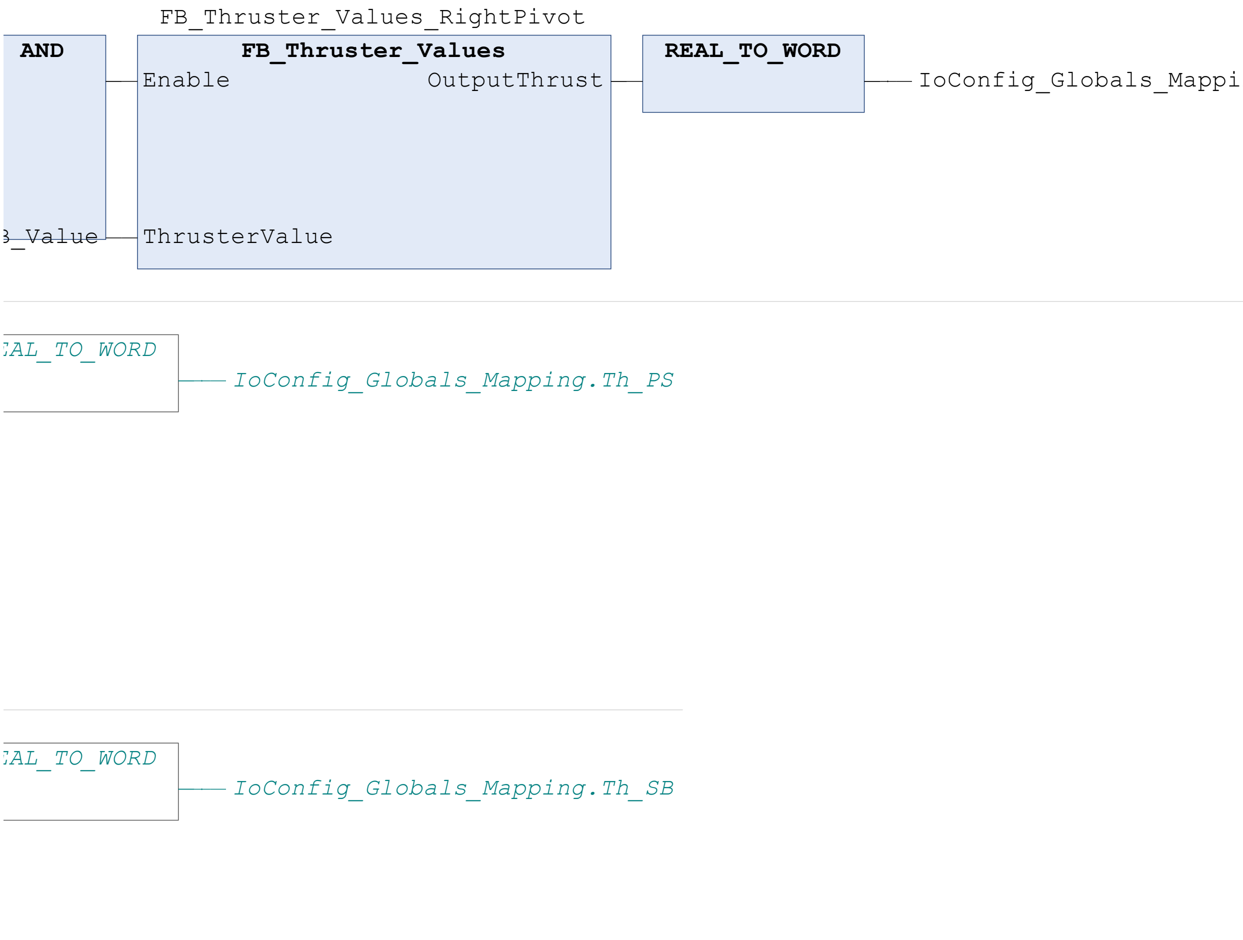

FB_Thruster_Values_Forward
FB_Thruster_Values
Enable
OutputThrust
ThrusterValue
REAL_TO_WORD
IoConfig_Globals_Mapping.Th_Forward
FB_Thruster_Values_Backward
FB_Thruster_Values
Enable
OutputThrust
ThrusterValue
REAL_TO_WORD
IoConfig_Globals_Mapping.Th_Backwa
REAL_TO_WORD
IoConfig_Globals_Mapping.Th_PS
REAL_TO_WORD
IoConfig_Globals_Mapping.Th_SB
FB_Thruster_Values_LeftPivot
AND
FB_Thruster_Values
Enable
OutputThrust
ThrusterValue
REAL_TO_WORD
IoConfig_Globals_Mapping

l

ırd

ɟ.Th_PS


11
OR
Global_Variables.ClockWMotion OR simGVL.simRightPivot
Global_Variables.CounterClockMotion OR simGVL.simLeftPivot
Global_Variables.RightMotion OR simGVL.simRight
Global_Variables.LeftMotion OR simGVL.simLeft
Global_Variables.EnableManual OR simGVL.simBtnBool
12
FB_Thruster_Values_Left
OR
AND
FB_Thruster_Values
simGVL.simLeft
simGVL.simRight
simGVL.simBtnBool
Enable
OutputThrust
Left_Out
ThrusterValue
OR
simGVL.simLeftPivot
simGVL.simRightPivot
13
FB_Thruster_Values_Right
OR
AND
FB_Thruster_Values
simGVL.simRight
simGVL.simLeft
simGVL.simBtnBool
Enable
OutputThrust
Right_Out
ThrusterValue


FB_Thruster_Values_RightPivot
AND
FB_Thruster_Values
Enable
OutputThrust
REAL_TO_WORD
IoConfig_Globals_Mappin
3_Value
ThrusterValue
AL_TO_WORD
IoConfig_Globals_Mapping.Th_PS
AL_TO_WORD
IoConfig_Globals_Mapping.Th_SB

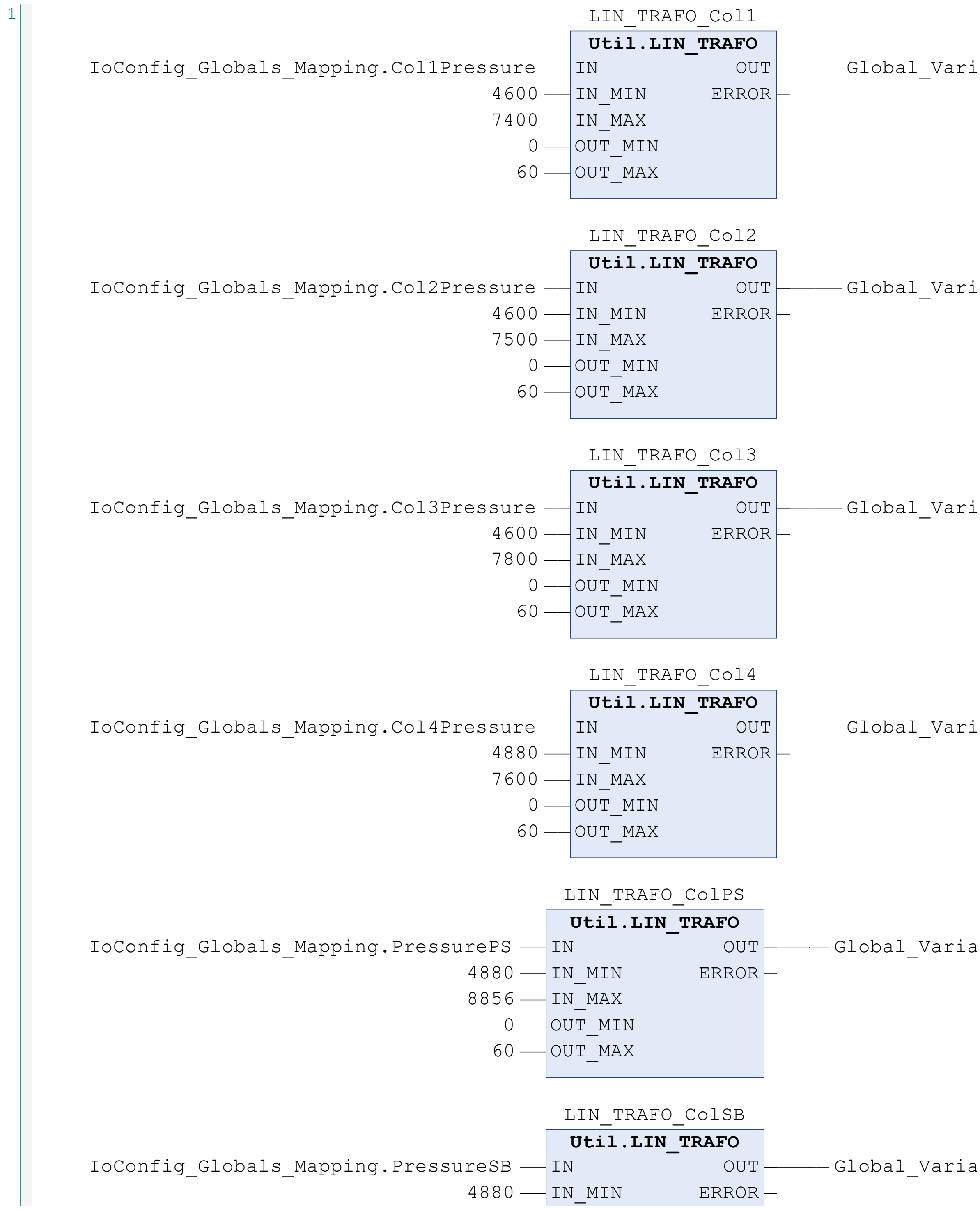
```
ıg.Th_SB
```

3.1.1.4 Folder: Scaling

3.1.1.4.1 POU: FB_PressureScaling

```
PROGRAM  FB_PressureScaling
VAR

    LIN_TRAFO_Col1 :  Util . LIN_TRAFO ;
    LIN_TRAFO_Col2 :  Util . LIN_TRAFO ;
    LIN_TRAFO_Col3 :  Util . LIN_TRAFO ;
    LIN_TRAFO_Col4 :  Util . LIN_TRAFO ;
    LIN_TRAFO_ColPS :  Util . LIN_TRAFO ;
    LIN_TRAFO_ColSB :  Util . LIN_TRAFO ;

    LIN_TRAFO_ColFilter :  Util . LIN_TRAFO ;

    FbLowPassFilter_0  :  WagoAppBuildingHVAC . FbLowPassFilterAI ;
    FbLowPassFilter_1  :  WagoAppBuildingHVAC . FbLowPassFilterAI ;

    FB_Filter_0  :  FB_Filter ;

    rOutput :  REAL ;
END_VAR

```

3.1.1.4.1 POU: FB_PressureScaling



ables.PressureCol1

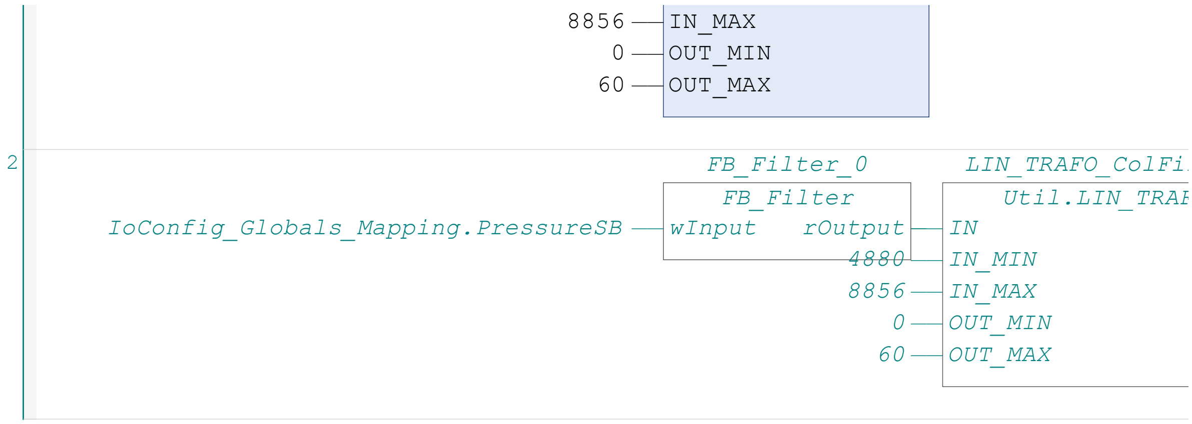
ables.PressureCol2

ables.PressureCol3

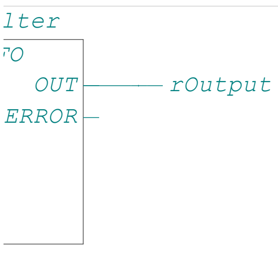
ables.PressureCol4

bles.PressurePS

bles.PressureSB


8856
IN_MAX
0
OUT_MIN
60
OUT_MAX
2
FB_Filter_0
FB_Filter
IoConfig_Globals_Mapping.PressureSB
wInput
rOutput
LIN_TRAFO_ColFi
Util.LIN_TRAF
IN
4880
IN_MIN
8856
IN_MAX
0
OUT_MIN
60
OUT_MAX

3.1.1.4.1 POU: FB_PressureScaling



3.1.1.5 POU: PLC_PRG

```
PROGRAM PLC_PRG
VAR
END_VAR
```

3.1.1.6 POU: POU_Test

```
PROGRAM POU_Test
VAR

END_VAR
```

3.1.1.7 POU: POU_Test_PID

```
PROGRAM POU_Test_PID
VAR
FB_CheckPitchRoll_1 : FB_CheckPitchRoll ;
    draft : REAL ;


END_VAR
```

3.1.1.8 Task Configuration: Task configuration

Max. number of tasks: 15
Max. number of cyclic tasks: 15
Max. number of freewheeling tasks: 15
Max. number of event tasks: 15
Max. number of external event tasks: 8
Max. number of status tasks: 15

System Events:

3.1.1.8.1 Task: PLC_Task

Priority: 15
Type: Cyclic
Interval: 50   Unit: ms
Watchdog: Inactive
POUs: FB_PressureScaling
POU_Platform_General
POU_ThrusterControl
POU_TestPumps

3.1.1.8.1.1 Program call: FB_PressureScaling

3.1.1.8.1.2 Program call: POU_Platform_General

3.1.1.8.1.3 Program call: POU_ThrusterControl

3.1.1.8.1.4 Program call: POU_TestPumps

3.1.1.8.2 Task: Task_Winch

Priority: 15
Type: Cyclic
Interval: t#50ms   Unit: ms
Watchdog: Inactive
POUs:

3.1.1.8.3 Task: TrendRecordingTask

Priority: 15
Type: Cyclic
Interval: 100 Unit: ms
Watchdog: Inactive
POUs: VisuTrendStorageAccess.GlobalInstances.g_TrendRecordingManager.CyclicCall

3.1.1.8.3.1 Program call: VisuTrendStorageAccess.GlobalInstances.g_Trend

3.1.1.9 Trace: Trace_ROV

Settings:

Record 'Trace_ROV':
Trigger variable:
Trigger edge: None
Post trigger (samples): 51
Trigger value:
Task: PLC_Task
Measure in every: 1-th cycle
Record condition:
Buffer size: 41
Comment:
POU for visualisation: False
Variables: POU_Platform_General.ROVDepth
POU_Platform_General.ROVOxygenWater
POU_Platform_General.ROVTemp
POU_Platform_General.ROVWaterTemp
POU_Platform_General.ROVHeading
POU_Platform_General.current

3.1.1.10 Trace: Trace_Stabilization

Settings:

Record 'Trace_Stabilization':
Trigger variable:
Trigger edge: None
Post trigger (samples): 51
Trigger value:
Task: PLC_Task
Measure in every: 1-th cycle
Record condition:

Buffer size: 41
Comment:
POU for visualisation: False
Variables: IoConfig_Globals_Mapping.Col1In
IoConfig_Globals_Mapping.Col1Out
IoConfig_Globals_Mapping.Col2In
IoConfig_Globals_Mapping.Col2Out
IoConfig_Globals_Mapping.Col3In
IoConfig_Globals_Mapping.Col3Out
IoConfig_Globals_Mapping.Col4In
IoConfig_Globals_Mapping.Col4Out
Global_Variables.Gyro_Pitch
Global_Variables.Gyro_Roll
POU_Stablilization.corrPitch
POU_Stablilization.corrRoll
Global_Variables.PressurePS
Global_Variables.PressureSB
Global_Variables.PressureCol1
Global_Variables.PressureCol2
Global_Variables.PressureCol3
Global_Variables.PressureCol4

3.1.1.11 Trace: Trace_USV

Settings:

Record 'Trace_USV':
Trigger variable:
Trigger edge: None
Post trigger (samples): 51
Trigger value:
Task: PLC_Task
Measure in every: 1-th cycle
Record condition:
Buffer size: 41
Comment:
POU for visualisation: False

Variables: IoConfig_Globals_Mapping.Th_Forward
IoConfig_Globals_Mapping.Th_Backward
IoConfig_Globals_Mapping.Th_PS
IoConfig_Globals_Mapping.Th_SB
Global_Variables.GPS_Heading
Global_Variables.GPS_Latitude
Global_Variables.GPS_Longitude
Global_Variables.GPS_Speed
Global_Variables.GUI_Latitude
Global_Variables.GUI_Longitude
Global_Variables.Gyro_Pitch
Global_Variables.Gyro_Roll
Global_Variables.Gyro_Yaw
IoConfig_Globals_Mapping.Current_Logg
POU_ThrusterControl.Setpoint_Angle_Autopilot
POU_ThrusterControl.distance
POU_ThrusterControl.SpeedValue_Autopilot
POU_ThrusterControl.Latitude_DP
POU_ThrusterControl.Longitude_DP

3.2 Device: Kbus

K-BUS Parameters

Parameters:

Name:	Type:	Value:	Default Value:	Unit:	Description:
k-bus cycle time	UINT	10	10		cycle time in [ms]
k-bus event control 1					cycle offset and cycle number
offset	USINT	0	0		offset in number of cycles
cycle number	USINT	2	2		cycle number (disable event with zero)
k-bus event control 2					cycle offset and cycle number
offset	USINT	1	1		offset in number of cycles
cycle number	USINT	4	4		cycle number (disable event with zero)
k-bus event control 3					cycle offset and cycle number
offset	USINT	3	3		offset in number of cycles
cycle number	USINT	8	8		cycle number (disable event with zero)
k-bus event control 4					cycle offset and cycle number
offset	USINT	7	7		offset in number of cycles
cycle number	USINT	16	16		cycle number (disable event with zero)
program start interlock	BOOL	False	TRUE		locks on configuration error

Information

Name: Kbus
Vendor: WAGO
Categories:
Type: 32778
ID: Wago 750-Series Local Bus Interface

Version: 1.4.0.2
Description: WAGO Kbus Interface

3.2.1 Device: Power_Supply_24_VDC

K-BUS Parameters

Parameters:

Name:	Type:	Value:	Default Value:	Unit:	Description:
K-BUS module slot index	BYTE	0	0		K-BUS slot index of the module (1 indexed)
Optional module	BOOL	0	0		Mark module as optional
Passive module	BOOL	1	1		Mark module as passive
Module compare value	STRING	'0000025AUUUUUUUU'	'0000025AUUUUUUUU'		Desired value
Module attitude	STRING	0750-0602	0750-0602		Module attitude

Information

Name: 0750-0602
Vendor: WAGO
Categories:
Type: 32776
ID: 07500602
Version: 0.0.0.9
Order number: 0750-0602
Description: Power Supply 24 VDC

3.2.2 Device: _8DI_24_VDC_3ms

K-BUS Parameters

Parameters:

Name:	Type:	Value:	Default Value:	Unit:	Description:
K-BUS module slot index	BYTE	1	0		K-BUS slot index of the module (1 indexed)
Optional module	BOOL	0	0		Mark module as optional
Module compare value	STRING	'UUUU8801UUUUUUUU'	'UUUU8801UUUUUUUU'		Desired value
Module attitude	STRING	0750-0430	0750-0430		Module attitude

Information

Name: 0750-0430
Vendor: WAGO
Categories:
Type: 32776
ID: 07500430

Version:	0.0.0.14
Order number:	0750-0430
Description:	8DI 24 VDC 3ms

3.2.3 Device: _8DO_24_VDC_0_5A

K-BUS Parameters

Parameters:

Name:	Type:	Value:	Default Value:	Unit:	Description:
K-BUS module slot index	BYTE	2	0		K-BUS slot index of the module (1 indexed)
Optional module	BOOL	0	0		Mark module as optional
Module compare value	STRING	'UUUU8802UUUUUUUU'	'UUUU8802UUUUUUUU'		Desired value
Module attitude	STRING	0750-0530	0750-0530		Module attitude

Information

Name:	0750-0530
Vendor:	WAGO
Categories:	
Type:	32776
ID:	07500530
Version:	0.0.0.15
Order number:	0750-0530
Description:	8DO 24 VDC 0.5A

3.2.4 Device: _8DO_24_VDC_0_5A_1

K-BUS Parameters

Parameters:

Name:	Type:	Value:	Default Value:	Unit:	Description:
K-BUS module slot index	BYTE	3	0		K-BUS slot index of the module (1 indexed)
Optional module	BOOL	0	0		Mark module as optional
Module compare value	STRING	'UUUU8802UUUUUUUU'	'UUUU8802UUUUUUUU'		Desired value
Module attitude	STRING	0750-0530	0750-0530		Module attitude

Information

Name:	0750-0530
Vendor:	WAGO
Categories:	
Type:	32776
ID:	07500530

Version: 0.0.0.15
Order number: 0750-0530
Description: 8DO 24 VDC 0.5A

3.2.5 Device: _4AI_10_VDC_SE

K-BUS Parameters

Parameters:

Name:	Type:	Value:	Default Value:	Unit:	Description:
K-BUS module slot index	BYTE	4	0		K-BUS slot index of the module (1 indexed)
Optional module	BOOL	0	0		Mark module as optional
Module compare value	STRING	'000001C9UUUUUUUU'	'000001C9UUUUUUUU'		Desired value
Module attitude	STRING	0750-0457	0750-0457		Module attitude

Information

Name: 0750-0457
Vendor: WAGO
Categories:
Type: 32776
ID: 07500457
Version: 0.0.0.9
Order number: 0750-0457
Description: 4AI ±10 VDC SE

3.2.6 Device: _4AI_10_VDC_SE_1

K-BUS Parameters

Parameters:

Name:	Type:	Value:	Default Value:	Unit:	Description:
K-BUS module slot index	BYTE	5	0		K-BUS slot index of the module (1 indexed)
Optional module	BOOL	0	0		Mark module as optional
Module compare value	STRING	'000001C9UUUUUUUU'	'000001C9UUUUUUUU'		Desired value
Module attitude	STRING	0750-0457	0750-0457		Module attitude

Information

Name: 0750-0457
Vendor: WAGO
Categories:
Type: 32776
ID: 07500457

Version: 0.0.0.9
Order number: 0750-0457
Description: 4AI ±10 VDC SE

3.2.7 Device: _4AO_0_10_VDC

K-BUS Parameters

Parameters:

Name:	Type:	Value:	Default Value:	Unit:	Description:
K-BUS module slot index	BYTE	6	0		K-BUS slot index of the module (1 indexed)
Optional module	BOOL	0	0		Mark module as optional
Module compare value	STRING	'0000022FUUUUUUUU'	'0000022FUUUUUUUU'		Desired value
Module attitude	STRING	0750-0559	0750-0559		Module attitude

Information

Name: 0750-0559
Vendor: WAGO
Categories:
Type: 32776
ID: 07500559
Version: 0.0.0.13
Order number: 0750-0559
Description: 4AO 0-10 VDC

3.2.8 Device: Stepper_Controller_RS422_24_VDC_20mA

K-BUS Parameters

Parameters:

Name:	Type:	Value:	Default Value:	Unit:	Description:
K-BUS module slot index	BYTE	7	0		K-BUS slot index of the module (1 indexed)
Optional module	BOOL	0	0		Mark module as optional
Module compare value	STRING	'0000029EUUUUUUUU'	'0000029EUUUUUUUU'		Desired value
Module attitude	STRING	0750-0670	0750-0670		Module attitude

Information

Name: 0750-0670
Vendor: WAGO
Categories:
Type: 32776
ID: 07500670

Version: 0.0.0.10
Order number: 0750-0670
Description: Stepper Controller RS422/24 VDC 20mA

3.2.9 Device: Stepper_Controller_RS422_24_VDC_20mA_1

K-BUS Parameters

Parameters:

Name:	Type:	Value:	Default Value:	Unit:	Description:
K-BUS module slot index	BYTE	8	0		K-BUS slot index of the module (1 indexed)
Optional module	BOOL	0	0		Mark module as optional
Module compare value	STRING	'0000029EUUUUUUUU'	'0000029EUUUUUUUU'		Desired value
Module attitude	STRING	0750-0670	0750-0670		Module attitude

Information

Name: 0750-0670
Vendor: WAGO
Categories:
Type: 32776
ID: 07500670
Version: 0.0.0.10
Order number: 0750-0670
Description: Stepper Controller RS422/24 VDC 20mA

3.2.10 Device: Inc_Encoder_24_VDC_SE_32bits

K-BUS Parameters

Parameters:

Name:	Type:	Value:	Default Value:	Unit:	Description:
K-BUS module slot index	BYTE	9	0		K-BUS slot index of the module (1 indexed)
Optional module	BOOL	0	0		Mark module as optional
Module compare value	STRING	'0000027D00000002'	'0000027D00000002'		Desired value
Module attitude	STRING	0750-0637/0000-0002	0750-0637/0000-0002		Module attitude

Information

Name: 0750-0637/0000-0002
Vendor: WAGO
Categories:
Type: 32776
ID: 0750063700000002

Version: 0.0.0.12
Order number: 0750-0637/0000-0002
Description: Inc. Encoder 24 VDC SE 32bits

3.3 Connector: MODBUS

MODBUS I/O Mapping

3.3.1 Device: MODBUS

Information

Name: MODBUS
Vendor: WAGO
Categories:
Type: 32777
ID: 1006 0001
Version: 1.1.1.15
Order number: n/a
Description: This device implements master and slave functionality for MODBUS.

3.3.1.1 Device: LocalDeviceModbus

MODBUS Slave Parameters

Parameters:

Name:	Type:	Value:	Default Value:	Unit:	Description:
OffsetMap	DWORD	16			
OffsetMap	DWORD	0			
OffsetMap	DWORD	64			
OffsetMap	DWORD	128			
OffsetMap	DWORD	192			
OffsetMap	DWORD	400			
OffsetMap	DWORD	256			
OffsetMap	DWORD	320			
OffsetMap	DWORD	400			
OffsetMap	DWORD	32			
OffsetMap	DWORD	448			
OffsetMap	DWORD	0			
OffsetMap	DWORD	1			
OffsetMap	DWORD	2			
OffsetMap	DWORD	3			
OffsetMap	DWORD	4			
OffsetMap	DWORD	5			
OffsetMap	DWORD	6			
OffsetMap	DWORD	7			
OffsetMap	DWORD	8			
OffsetMap	DWORD	9			
OffsetMap	DWORD	10			

OffsetMap	DWORD	12			
OffsetMap	DWORD	13			
OffsetMap	DWORD	14			
OffsetMap	DWORD	15			
OffsetMap	DWORD	288			
OffsetMap	DWORD	640			
OffsetMap	DWORD	672			
OffsetMap	DWORD	704			
OffsetMap	DWORD	768			
OffsetMap	DWORD	289			
OffsetMap	DWORD	64			
OffsetMap	DWORD	128			
OffsetMap	DWORD	224			
OffsetMap	DWORD	192			
OffsetMap	DWORD	384			
OffsetMap	DWORD	400			
OffsetMap	DWORD	416			
OffsetMap	DWORD	387			
OffsetMap	DWORD	388			
OffsetMap	DWORD	290			
OffsetMap	DWORD	11			
OffsetMap	DWORD	352			
OffsetMap	DWORD	448			
OffsetMap	DWORD	480			
OffsetMap	DWORD	512			
OffsetMap	DWORD	544			
Node ID	UINT	1	1		Used as slave address in RTU and as unit identifier in TCP/UDP
PLC stop behaviour	UDINT	0	2		
Fieldbus error behaviour	UDINT	0	2		
Response Delay	UINT	0	0		Used to delay responses in order to avoid high system load.
Watchdog settings					
Timeout	UINT	0	0	ms	Watchdog reset timeout.
Mode	UDINT	0	0		Selects modbus operation mode.
Explicit Trigger	UDINT	0	0		Enables explicit trigger on command WATCHDOG_START for simple mode.
Trigger on Status	UDINT	0	0		Enables trigger additionally on status register read for simple mode.
TCP connection reset	UDINT	0	0		Enables release of all established Modbus TCP connections when watchdog expires.

MODBUS Slave I/O Mapping

Input Parameters:

Mapping:	Channel:	Type:	Address:	Unit:	Description:
Application.Global_Variables.GPS_NumbersOfSatelites		INT			
Application.Global_Variables.GPS_Enabled		BOOL			
Application.Global_Variables.GPS_Latitude		LREAL			
Application.Global_Variables.GPS_Longitude		LREAL			

Application.Global_Variables.GPS_Heading		LREAL			
Application.Global_Variables.ConnectionCheck		BOOL			
Application.Global_Variables.Gyro_Pitch		REAL			
Application.Global_Variables.Gyro_Roll		REAL			
Application.Global_Variables.Gyro_Yaw		REAL			
Application.Global_Variables.GPS_Speed		REAL			
Application.Global_Variables.FwdMotion		BOOL			
Application.Global_Variables.BackMotion		BOOL			
Application.Global_Variables.RightMotion		BOOL			
Application.Global_Variables.LeftMotion		BOOL			
Application.Global_Variables.ClockWMotion		BOOL			
Application.Global_Variables.EnableLight		BOOL			
Application.Global_Variables.EnableFlute		BOOL			
Application.Global_Variables.PlatformEnable		BOOL			
Application.Global_Variables.EnableAuto		BOOL			
Application.Global_Variables.EnableManual		BOOL			
Application.Global_Variables.WinchUp		BOOL			
Application.Global_Variables.WinchDown		BOOL			
Application.Global_Variables.Winch_Lock_On		BOOL			
Application.Global_Variables.Winch_Lock_Off		BOOL			
Application.Global_Variables.Start_Pump		BOOL			
Application.Global_Variables.ThrusterSpeed		INT			
Application.Global_Variables.WinchSpeed		INT			
Application.Global_Variables.GUI_Latitude		REAL			
Application.Global_Variables.GUI_Longitude		REAL			
Application.Global_Variables.CounterClockMotion		BOOL			
Application.Global_Variables.GUI_ConCheck		BOOL			
Application.Global_Variables.Enable_DP		BOOL			
Application.Global_Variables.ROVTemp		REAL			
Application.Global_Variables.ROVDepth		REAL			
Application.Global_Variables.ROVWaterTemp		REAL			
Application.Global_Variables.ROVOxygenWater		REAL			
Application.Global_Variables.ROVHeading		REAL			

Output Parameters:

Mapping:	Channel:	Type:	Address:	Unit:	Description:
Application.Global_Variables.platLat		REAL			
Application.Global_Variables.PlatLong		REAL			
Application.Global_Variables.platYaw		REAL			
Application.Global_Variables.platRoll		REAL			
Application.Global_Variables.platHeading		REAL			
Application.Global_Variables.platSpeed		REAL			
Application.Global_Variables.platROVLocked		BOOL			
Application.Global_Variables.platROVUpperPos		BOOL			
Application.Global_Variables.platDP_ModeEnabled		BOOL			
Application.Global_Variables.platAutopilot_Enabled		BOOL			
Application.Global_Variables.platManual_ModeEnabled		BOOL			

Information

Name: MODBUS Slave
Vendor: WAGO

Categories:	
Type:	32777
ID:	1006 0001
Module ID:	Slave
Version:	1.1.1.15
Order number:	n/a
Description:	A MODBUS Slave responds as server to requests from a set of masters.

3.3.1.1.1 Device: TcpSettings

MODBUS Slave Parameters

Parameters:

Name:	Type:	Value:	Default Value:	Unit:	Description:
TCP Port	UINT	502	502		An TCP Server port to accept connections from a set of masters.
TCP connection watchdog	UINT	200	2000	10ms	The server resets a client connection if no valid request received within this time.
Type of Service settings					
Low Delay	UDINT	1	1		
High Throughput	UDINT	0	0		
High Reliability	UDINT	0	0		
TCP keepalive settings					
Enabled	UDINT	0	0		Activates TCP keepalive for Modbus connections.
Idle Time	UINT	7200	7200	Seconds	Time until keepalive probe starts.
Interval	UINT	1	1	Seconds	Interval between keepalive probes.
Count	UINT	10	10		Number of keepalive probes.

Information

Name:	Modbus TCP Slave
Vendor:	WAGO
Categories:	
Type:	32777
ID:	1006 0001
Module ID:	TCP Slave
Version:	1.1.1.15
Order number:	n/a
Description:	Modbus TCP Slave

3.4 Connector: Serial

N Platform source code

```
from VideoStream import videoStream

from modbusTcp import startModbusDataThreads
from modbusWriter import modbusClient
from SerialReadGyro import SerialReadGyro
import threading
import serial


def main():
    modbusIpadress = "192.168.0.112"
    t = threading.Thread(target=videoStream,name="VidThread",args=("192.168.0.101",12345))
    t.start()
    #startModbusDataThreads(modbusIpadress)
    modclient=modbusClient(modbusIpadress)
    modclient.start()







if __name__ == '__main__':main()
```

```
import gps
from threading import Thread
import queue

class gpsReader():

    def __init__(self):


        # Gps data variables
        self.GpsTime= "asa"
        self.Speed=0
        self.Latitude=0
        self.Heading=0
        self.Longitude=0
        self.NumberOfsatelites=0
        #queue for sharing data betweem thrread
        self.q = queue.LifoQueue()
        #need to implement

        # Listen on port 2947 (gpsd) of localhost
        self.session = gps.gps("localhost", "2947")
        self.session.stream(gps.WATCH_ENABLE | gps.WATCH_NEWSTYLE)

    def start(self):

        t = Thread(target=self.run, name="gpsReaderThread", args=())
        t.daemon = True
        t.start()



    # outQueue.put(0)
    def run(self):
        try:
            while True:


                self.NumberOfsatelites=self.session.satellites_used
                report = self.session.next()
                #print("Number of satelites :" + str(self.NumberOfsatelites))
```

```


                # Wait for a 'TPV' report from gpsd
                # To see all report data, uncomment the line below
                # print(report)
                if report['class'] == 'TPV':
                    if hasattr(report, 'time'):
                        #print("time :" + report.time)
                        self.GpsTime = report.time

                    if hasattr(report, 'track'):
                        #print("Heading :" + str(report.track))
                        self.Heading = report.track
                    if hasattr(report, 'lon'):
                        #print("Langitude :" + str(report.lon))
                        self.Longitude = report.lon

                    if hasattr(report, 'lat'):
                        #print("Latitude :" + str(report.lat))
                        self.Latitude = report.lat
                    if hasattr(report, 'speed'):
                        self.Speed = (report.speed * gps.MPS_TO_KPH)

                    gpsDataArray=[self.GpsTime,self.Heading,self.Longitude,
                                  self.Latitude,self.Speed,self.NumberOfsatelites]
                    self.q.put(gpsDataArray)

        except KeyError:
            pass
        except KeyboardInterrupt:
            quit()
        except StopIteration:
            session = None
            print("GPSD has terminated")

    def getGpsData(self):

        # Return the latest gpsdata:arrray[time,heading,
```

```
long,lat,speed]
        if not self.q.empty():
            newdata = self.q.get()

            while not self.q.empty():
                trashBin = self.q.get()

            return newdata
        else:
            # print("empty Queue in seiralRead")
            noData = [None] * 6
            return noData

    def queReady(self):
        if not self.q.empty():
            return True
        else:
            return False
```

```

import time
import socket
import cv2


def videoStream(ipadress,port):


    cam = cv2.VideoCapture(0)
    cam.set(cv2.CAP_PROP_FPS,30)
    UDP_IP = ipadress
    UDP_PORT = port
    sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)



    # warmup camera
    time.sleep(0.1)
    while True:

        ret_val, img = cam.read(0)

        img = cv2.flip(img, 1)
        img = cv2.resize(img, (320, 240))

        x = [int(cv2.IMWRITE_JPEG_QUALITY), 80]
        _, compressed = cv2.imencode(".jpg", img, x)
        sock.sendto(compressed, (UDP_IP, UDP_PORT))


    rawCapture.truncate(0)




#if __name__ == "__main__": main()




```

```python
import serial
import time
from pymodbus.client.sync import ModbusTcpClient
from pymodbus.constants import Endian
from pymodbus.exceptions import ConnectionException
from pymodbus.payload import BinaryPayloadBuilder
from threading import Thread
from GpsReader import gpsReader
from SerialReadGyro import SerialReadGyro


#The modbusclient reads the value from the GPS and the Gyro, and sends the data to the Wago PLC.
class modbusClient():

    def __init__(self,iAdress):
        self.client = ModbusTcpClient(iAdress)
        self.builder = BinaryPayloadBuilder(byteorder=Endian.Big,
                                            wordorder=Endian.Little)
        self.gpsreader = gpsReader()
        self.gyroreader = SerialReadGyro(serialPort=serial.Serial('/dev/ttyACM1', 19200, timeout=5))

        self.gpsTime0 = "asa"
        self.timeOutCheckTime = 2
        self.gpsTimeOutTime = time.time()
        self.GpsEnabled = True
        self.connCheckTimer = time.time()
        self.connCheckVar=True

        ##GPSDATA#####
        self.GpsTime = "asa"
        self.GpsHeading = 0
        self.GpsLong = 0
        self.GpsLat = 0
        self.GpsSpeed = 0
        self.GpsNumberOfSat = 0

        ##GYRODATA##
        self.GyroHeading = 0
        self.GyroPitch = 0
        self.GyroRoll = 0

```

```
        ##cameraServoController###
        self.CameraRovPos=False
        self.camCoilNr=0

    def start(self):
        self.gpsreader.start()
        self.gyroreader.start()
        t = Thread(target=self.run, name="ModbusThread", args=())
        t.daemon = True
        t.start()

    # runns when called thread.start. reads data from gps and gyro
    def run(self):

        while True:

            #checking if there is connection to the GPS by looking for change in the clock, Checks status each 2 sec
            if self.gpsTimeOutTime + self.timeOutCheckTime < time.time():
                if self.GpsTime == self.gpsTime0 or self.GpsTime == None:
                    self.GpsEnabled = False
                else:
                    self.GpsEnabled = True
                    self.gpsTime0 = self.GpsTime
                    self.gpsTimeOutTime = time.time()
                    # Gps enable flag on coil 32912
                self.client.write_coil(32768, [self.GpsEnabled] * 8, unit=1)
                print("GPS ENABLED: "+str(self.GpsEnabled))

            # Switching a bit for so the plc can notice connection loss. switching 1 time a sec.
            if self.connCheckTimer + 1 < time.time():
                if self.connCheckVar:
                    self.connCheckVar = False
                else:
                    self.connCheckVar = True

                self.client.write_coils(33168, [self.connCheckVar] * 8, unit=1)
```

```
                self.connCheckTimer = time.time()





            # if the gpsReader thread is ready with new values then send to Modbus
            if self.gpsreader.queReady():

                self.updateLocalGPsData(self.gpsreader.getGpsData())

                # write modbus Number of satelites on register 32008 on wago plc
                self.client.write_registers(32001, self.build16BitMessage(self.GpsNumberOfSat), unit=1)

                # write modbus Speed on register 32004 on wago plc

                self.client.write_registers(32002, self.build32BitMessage(self.GpsSpeed), unit=1)

                # write modbus Latitude on register 32012 on wago plc

                self.client.write_registers(32004, self.build64BitMessage(self.GpsLat), unit=1)

                # write modbus Heading on register 320012 on wago plc

                self.client.write_registers(32012, self.build64BitMessage(self.GpsHeading), unit=1)

                # write modbus Longitude on register 32008 on wago plc

                self.client.write_registers(32008, self.build64BitMessage(self.GpsLong), unit=1)

                #Change CameraPos
                self.client.read_coils(self.camCoilNr, 1, unit=1)
```

```python

            #if Gyro thread is ready with new data then send gyro data to plc
            if self.gyroreader.queReady():
                try:
                    self.updateLocalGyroData(self.gyroreader.getSerialData())

                    # write modbus GyroHeading on register 32004 on wago plc

                    self.client.write_registers(32025, self.build32BitMessage(self.GyroHeading), unit=1)

                    # write modbus GyroPitch on register 32004 on wago plc

                    self.client.write_registers(32016, self.build32BitMessage(self.GyroPitch), unit=1)

                    # write modbus GyroRoll on register 32004 on wago plc

                    self.client.write_registers(32020, self.build32BitMessage(self.GyroRoll), unit=1)
                except ConnectionException as e:
                    print ("exception write register gyrodata")
                    continue





    def updateLocalGPsData(self,gpsData):
        self.GpsTime = gpsData[0]
        self.GpsHeading = gpsData[1]
        self.GpsLong = gpsData[2]
        self.GpsLat = gpsData[3]
        self.GpsSpeed = gpsData[4]
        self.GpsNumberOfSat = gpsData[5]

    def updateLocalGyroData(self,gyroData):
        self.GyroHeading = gyroData[0]
        print("gyroHeading: "+ str(self.GyroHeading))
```

```
        self.GyroPitch = gyroData[2]
        print("gyroPitch: " + str(self.GyroPitch))
        self.GyroRoll = gyroData[1]
        print("gyroRoll: " + str(self.GyroRoll))


    def build16BitMessage(self,message):
        #encode 16 bit integer message
        self.builder.reset()
        self.builder.add_16bit_int(message)
        encoded16 = self.builder.to_registers()
        return encoded16


    def build32BitMessage(self, message):
        # encode 32 bit float message
        self.builder.reset()
        self.builder.add_32bit_float(message)
        encoded32 = self.builder.to_registers()
        return encoded32

    def build64BitMessage(self, message):
        # encode 64 bit float message
        self.builder.reset()
        self.builder.add_64bit_float(message)
        encoded64 = self.builder.to_registers()
        return encoded64
```

```python
import queue
import serial
from threading import Thread
import time


# Reads serialcom from the Gyroscope on the port "serialPort"
class SerialReadGyro():
    def __init__(self, serialPort):

        self.q = queue.LifoQueue()

        self.serialPort = serialPort

    def start(self):

        t = Thread(target=self.run, name="GyroThread", args=())
        t.daemon = True
        t.start()

    # runns when called thread.start. reads data from arduino and setts the input to the arranged value

    def run(self):
        try:
            while (True):

                # print("starting to read")
                data = str(self.serialPort.readline())


                ##removing b' flag at the beginning of the string
                cleanedString = data.split("b'")
                #print(cleanedString)

                # splitting the string where there is ;
                splitData = cleanedString[1].split(";")
                cleanLastPart = splitData[2].split("\\")

                # list type=string [0]=gyro heading(yaw) [1] = roll [2]= pitch
                splitData[2] = cleanLastPart[0]
```

```

                if splitData.__len__() == 3:
                    self.q.put(self.convertStringToFloat(splitData))




        except serial.SerialException as e:
            print("SerialPortException i SerialRead")

    def convertStringToFloat(self, l):
        # create a list with the size of 3 (0...2)& converts the string to float values
        list = [None] * 3
        list[0] = float(l[0])
        list[1] = float(l[1])
        list[2] = float(l[2])
        return list

    def getSerialData(self):

        # Return the latest serialData
        if not self.q.empty():
            newdata = self.q.get()

            # while not self.q.empty():
            # trashBin = self.q.get()

            return newdata
        else:
            # print("empty Queue in seiralRead")
            noData = [None] * 3
            return noData

    def queReady(self):
        if not self.q.empty():
            return True
        else:
            return False

```

O ROV source code

```python


from DataHandlerNew import datahandler
from UDPvideoStream import videoStream
import threading

#the main class of the ROV. Starts the datahandler and the viedeostreamThread

def main():
    # Use the Ip adress of the machine running the GUI here
    guiIpadress= '192.168.0.103'
    print("Starting")
    #creates a thread of the UDP videostream
    #t = threading.Thread(target=videoStream, name="thread1", args=(guiIpadress, 12342))
    #t.start()
    #starts the datahandler
    comm = datahandler(readGUIPort=9876,sendGUIDataPort=8765,iAdress=(guiIpadress))

if __name__ == '__main__':
    main()

```

```
import queue
import serial
from threading import Thread
import time

#Reads serialcom from the Arduino on the port "serialPort"
class SerialRead ():
    def __init__(self,serialPort,semaPhore):

        self.q = queue.LifoQueue()
        self.semaphore = semaPhore
        self.serialPort = serialPort
        self.dataArduino = bytearray(23)

    def start(self):

       t = Thread(target=self.run,name="thread3", args=())
       t.daemon = True
       t.start()


 #runns when called thread.start. reads data from arduino and setts the input to the arranged value

    def run(self):
        try :
            while (True):
                self.semaphore.acquire()

                data = self.serialPort.read(23)
                self.semaphore.release()
                if data.__len__()>0 :


                  arrangedData = self.checkDataArrangement(data)

                  self.q.put(arrangedData)

                  time.sleep(0.5)


        except serial.SerialException as e :
           print("SerialPortException i SerialRead")

```

```

    def getSerialData(self):

        # Return the latest serialData
        if not self.q.empty():
            newCommand = self.q.get()

           # while not self.q.empty():
               # trashBin = self.q.get()

            return newCommand
        else:
            #print("empty Queue in seiralRead")
            nodata=bytearray(23)
            return nodata

    def queReady(self):
      if not self.q.empty():
         return True
      else:
         return False




 #checs the position of dthe data read from arduino and sets it on the right place
 #@param data
 #@return arrangedData

    def checkDataArrangement(self, data):
        realData = bytearray(23)
        for x in range(0,22):


            #print(x)
            #print(str(data[x])+"Xval= "+ str(x))
            if data[x] == (128):
                 if x == 22:
                    realData[0] = data[x]
                    realData[1] = data[x - 22]
                    realData[2] = data[x - 21]
                    realData[3] = data[x - 20]
                    realData[4] = data[x - 19]
                    realData[5] = data[x - 18]
```

```
                    realData[6] = data[x - 17]
                    realData[7] = data[x - 16]
                    realData[8] = data[x - 15]
                    realData[9] = data[x - 14]
                    realData[10] = data[x - 13]
                    realData[11] = data[x - 12]
                    realData[12] = data[x - 11]
                    realData[13] = data[x - 10]
                    realData[14] = data[x - 9]
                    realData[15] = data[x - 8]
                    realData[16] = data[x - 7]
                    realData[17] = data[x - 6]
                    realData[18] = data[x - 5]
                    realData[19] = data[x - 4]
                    realData[20] = data[x - 3]
                    realData[21] = data[x - 2]
                    realData[22] = data[x - 1]

                elif (x == 21):
                    realData[0] = data[x]
                    realData[1] = data[x +1]
                    realData[2] = data[x - 21]
                    realData[3] = data[x - 20]
                    realData[4] = data[x - 19]
                    realData[5] = data[x - 18]
                    realData[6] = data[x - 17]
                    realData[7] = data[x - 16]
                    realData[8] = data[x - 15]
                    realData[9] = data[x - 14]
                    realData[10] = data[x - 13]
                    realData[11] = data[x - 12]
                    realData[12] = data[x - 11]
                    realData[13] = data[x - 10]
                    realData[14] = data[x - 9]
                    realData[15] = data[x - 8]
                    realData[16] = data[x - 7]
                    realData[17] = data[x - 6]
                    realData[18] = data[x - 5]
                    realData[19] = data[x - 4]
                    realData[20] = data[x - 3]
                    realData[21] = data[x - 2]
                    realData[22] = data[x - 1]

                elif (x == 20):
                    realData[0] = data[x]
```

```
                    realData[1] = data[x +1]
                    realData[2] = data[x +2]
                    realData[3] = data[x - 20]
                    realData[4] = data[x - 19]
                    realData[5] = data[x - 18]
                    realData[6] = data[x - 17]
                    realData[7] = data[x - 16]
                    realData[8] = data[x - 15]
                    realData[9] = data[x - 14]
                    realData[10] = data[x - 13]
                    realData[11] = data[x - 12]
                    realData[12] = data[x - 11]
                    realData[13] = data[x - 10]
                    realData[14] = data[x - 9]
                    realData[15] = data[x - 8]
                    realData[16] = data[x - 7]
                    realData[17] = data[x - 6]
                    realData[18] = data[x - 5]
                    realData[19] = data[x - 4]
                    realData[20] = data[x - 3]
                    realData[21] = data[x - 2]
                    realData[22] = data[x - 1]

                elif (x == 19):
                    realData[0] = data[x]
                    realData[1] = data[x +1]
                    realData[2] = data[x +2]
                    realData[3] = data[x +3]
                    realData[4] = data[x - 19]
                    realData[5] = data[x - 18]
                    realData[6] = data[x - 17]
                    realData[7] = data[x - 16]
                    realData[8] = data[x - 15]
                    realData[9] = data[x - 14]
                    realData[10] = data[x - 13]
                    realData[11] = data[x - 12]
                    realData[12] = data[x - 11]
                    realData[13] = data[x - 10]
                    realData[14] = data[x - 9]
                    realData[15] = data[x - 8]
                    realData[16] = data[x - 7]
                    realData[17] = data[x - 6]
                    realData[18] = data[x - 5]
                    realData[19] = data[x - 4]
                    realData[20] = data[x - 3]
```

```
                    realData[21] = data[x - 2]
                    realData[22] = data[x - 1]

                elif (x == 18):
                    realData[0] = data[x]
                    realData[1] = data[x +1]
                    realData[2] = data[x +2]
                    realData[3] = data[x +3]
                    realData[4] = data[x +4]
                    realData[5] = data[x - 18]
                    realData[6] = data[x - 17]
                    realData[7] = data[x - 16]
                    realData[8] = data[x - 15]
                    realData[9] = data[x - 14]
                    realData[10] = data[x - 13]
                    realData[11] = data[x - 12]
                    realData[12] = data[x - 11]
                    realData[13] = data[x - 10]
                    realData[14] = data[x - 9]
                    realData[15] = data[x - 8]
                    realData[16] = data[x - 7]
                    realData[17] = data[x - 6]
                    realData[18] = data[x - 5]
                    realData[19] = data[x - 4]
                    realData[20] = data[x - 3]
                    realData[21] = data[x - 2]
                    realData[22] = data[x - 1]

                elif (x == 17):
                    realData[0] = data[x]
                    realData[1] = data[x +1]
                    realData[2] = data[x +2]
                    realData[3] = data[x +3]
                    realData[4] = data[x +4]
                    realData[5] = data[x +5]
                    realData[6] = data[x - 17]
                    realData[7] = data[x - 16]
                    realData[8] = data[x - 15]
                    realData[9] = data[x - 14]
                    realData[10] = data[x - 13]
                    realData[11] = data[x - 12]
                    realData[12] = data[x - 11]
                    realData[13] = data[x - 10]
                    realData[14] = data[x - 9]
                    realData[15] = data[x - 8]
```

```
                    realData[16] = data[x - 7]
                    realData[17] = data[x - 6]
                    realData[18] = data[x - 5]
                    realData[19] = data[x - 4]
                    realData[20] = data[x - 3]
                    realData[21] = data[x - 2]
                    realData[22] = data[x - 1]

                elif (x == 16):
                    realData[0] = data[x]
                    realData[1] = data[x +1]
                    realData[2] = data[x +2]
                    realData[3] = data[x +3]
                    realData[4] = data[x +4]
                    realData[5] = data[x +5]
                    realData[6] = data[x + 6]
                    realData[7] = data[x - 16]
                    realData[8] = data[x - 15]
                    realData[9] = data[x - 14]
                    realData[10] = data[x - 13]
                    realData[11] = data[x - 12]
                    realData[12] = data[x - 11]
                    realData[13] = data[x - 10]
                    realData[14] = data[x - 9]
                    realData[15] = data[x - 8]
                    realData[16] = data[x - 7]
                    realData[17] = data[x - 6]
                    realData[18] = data[x - 5]
                    realData[19] = data[x - 4]
                    realData[20] = data[x - 3]
                    realData[21] = data[x - 2]
                    realData[22] = data[x - 1]

                elif (x == 15):
                    realData[0] = data[x]
                    realData[1] = data[x +1]
                    realData[2] = data[x +2]
                    realData[3] = data[x +3]
                    realData[4] = data[x +4]
                    realData[5] = data[x +5]
                    realData[6] = data[x + 6]
                    realData[7] = data[x +7]
                    realData[8] = data[x - 15]
                    realData[9] = data[x - 14]
                    realData[10] = data[x - 13]
```

```
                realData[11] = data[x - 12]
                realData[12] = data[x - 11]
                realData[13] = data[x - 10]
                realData[14] = data[x - 9]
                realData[15] = data[x - 8]
                realData[16] = data[x - 7]
                realData[17] = data[x - 6]
                realData[18] = data[x - 5]
                realData[19] = data[x - 4]
                realData[20] = data[x - 3]
                realData[21] = data[x - 2]
                realData[22] = data[x - 1]

            elif (x == 14):
                realData[0] = data[x]
                realData[1] = data[x +1]
                realData[2] = data[x +2]
                realData[3] = data[x +3]
                realData[4] = data[x +4]
                realData[5] = data[x +5]
                realData[6] = data[x + 6]
                realData[7] = data[x +7]
                realData[8] = data[x +8]
                realData[9] = data[x - 14]
                realData[10] = data[x - 13]
                realData[11] = data[x - 12]
                realData[12] = data[x - 11]
                realData[13] = data[x - 10]
                realData[14] = data[x - 9]
                realData[15] = data[x - 8]
                realData[16] = data[x - 7]
                realData[17] = data[x - 6]
                realData[18] = data[x - 5]
                realData[19] = data[x - 4]
                realData[20] = data[x - 3]
                realData[21] = data[x - 2]
                realData[22] = data[x - 1]

            elif (x == 13):
                realData[0] = data[x]
                realData[1] = data[x +1]
                realData[2] = data[x +2]
                realData[3] = data[x +3]
                realData[4] = data[x +4]
                realData[5] = data[x +5]
```

```
                    realData[6] = data[x + 6]
                    realData[7] = data[x +7]
                    realData[8] = data[x +8]
                    realData[9] = data[x +9]
                    realData[10] = data[x - 13]
                    realData[11] = data[x - 12]
                    realData[12] = data[x - 11]
                    realData[13] = data[x - 10]
                    realData[14] = data[x - 9]
                    realData[15] = data[x - 8]
                    realData[16] = data[x - 7]
                    realData[17] = data[x - 6]
                    realData[18] = data[x - 5]
                    realData[19] = data[x - 4]
                    realData[20] = data[x - 3]
                    realData[21] = data[x - 2]
                    realData[22] = data[x - 1]

                elif (x == 12):
                    realData[0] = data[x]
                    realData[1] = data[x +1]
                    realData[2] = data[x +2]
                    realData[3] = data[x +3]
                    realData[4] = data[x +4]
                    realData[5] = data[x +5]
                    realData[6] = data[x + 6]
                    realData[7] = data[x +7]
                    realData[8] = data[x +8]
                    realData[9] = data[x +9]
                    realData[10] = data[x +10]
                    realData[11] = data[x - 12]
                    realData[12] = data[x - 11]
                    realData[13] = data[x - 10]
                    realData[14] = data[x - 9]
                    realData[15] = data[x - 8]
                    realData[16] = data[x - 7]
                    realData[17] = data[x - 6]
                    realData[18] = data[x - 5]
                    realData[19] = data[x - 4]
                    realData[20] = data[x - 3]
                    realData[21] = data[x - 2]
                    realData[22] = data[x - 1]

                elif (x == 11):
                    realData[0] = data[x]
```

```
                    realData[1] = data[x +1]
                    realData[2] = data[x +2]
                    realData[3] = data[x +3]
                    realData[4] = data[x +4]
                    realData[5] = data[x +5]
                    realData[6] = data[x + 6]
                    realData[7] = data[x +7]
                    realData[8] = data[x +8]
                    realData[9] = data[x +9]
                    realData[10] = data[x +10]
                    realData[11] = data[x + 11]
                    realData[12] = data[x - 11]
                    realData[13] = data[x - 10]
                    realData[14] = data[x - 9]
                    realData[15] = data[x - 8]
                    realData[16] = data[x - 7]
                    realData[17] = data[x - 6]
                    realData[18] = data[x - 5]
                    realData[19] = data[x - 4]
                    realData[20] = data[x - 3]
                    realData[21] = data[x - 2]
                    realData[22] = data[x - 1]

                elif (x == 10):
                    realData[0] = data[x]
                    realData[1] = data[x +1]
                    realData[2] = data[x +2]
                    realData[3] = data[x +3]
                    realData[4] = data[x +4]
                    realData[5] = data[x +5]
                    realData[6] = data[x + 6]
                    realData[7] = data[x +7]
                    realData[8] = data[x +8]
                    realData[9] = data[x +9]
                    realData[10] = data[x +10]
                    realData[11] = data[x + 11]
                    realData[12] = data[x +12]
                    realData[13] = data[x - 10]
                    realData[14] = data[x - 9]
                    realData[15] = data[x - 8]
                    realData[16] = data[x - 7]
                    realData[17] = data[x - 6]
                    realData[18] = data[x - 5]
                    realData[19] = data[x - 4]
                    realData[20] = data[x - 3]
```

```
                        realData[21] = data[x - 2]
                        realData[22] = data[x - 1]
                    elif (x == 9):
                        realData[0] = data[x]
                        realData[1] = data[x +1]
                        realData[2] = data[x +2]
                        realData[3] = data[x +3]
                        realData[4] = data[x +4]
                        realData[5] = data[x +5]
                        realData[6] = data[x + 6]
                        realData[7] = data[x +7]
                        realData[8] = data[x +8]
                        realData[9] = data[x +9]
                        realData[10] = data[x +10]
                        realData[11] = data[x + 11]
                        realData[12] = data[x +12]
                        realData[13] = data[x +13]
                        realData[14] = data[x - 9]
                        realData[15] = data[x - 8]
                        realData[16] = data[x - 7]
                        realData[17] = data[x - 6]
                        realData[18] = data[x - 5]
                        realData[19] = data[x - 4]
                        realData[20] = data[x - 3]
                        realData[21] = data[x - 2]
                        realData[22] = data[x - 1]
                    elif (x == 8):
                        realData[0] = data[x]
                        realData[1] = data[x +1]
                        realData[2] = data[x +2]
                        realData[3] = data[x +3]
                        realData[4] = data[x +4]
                        realData[5] = data[x +5]
                        realData[6] = data[x + 6]
                        realData[7] = data[x +7]
                        realData[8] = data[x +8]
                        realData[9] = data[x +9]
                        realData[10] = data[x +10]
                        realData[11] = data[x + 11]
                        realData[12] = data[x +12]
                        realData[13] = data[x +13]
                        realData[14] = data[x +14]
                        realData[15] = data[x - 8]
                        realData[16] = data[x - 7]
                        realData[17] = data[x - 6]
```

```
                    realData[18] = data[x - 5]
                    realData[19] = data[x - 4]
                    realData[20] = data[x - 3]
                    realData[21] = data[x - 2]
                    realData[22] = data[x - 1]
                elif (x == 7):
                    realData[0] = data[x]
                    realData[1] = data[x +1]
                    realData[2] = data[x +2]
                    realData[3] = data[x +3]
                    realData[4] = data[x +4]
                    realData[5] = data[x +5]
                    realData[6] = data[x + 6]
                    realData[7] = data[x +7]
                    realData[8] = data[x +8]
                    realData[9] = data[x +9]
                    realData[10] = data[x +10]
                    realData[11] = data[x + 11]
                    realData[12] = data[x +12]
                    realData[13] = data[x +13]
                    realData[14] = data[x +14]
                    realData[15] = data[x +15]
                    realData[16] = data[x - 7]
                    realData[17] = data[x - 6]
                    realData[18] = data[x - 5]
                    realData[19] = data[x - 4]
                    realData[20] = data[x - 3]
                    realData[21] = data[x - 2]
                    realData[22] = data[x - 1]

                elif (x == 6):
                    realData[0] = data[x]
                    realData[1] = data[x +1]
                    realData[2] = data[x +2]
                    realData[3] = data[x +3]
                    realData[4] = data[x +4]
                    realData[5] = data[x +5]
                    realData[6] = data[x + 6]
                    realData[7] = data[x +7]
                    realData[8] = data[x +8]
                    realData[9] = data[x +9]
                    realData[10] = data[x +10]
                    realData[11] = data[x + 11]
                    realData[12] = data[x +12]
                    realData[13] = data[x +13]
```

```
                    realData[14] = data[x +14]
                    realData[15] = data[x +15]
                    realData[16] = data[x +16]
                    realData[17] = data[x - 6]
                    realData[18] = data[x - 5]
                    realData[19] = data[x - 4]
                    realData[20] = data[x - 3]
                    realData[21] = data[x - 2]
                    realData[22] = data[x - 1]

                elif (x == 5):
                    realData[0] = data[x]
                    realData[1] = data[x +1]
                    realData[2] = data[x +2]
                    realData[3] = data[x +3]
                    realData[4] = data[x +4]
                    realData[5] = data[x +5]
                    realData[6] = data[x + 6]
                    realData[7] = data[x +7]
                    realData[8] = data[x +8]
                    realData[9] = data[x +9]
                    realData[10] = data[x +10]
                    realData[11] = data[x + 11]
                    realData[12] = data[x +12]
                    realData[13] = data[x +13]
                    realData[14] = data[x +14]
                    realData[15] = data[x +15]
                    realData[16] = data[x +16]
                    realData[17] = data[x +17]
                    realData[18] = data[x - 5]
                    realData[19] = data[x - 4]
                    realData[20] = data[x - 3]
                    realData[21] = data[x - 2]
                    realData[22] = data[x - 1]

                elif (x == 4):
                    realData[0] = data[x]
                    realData[1] = data[x +1]
                    realData[2] = data[x +2]
                    realData[3] = data[x +3]
                    realData[4] = data[x +4]
                    realData[5] = data[x +5]
                    realData[6] = data[x + 6]
                    realData[7] = data[x +7]
                    realData[8] = data[x +8]
```

```
                        realData[9] = data[x +9]
                        realData[10] = data[x +10]
                        realData[11] = data[x + 11]
                        realData[12] = data[x +12]
                        realData[13] = data[x +13]
                        realData[14] = data[x +14]
                        realData[15] = data[x +15]
                        realData[16] = data[x +16]
                        realData[17] = data[x +17]
                        realData[18] = data[x +18]
                        realData[19] = data[x - 4]
                        realData[20] = data[x - 3]
                        realData[21] = data[x - 2]
                        realData[22] = data[x - 1]

                    elif (x == 3):
                        realData[0] = data[x]
                        realData[1] = data[x +1]
                        realData[2] = data[x +2]
                        realData[3] = data[x +3]
                        realData[4] = data[x +4]
                        realData[5] = data[x +5]
                        realData[6] = data[x + 6]
                        realData[7] = data[x +7]
                        realData[8] = data[x +8]
                        realData[9] = data[x +9]
                        realData[10] = data[x +10]
                        realData[11] = data[x + 11]
                        realData[12] = data[x +12]
                        realData[13] = data[x +13]
                        realData[14] = data[x +14]
                        realData[15] = data[x +15]
                        realData[16] = data[x +16]
                        realData[17] = data[x +17]
                        realData[18] = data[x +18]
                        realData[19] = data[x +19]
                        realData[20] = data[x - 3]
                        realData[21] = data[x - 2]
                        realData[22] = data[x - 1]

                    elif (x == 2):
                        realData[0] = data[x]
                        realData[1] = data[x +1]
                        realData[2] = data[x +2]
                        realData[3] = data[x +3]
```

```
                    realData[4] = data[x +4]
                    realData[5] = data[x +5]
                    realData[6] = data[x + 6]
                    realData[7] = data[x +7]
                    realData[8] = data[x +8]
                    realData[9] = data[x +9]
                    realData[10] = data[x +10]
                    realData[11] = data[x + 11]
                    realData[12] = data[x +12]
                    realData[13] = data[x +13]
                    realData[14] = data[x +14]
                    realData[15] = data[x +15]
                    realData[16] = data[x +16]
                    realData[17] = data[x +17]
                    realData[18] = data[x +18]
                    realData[19] = data[x +19]
                    realData[20] = data[x +20]
                    realData[21] = data[x - 2]
                    realData[22] = data[x - 1]

                elif (x == 1):
                    realData[0] = data[x]
                    realData[1] = data[x +1]
                    realData[2] = data[x +2]
                    realData[3] = data[x +3]
                    realData[4] = data[x +4]
                    realData[5] = data[x +5]
                    realData[6] = data[x + 6]
                    realData[7] = data[x +7]
                    realData[8] = data[x +8]
                    realData[9] = data[x +9]
                    realData[10] = data[x +10]
                    realData[11] = data[x + 11]
                    realData[12] = data[x +12]
                    realData[13] = data[x +13]
                    realData[14] = data[x +14]
                    realData[15] = data[x +15]
                    realData[16] = data[x +16]
                    realData[17] = data[x +17]
                    realData[18] = data[x +18]
                    realData[19] = data[x +19]
                    realData[20] = data[x +20]
                    realData[21] = data[x +21]
                    realData[22] = data[x - 1]

```

```
                else:
                    realData = data



        return realData



```

```python
import serial
import queue
from threading import Thread




class SerialSend ():
    def __init__(self,serialPort,semaPhore):

        #Semaphore to avoid serail send/recieve collision.
        self.semaphore = semaPhore
        #Port of the serial(USB port)
        self.serialPort = serialPort
        #Data holder for the class
        self.q = queue.LifoQueue()


    #Creates an thread of the run function, and starts it.
    def start(self):
        # Start the thread send arduino data
        t = Thread(target=self.run,name="thread2", args=())
        t.daemon = True
        t.start()

    #sends data to the serialport.
    def run(self):
        try:

            while (True):
                #checks if the queue contains any data
                if not self.q.empty():
                    dataToArduino = self.getNewDataString()
                    for x in range(0,5):
                        print(dataToArduino[x])

                    #aquires the semaphore so the serialport can be used.
                    self.semaphore.acquire()


                    print("SEND Serial " + str(dataToArduino))
```

```
                    #sends the data to the serialport
                    self.serialPort.write(dataToArduino)
                    #releases the semaphore so the readerclass can use the port.
                    self.semaphore.release()

        except serial.portNotOpenError:
                print("SerialPortException i SerialSend")

    # Updates the data to be sent to the SerialPort
    def updateDataString(self,newData):
        self.q.put(newData)

    # Returns the lates queue object,and flushes the queue. If there is none and empty array is returned.
    def getNewDataString(self):

        if not self.q.empty():
            newCommand = self.q.get()
            while not self.q.empty():
                trashBin = self.q.get()


            return newCommand
        else:
            nodata=bytearray(6)
            return nodata
```

```python
import queue
import socket
from threading import Thread
from struct import *


#Reads data sendt from GUI throug UDP com
class UDPreadGUI(Thread):

    def __init__(self,Port):
        Thread.__init__(self)
        #UDP read port
        self.port = Port
        #byte array of 6 bytes
        self.data = bytearray(6)
        #UDP socket
        self.sock = socket.socket(socket.AF_INET,  # Internet
                             socket.SOCK_DGRAM)  # UDP
        #connect to the socket with local adress
        self.sock.bind(('',self.port))
        #queue makes it possible to access data from the thread without data collision.
        self.q = queue.LifoQueue()

    # Creates an thread of the run function, and starts it.
    def start(self):

        t = Thread(target=self.run, args=())
        t.daemon = True
        t.start()


     #runs this method when thread.start is called
     #gets the data from GUI

    def run(self):
        while True:
            try:

                #reads 6 bytes of data from socket.
                self.data = self.sock.recv(6)


```

```
                for x in range(0, 6):
                    print(self.data[x])
               # print("Read UDP: " + str(data2))

                #puts the data to the queue
                self.q.put(self.data)

            except socket.error as e :
                print("uhdfuhdf")




     #retuns data recieved from GUI
     # @return data


    def getDataGUI(self):

        # Return the latest queue data and removes the rest of the data if there is any left.
        # #returns an empty array of 0 if the function is called without data in the queue.
        if not self.q.empty():
            newCommand = self.q.get()
            while not self.q.empty():
                trashBin = self.q.get()

            return newCommand
        else:

            nodata=bytearray(6)
            return nodata


    #checks if the queue contains new data.
    def queReady(self):
        if not self.q.empty():
            return True
        else:
            return False
```

```python
import queue
import socket
import time
from threading import Thread

class UDPsendData(Thread):
    # class constructor
    def __init__(self,Port,Adress):
        Thread.__init__(self)
        #UDP send port
        self.port = Port
        #Adress of the reciever
        self.address = Adress
        #data to be sent, array of 23 bytes
        self.data = bytearray(23)
        #queue storage variable for new data to be sent.
        self.q = queue.LifoQueue()




    def start(self):
        # creates and UDP socket.
        self.sock = socket.socket(socket.AF_INET,  # Internet
                             socket.SOCK_DGRAM)
        #Creates a thread of the run function and Starts the thread
        t = Thread(target=self.run, args=())
        t.daemon = True
        t.start()

    def  run(self):

        while (True):
            try:
                #checks if there is any data to be sent
                if self.queReady():

                    packet1 = self.getNewDataString()



                    self.sock.sendto(packet1, (self.address, self.port))
```

```
                    time.sleep(0.5)

            except Exception as e:
                print(e)



    #Updates the data to be sent to the GUI
    def updateDataString(self,newData):
        self.q.put(newData)


    # Returns the lates queue object. If there is none and empty array is returned.
    def getNewDataString(self):


        if not self.q.empty():
            newCommand = self.q.get()
            while not self.q.empty():
                trashBin = self.q.get()

            return newCommand
        else:
            nodata=bytearray(23)

            return nodata

    #checks if the queue contains any data.
    def queReady(self):
        if not self.q.empty():
            return True
        else:
            return False
```

```
import socket
import serial
from SerialRead import SerialRead
from SerialSend import SerialSend
from UDPsendData import UDPsendData
from UDPreadGUI import UDPreadGUI
import threading
from ThrusterControl import ThrusterControl
import time


class datahandler:
    # class constructor
    def __init__(self,readGUIPort,sendGUIDataPort,iAdress):
        #setting up gui data variables

        self.senderUDP = UDPsendData(sendGUIDataPort, iAdress)
        self.readerUDP = UDPreadGUI(readGUIPort)
        #setting up serial variables
        self.SerialSemaphore = threading.Semaphore()
        self.senderSerial = SerialSend(serialPort=serial.Serial('/dev/ttyUSB0', 19200,timeout=5),semaPhore=self.SerialSemaphore)
        self.readerSerial = SerialRead(serialPort=serial.Serial('/dev/ttyUSB0', 19200,timeout=5),semaPhore=self.SerialSemaphore)
        self.thrusterControl = ThrusterControl()
        self.startThreads()
        #command variables from GUI thruster cmnd:(fwd, back etc..)power:0-100 thruster.
        # Used in the ThrusterControl class
        self.thrusterCmd = 0
        self.thrusterPower = 0

        #starting the While loop for logic
        self.runDatahandler()


     # starting the threads when called

    def  startThreads(self):
        try:
            #thrusterControl.start();
```

```
            self.readerUDP.start()

            self.senderSerial.start()
            self.readerSerial.start()
        except RuntimeError:
            print("Cant Start Threads in Datahandler StartThread functionn")
        try:  #Starts sending to GUI threadfo
            self.senderUDP.start()

        except RuntimeError:
            print("Cant Start Threads in Datahandler StartThread senderUDP and SerialCom")




     #Setting the trustervalues
     #@param command
     #@param thrusterPower""""

    def runDatahandler(self):
        time.sleep(2)
        while True:
            if self.readerUDP.queReady():
                #get reformat data to send to arduino
                self.guiReadData= self.dataToArduino()
                #update serial data to arduino
                self.senderSerial.updateDataString(self.guiReadData)
            #get last serial read array
            if self.readerSerial.queReady():
                self.guiSendData= self.readerSerial.getSerialData()
                #send data to the gui
                self.senderUDP.updateDataString(self.guiSendData)




    #Returns the  Bytearray to be sendt to arduino
    #@return dataToArduino

    def dataToArduino(self):
```

```
        data = self.readerUDP.getDataGUI()
       #// ************** Thruster Variables ***************** //
        self.thrusterCmd = data[0]
        self.thrusterPower = data[2]
        self.thrusterControl.thrusterValues(self.thrusterCmd, self.thrusterPower)
#//*********************************************************//
        dataToArduino = bytearray(6)


        dataToArduino[0] = 101   #//Flag byte ( 101 )
        dataToArduino[1] = data[3]    #// Start byte ( 0 to 1 )
        dataToArduino[2] = self.convertToSignedValues(value=self.thrusterControl.getThrusterThree()) #// Thruster 1 ( - 100 to 100 )
        dataToArduino[3] = self.convertToSignedValues(value=self.thrusterControl.getThrusterTwo())   #// Thruster 2 ( - 100 to 100 )
        dataToArduino[4] = self.convertToSignedValues(value=self.thrusterControl.getThrusterOne())#// Thruster 3 ( - 100 to 100 )
        dataToArduino[5] = data[1]        #// Light control ( 0 - 100 ) vAR DATA[2]
        return dataToArduino


    #the arduino is using signe bytes, range: -128-127, python is using unsigned range: 0-255
    #in binary from 0-127 is the same in both arduino and python.
    # but over 128, reads as negativ by the arduino 128=-128
    #if i want to send -100, i will send 255-100 +1
    def convertToSignedValues(self,value):
        if value>=0:
            return value
        else:
            newValue= 255+value+1
            return newValue


```

```python
import time
import socket
import cv2



def videoStream(ipadress,port):
    cam = cv2.VideoCapture(0)
    #cam.set(cv2.CAP_PROP_FPS,30)
    UDP_IP = ipadress
    UDP_PORT = port
    sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM
)



    # warmup camera
    time.sleep(0.1)
    while True:
        ret_val, img = cam.read(0)

        img = cv2.flip(img, 1)
        img = cv2.resize(img, (320, 240))

        x = [int(cv2.IMWRITE_JPEG_QUALITY), 80]
        _, compressed = cv2.imencode(".jpg", img, x)
        sock.sendto(compressed, (UDP_IP, UDP_PORT))

    rawCapture.truncate(0)
```

```
class ThrusterControl(object):
    # class constructor
    def __init__(self):
        self.selfthrusterOne = 0
        self.thrusterTwo = 0
        self.thrusterThree = 0






 # sets the right thrustervalue from the command sendt from GUI.
 # @param command  can be fwd,left,right ++
 # @param power the persentage of the thrustervalue


    def thrusterValues(self, command,power):

        if (command == 2): # // FWD
            self.thrusterOne = -power
            self.thrusterTwo = power

        elif (command == 18):  # Slide FWD Right

            self.thrusterTwo = power
            self.thrusterThree = power
        elif command == 16: # Right
            calculationVar = power * 58  # ThrusterOne and ThrusterTwo need 58% of ThrusterThree's Power.
            self.thrusterOne = round(calculationVar/(100))
            self.thrusterTwo = round(calculationVar/(100))
            self.thrusterThree = power

        elif command == 20:  # Slide Back Right
            self.thrusterOne = power
            self.thrusterThree = power
        elif command == 4:# Back
            self.thrusterOne = power
            self.thrusterTwo = -power
        elif command == 12: # Slide Back Left


            self.thrusterTwo = -power
```

```
            self.thrusterThree = -power
        elif command == 8:  #Left
            calculationVar = power * 58  # ThrusterOne and ThrusterTwo need 58% of ThrusterThree's Power.
            self.thrusterOne = round(-(calculationVar/(100)))
            self.thrusterTwo = round(-(calculationVar/(100)))
            self.thrusterThree =-power
        elif command == 10: # Slide Fwd Left
            self.thrusterOne = -power
            self.thrusterThree = -power
        elif (command == 32): # Rotate Left
            self.thrusterOne = -power
            self.thrusterTwo = -power
            self.thrusterThree = power
        elif (command == 64): # Rotate Right
            self.thrusterOne = power
            self.thrusterTwo = power
            self.thrusterThree = -power
        else:
            # Do nothing
            self.thrusterOne = 0
            self.thrusterTwo = 0
            self.thrusterThree = 0



     # return the trusterone value
     # @return thrusterOne

    def getThrusterOne(self):
        return self.thrusterOne



    #return the trusterTwo value
    #return thrusterTwo

    def getThrusterTwo(self):
        return self.thrusterTwo



    #return the trusterTree value
```

```
    #return thrusterThree

    def getThrusterThree(self):
        return self.thrusterThree



```

```
import time
import socket
import cv2


cam = cv2.VideoCapture(0)
cam.set(cv2.CAP_PROP_FPS,30)
UDP_IP = '192.168.0.103'
UDP_PORT = 12342
sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)



# warmup camera
time.sleep(0.1)
while True:
    ret_val, img = cam.read(0)

    img = cv2.rotate(img,cv2.ROTATE_90_COUNTERCLOCKWISE)
    img = cv2.resize(img, (320, 240))

    x = [int(cv2.IMWRITE_JPEG_QUALITY), 80]
    _, compressed = cv2.imencode(".jpg", img, x)
    sock.sendto(compressed, (UDP_IP, UDP_PORT))

rawCapture.truncate(0)
```

Bibliography

[1] Akvakultur. URL https://www.ssb.no/fiskeoppdrett.

[2] Geilson Loureiro Adalberto Coelho Silva. System integration issues - causes, consequences mitigations. *Paper*, 2011. URL https://ieeexplore.ieee.org/document/6118134/authors#authors.

[3] Molland Anthony. The maritime engineering reference book: A guide to ship design, construction and operation, 2008.

[4] AM Basics. What is additive manufacturing? URL http://additivemanufacturing.com/basics/.

[5] Catb. Gps service daemon. URL http://catb.org/gpsd/index.html.

[6] Fargo controlls. Operating principles for inductive proximity sensors. URL https://www.fargocontrols.com/sensors/inductive_op.html.

[7] Hobbytronics. Vk-162 waterproof usb 50 channel gps receiver. URL http://www.hobbytronics.co.uk/ublox-gps-usb.

[8] The Nautical Institute. What is dynamic positioning? URL https://web.archive.org/web/20130125101320/http://www.nautinst.org/en/dynamic-positioning/what-is-dynamic-positioning/index.cfm.

[9] Mark Claypool Jae Chung and Yali Zhu. Measurement of the congestion responsiveness of realplayer streaming video over udp. *Paper*, 2003. URL https://pdfs.semanticscholar.org/90ad/0ab60631ea3d8fcb8a7673707209f106de0b.pdf.

[10] JxMaps. Jxmaps. URL https://www.teamdev.com/downloads/jxmaps/docs/index.html.

[11] Microcontrollers Lab. Pid controller implementation using arduino. URL https://microcontrollerslab.com/pid-controller-implementation-using-arduino/.

[12] Candidate number: 10008 10007 10006. Aquafarm inspection - rov. *Project report*, 30.11.2016.

[13] Candidate number: 10031 10038 10004 10027. Sea farm platform. *Bachelor thesis*, 02.06.2017.

[14] Candidate number: 10062 10055 10060 10002. Simple winch for seafarm. *Project report*, 23.11.2017.

[15] Jørgen Berge Trygstad Vegard Rogne Houxiang Zhang Ottar L. Osen, Rolf-Inge Sandvik. A novel low cost rov for aquaculture application. *Paper*, 21.09.2017.

[16] Yolande Poirier. Medusa: Gauges for javafx. URL https://community.oracle.com/docs/DOC-992746.

[17] Sparkfun. Sparkfun 9dof razor imu m0. URL https://www.sparkfun.com/products/14001.

[18] Ultimaker. Pla, . URL https://ultimaker.com/en/products/materials/pla.

[19] Ultimaker. Pla tough, . URL https://ultimaker.com/en/products/materials/tough-pla.

[20] Chris Veness. Calculate distance, bearing and more between latitude/longitude points. URL https://www.movable-type.co.uk/scripts/latlong.html.

[21] WAGO. Controller pfc100, . URL https://www.wago.com/global/plcs-%E2%80%93-controllers/controller-pfc100/p/750-8100.

[22] WAGO. Stepper controller, . URL https://www.wago.com/global/i-o-systems/stepper-controller/p/750-670.

[23] Håvard Nordahl Ørnulf Jan Rødseth. Definition for autonomous merchant ships. *Paper*, 10.10.2017. URL http://nfas.autonomous-ship.org/resources/autonom-defs.pdf.