



NTNU

Kunnskap for en bedre verden

Bacheloroppgave

Bachelor i ingeniørfag, Automatiseringsteknikk

Sea farm platform

10031, 10038, 10004, 10027

Totalt antall sider inkludert forsiden: 412

Innlevert Ålesund, 02.06.2017

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.	<input type="checkbox"/>
2.	Jeg/vi erklærer videre at denne besvarelsen: <ul style="list-style-type: none">• 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.	<input type="checkbox"/>
3.	Jeg/vi er kjent med at brudd på ovennevnte er å <u>betrakte som fusk</u> 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.	<input type="checkbox"/>
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	<input type="checkbox"/>
5.	Jeg/vi er kjent med at høgskolen vil behandle alle saker hvor det forligger mistanke om fusk etter NTNUs studieforskrift.	<input type="checkbox"/>
6.	Jeg/vi har satt oss inn i regler og retningslinjer i bruk av kilder og referanser på biblioteket sine nettsider	<input type="checkbox"/>

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 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](#))

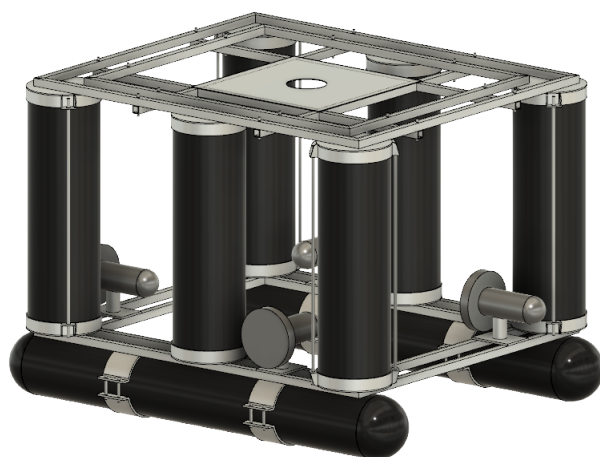
Dato: 02.06.2017



NTNU

Kunnskap for en bedre verden

Sea farm platform



June 2017

BACHELOR THESIS

Faculty of information and electrical engineering

Norwegian University of Science and Technology

Supervisors: Ottar Lauritz Osen, Dr. Houxiang Zhang

Preface

This bachelor thesis is written by four students from Automatiseringsteknikk at NTNU Ålesund, and marks the end of a three year bachelor degree. The purpose of the project is to develop a low-cost USV, capable of handling several tasks due to its versatile design.

This type of technology is mainly developed by large companies such as Rolls-Royce and Kongsberg. This is expensive technology to purchase and have high operating cost. This USV however, is meant as an affordable alternative for smaller companies that could benefit using automated surveillance or inspection at sea.

What intrigued us with this project, was that it included many of the fields that our three year study contained. Many of the topics and fields learn earlier were theoretical principles but never physically tested. This project helped us better understand many of these theories.

Acknowledgement

We would like to thank the following persons for their great help with this bachelor thesis.

- Our mentors Ottar Osen and Houxiang Zhang for help and support.
- Professor Henrique M. Gaspar and professor Karl Henning Halse for help with bouncy and stability calculation.
- Associate Professor Girts Strazdins for Linux and wireless communication assistance.
- Senior engineer Anders Sætersmoen for help with equipment and orders.
- Assistant Professor Arne Styve for software project management and software development support.
- Family and friends for time and support over the last years.
- Senior engineer Anders Sætersmoen for help with equipment and orders.
- Engineer André Tranvåg and staff for help in manufacturing parts
- Engineer Øyvind Andre Hanken for help with practical matters and equipment.
- Assistant Professor Mikael Tollefsen for android software development support.
- Pipelife Norge AS for sponsoring PE tubes.
- Anda-Olsen AS for sponsoring batteries.
- Tingstad AS for sponsoring stainless bolts.

Summary and Conclusions

This report concerns the development of a sea farm vessel, a concept by Ottar L. Osen, and is given as a bachelor thesis from NTNU in Ålesund. The purpose of this project is to develop the concept to a full-size working prototype. The prototype is meant to be the foundation of future work, towards a complete system. A detailed description regarding how to design this type of sea farm vessel is found in the report, and is focused on functionality and low-cost components. Implemented features in this project are a active self-stabilising system, autopilot and a manual manoeuvring functionality. The prototype is designed as the base for handling a remotely operated vehicle doing sub sea inspections. The control system runs on an Odroid XU4 mini-computer with the software programmed in Java. The user interface is based on a Lenovo tablet with an Android application, also programmed in Java. As for instrumentation, the Arduino is a key component with inexpensive sensors for input. The final test results shows that a full-size prototype with autopilot and active self-stabilisation, can be developed with the solutions presented.

Contents

Preface	i
Acknowledgement	ii
Summary and Conclusions	iii
Acronyms	2
1 Introductions	11
1.1 Background	12
1.2 Problem Formulation	12
1.3 Literature Survey	13
1.4 Objectives	13
1.5 Limitations	13
1.6 Approach	13
1.7 Structure of the Report	14
2 Theoretical basis	15
2.1 Buoyancy and stability	15
2.1.1 Buoyancy	15
2.1.2 Angle of list	16
2.1.3 Stability	16
2.1.4 Inertia	18
2.2 Motion variables	19
2.3 Reference Frames	20
2.3.1 Earth-Centred Reference Frames	20
2.3.2 Geographic Reference Frames	21

2.3.3 Conversion from geodetic reference frame to NED reference frame	22
2.4 Kinematics	23
2.4.1 Principal rotations	23
2.5 Mathematical optimisation	25
2.5.1 Convex Optimisation	25
2.5.2 Linear Optimisation	26
2.6 Wireless Communication	26
2.7 Dynamic Positioning	30
2.7.1 Classification	30
2.8 Euler angles	30
2.9 Haversine formula	31
2.10 PID Controller	32
2.10.1 Ziegler-Nichols method	33
2.11 Kalman filter	34
2.12 Linux	34
2.13 Android Application Programming	35
2.14 Java Programming	35
2.14.1 Thread	35
2.14.2 Concurrent Programming in Java	35
2.15 GPS	36
2.16 Inertial Measurement Unit	36
2.17 Communications Protocol	36
2.17.1 OSI Model	37
2.17.2 TCP	37
2.17.3 UDP	38
2.17.4 Internet Protocol	38
2.17.5 Socket	38
2.17.6 USB	39
2.17.7 Wireless LAN	39
2.17.8 I2C	40

2.17.9 SPI	40
3 Method	42
3.1 Project Organisation	42
3.2 Data	43
3.2.1 NMEA 0183 Standard	43
3.3 Control device	45
3.4 GUI	45
3.4.1 GUI design	45
3.5 Platform design and modelling	48
3.5.1 Materials	48
3.5.2 CAD tools	49
3.6 Stability and buoyancy calculation	49
3.7 Stability system	50
3.7.1 Stabilisation method	50
3.7.2 Control system	51
3.7.3 Sensors	51
3.8 Autopilot	52
3.8.1 Sensor	52
3.8.2 Testing the autopilot	52
3.8.3 Collision avoidance	53
3.9 Communication	54
3.10 Video stream	55
3.11 Software	55
3.11.1 NetBeans IDE 8.2	55
3.11.2 Android Studio	56
3.11.3 Arduino IDE	56
3.11.4 inSSIDer Home	56
3.12 Software development	56
3.12.1 Overview	56

3.12.2 Libraries	57
3.13 Materials	59
3.13.1 Lenovo Tab 2 A10	59
3.13.2 Odroid XU4	59
3.13.3 Linksys TL-WN722N V2	59
3.13.4 Linksys Archer c5 v2	60
3.13.5 Arduino Uno	60
3.13.6 Arduino Mega	61
3.13.7 Haswing 20	61
3.13.8 Bilge pump	62
3.13.9 Relay module	62
3.13.10 Pololu motor controller	62
3.13.11 CP1232 Battery	63
3.13.12 Multiplexer	64
3.13.13 9DoF IMU	64
3.13.14 Pressure sensor	65
3.13.15 Check valve	65
3.13.16 GPS module	65
4 Result	66
4.1 Designing the platform	66
4.1.1 Four legged version	67
4.1.2 Hexagon model 1	68
4.1.3 Hexagon model 2	69
4.1.4 Hexagon model 3	70
4.1.5 Hexagon model 4	71
4.1.6 Octagon model	72
4.1.7 Catamaran model	73
4.1.8 Rectangular model 1	74
4.1.9 Simulation-sketches	75

4.2	Data collection and calculation	75
4.3	Choosing design	77
4.4	Final design	79
4.4.1	Rectangular model 2	80
4.4.2	Rectangular model 3	81
4.4.3	Rectangular model 4	82
4.4.4	Weighing of parts	83
4.4.5	Buoyancy test in water	85
4.5	Stabilisation system	87
4.5.1	Water vs air as control medium	87
4.6	Mounting the stability system	90
4.6.1	Water pumps and hardware	90
4.6.2	Water level sensors	92
4.6.3	IMU	93
4.6.4	Control system	93
4.6.5	Stabilisation software	94
4.6.6	Stabilising time	95
4.7	Thruster allocation	96
4.7.1	Thruster configuration	97
4.7.2	Actuator Models	99
4.7.3	Solution by quadratic programming and JOptimizer	99
4.8	Software	102
4.8.1	Flow chart of the complete system	102
4.8.2	Server-Client	103
4.8.3	Float chart software	104
4.8.4	Class Diagram	107
4.8.5	Graphical User Interface	107
4.8.6	Server application	109
4.8.7	Reading sensors on the platform	109
4.8.8	Sensor data processing on the platform	110

4.8.9	Autopilot and Dynamic positioning	113
4.8.10	PID control for thrust	114
4.8.11	Thruster control	114
4.9	Results from tests at sea	114
4.9.1	Autopilot mode	115
4.9.2	Manual mode	115
4.10	Results from wave test	115
4.10.1	Results from one vs four IMUs	116
4.10.2	Kalman filter	118
4.10.3	Low-pass filter	120
4.10.4	Platform movement from waves	121
4.11	Wireless communication	122
5	Discussion	126
5.1	Test results	126
5.1.1	Platform design and buoyancy	126
5.1.2	Stability system	126
5.1.3	Software solutions	127
5.1.4	Autopilot and Dynamic Positioning	128
5.1.5	Wireless Communication system	129
5.1.6	Thruster feedback	129
5.2	Stabilisation method	129
5.3	Version control Git	130
5.4	Necessary improvements	130
5.4.1	Buoyancy	130
5.4.2	Motor controllers	130
5.4.3	Client - Server connection	130
5.5	Experiences	131
5.5.1	Size and complexity of the project	131
5.5.2	Project planning	131

5.5.3	Working as a team	131
5.6	Possible operations for a semi-submersible USV	131
6	Conclusions	133
6.1	Further development	134
	Appendices	135
A	Preproject report	135
B	Gantt diagram	135
C	Project A3	135
D	Progress report 26.01.17	135
E	Progress report 10.02.17	135
F	Progress report 24.02.17	135
G	Progress report 24.03.17	135
H	Progress report 02.05.17	135
I	Meeting report 13.01.17	135
J	Meeting report 30.01.17	135
K	Meeting report 10.02.17	135
L	Meeting report 24.03.17	135
M	Meeting report 02.05.17	135
N	Platform mechanical drawings	136
O	Electrical drawing stability system	136
P	Electrical drawing thruster control	136
Q	Stability calculation Rectangular platform	136
R	Stability calculation Hexagon platform	136
S	Server source code	136
T	Client source code	136
U	Arduino source code	136
	Bibliography	137

Terminology

PID Proportional integral derivative controller

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

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

RxTx A Serial communication library

USB Universal Serial Bus

LAN Local Area Network

Three Way Handshake A three step method to establish TCP connection

JSON object a text based standard for data exchange in computer programming.

IDE Integrated Development Environment, software for computer programming

OSI-model seven-layer model for network communication

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

ACK acknowledge message

Abbreviations

IEEE Institute of Electrical and Electronic Engineers

AES Advanced Encryption Standard

I2C Inter Integrated Circuit

Gnd Ground in electronical circuits

USV Unmanned Surface Vehicle

IMU Inertial Measurement Unit, sensor with accelerometer, gyro and magnetometer

DOF Degrees of Freedom, number of configurations for a object

DP Dynamic Positioning

ECI Earth-Centred inertial frame

ECEF Earth-Centred Reference Frame

NED North-East-Down coordinate system

BODY Body-fixed reference frame

ROV Remote Operated Vehicle

IMO International Maritime Organization

GPS Global Positioning System

DGPS Differential Global Positioning System

EGNOS European Geostationary Overlay Service

OSI Open Systems Interconnection

CPU Central Processing Unit

SPI Serial Peripheral Interface

MOSI Master Output Slave Input

MISO Master Input Slave Output

SS Slave Select

SCK/SCLK Serial Clock

List of Figures

2.1	Angle of list.	16
2.2	Pressure centres.	17
2.3	Steiner's theorem	19
2.4	Body-fixed reference points.	20
2.5	ECEF, ECI, NED and BODY reference frames	22
2.6	Rotation over yaw angle ψ	24
2.7	Rotation over pitch angle θ	24
2.8	Rotation over pitch angle ϕ	24
2.9	Omnidirectional radiation pattern	27
2.10	Omnidirectional radiation pattern with antenna gain	28
2.11	Simplified wireless communication system	28
2.12	Block diagram of a PID controller	33
2.13	Block diagram of a feedback system with process $G(s)$, PID controller and a sensor with transfer function $H_s(s)$	33
2.14	OSI-model	37
2.15	39
2.16	39
3.1	NMEA messages, GPRMC and GPGGA	43

3.2 Client design 1 Manual Mode	46
3.3 Client design 2 Manual Mode	46
3.4 Client design 3 Manual Mode	47
3.5 Client design 1 Auto Pilot	47
3.6 Arduino test rig	53
3.7 Illustration of object detection	54
3.8 Lenovo Tab 2 A10	59
3.9 Odroids XU4	59
3.10 Linksys Archer c5 v2	60
3.11 Arduino Uno	60
3.12 Arduino Mega	61
3.13 Haswing 20	61
3.14 Biltema bilge pump	62
3.15 Sainsmart 4-channel relay module	62
3.16 Pololu simple motor controller	62
3.17 CP1232 Battery	63
3.18 TCA9548A	64
3.19 Adafruit 10-DoF	64
3.20 MPX2010DP	65
3.21 Biltema check valve	65
3.22 ublock vk-162	65
4.1 Four legged version	67
4.2 Four leg Bow	67
4.3 Hexagon model 1	68

4.4 Hexagon model 1 Top	68
4.5 Hexagon model 2	69
4.6 Hexagon model 2 Bow	69
4.7 Hexagon model 3	70
4.8 Hexagon model 3 Bow	70
4.9 Hexagon model 4	71
4.10 Hexagon model 4 Top	71
4.11 Octagon model	72
4.12 Octagon model Bow	72
4.13 Catamaran model	73
4.14 Catamaran model Top	73
4.15 Rectangular model 1	74
4.16 Rectangular model 1 Top	74
4.17 Simple hexagon	75
4.18 Simple catamaran	75
4.19 Simple rectangular	75
4.20 Rectangular model 2	80
4.21 Rectangular model 2 Top	80
4.22 Rectangular model 3	81
4.23 Rectangular model 3 Top	81
4.24 Rectangular model 4	82
4.25 Rectangular model 4 Top	82
4.26 Weighing of parts	83
4.27 Battery packs are weighed	84
4.28 Finished platform weighed	84

4.29 Finished platform weighed	85
4.30 Finished platform weighed	86
4.31 Finished platform weighed	86
4.32 Test with air	87
4.33 Test with air	87
4.34 Test with water pumps	88
4.35 Test with water pumps	88
4.36 Placement of the water pumps	90
4.37 Flow capacity to pressure table	91
4.38 Ventilation hole	91
4.39 Amplifier array	92
4.40 Array of six MPX2010DP	93
4.41 Inertial measurement unit	93
4.42 Payload placement	95
4.43 Thruster configuration	97
4.44 Control force	99
4.45 General flow chart for complete system	102
4.46 Float chart client	105
4.47 Float chart server	106
4.48 Clas Diagram Client	107
4.49 Mode GUI	107
4.50 Manual mode GUI	108
4.51 Autopilot GUI	108
4.52 Flow chart sensor data	110
4.53 One vs Four IMU	116

4.54 One vs Four IMU	117
4.55 Kalman filter	118
4.56 Kalman filter	119
4.57 Low-pass filter	120
4.58 Low-pass filter	121
4.59 Low-pass filter	122
4.60 Wireless range max distance	124
4.61 Wireless range comparison	125

List of Tables

2.1	The notation of SNAME (1950) for marine vessels [11]	19
2.2	Ziegler-Nichols method	34
3.1	\$GPGGA	43
3.2	\$GPRMC	44
4.1	Model comparison	78
4.2	Overview	83
4.3	Stability system, water and air comparison	89
4.4	Stabilization before draft compansation	96
4.5	Stabilization and draft compansation	96
4.6	Stabilization with fixed payload in position A with changes in Amplitude A and period time T on the waves	96
4.7	Definition of actuators and variables	101

Chapter 1

Introduction

NTNU, Rolls Royce and Kongsberg are some of the largest companies that currently are developing autonomous water crafts and its technology in Norway. NTNU in Ålesund wish to further develop their USV. The current USV is a small dinghy fitted with a dynamic positioning system, developed through previous bachelor theses. This project aims to further develop the concept of the dinghy into a low-cost, small scale platform, with capacity to carry sensors and small ROVs. The design is inspired by a semi-submersible platform because of its great abilities to withstand waves. A small platform such as this can perform a wide range of maritime operations within sheltered waters and lakes, such as surveying, inspection of fish farms, collecting maritime sensor data and hoisting.

Challenges such as design, stability control, navigation, obstacle avoidance and a control application for android devices has been the centre of development. Many of these challenges are currently being researched by the maritime industry. In this project, evaluations have been made to keep the concept focused on low-cost and flexibility. A prototype was designed and built for testing the solutions of the stability control, navigation, obstacle avoidance and the control application.

1.1 Background

Norway is among the world's leading maritime nations, and is experiencing a new era of changes where efficiency and cost reduction are central. Development through innovation, Industry 4.0 and automation could contribute to increasing the Norwegian industry's competitive ability in the future. Automated systems used in maritime operations and autonomous maritime crafts are pioneering domains within the Norwegian industrial research, where NTNU is among the leading researching groups.

Currently in the maritime industry there is an increased number of operations, where humans are being replaced by USVs (Unmanned surface vehicles) or ASVs (Autonomous surface vehicles). This allows operations to be carried through using smaller vessels equipped with the necessary sensors and instruments. This redistribution of workforce results in less expensive production and operating costs, and a safer working environment for the operators, as they can be stationed at shore.

1.2 Problem Formulation

Would a semi-submerged autonomous platform be suitable as a data collecting or monitoring unit in coastal areas?

The project can be divided into two main sections. One part is to design and create a seaworthy prototype for maritime operations, and the other is to develop a control system for the platform.

Problems to be addressed

- Design and build new seaworthy prototype
- Develop solutions to deal with autopilot, dynamical positioning, manual control, stabilisation and object detection

1.3 Literature Survey

This thesis is based a earlier bachelor thesis "USV" - Unmanned surface vessel published in 2016.

1.4 Objectives

The Objectives for this bachelor thesis are:

1. Make a seaworthy prototype that can be used for further development
2. Implement active stabilisation
3. Create a control unit software for a suitable device
4. Implement manual control
5. Implement Autopilot and DP

1.5 Limitations

In this particular project regarding a semi sub, all necessary facilities for testing are located on, or in the immediate area around the campus. NTNU in Ålesund has a wide and long experience in the field of offshore vessels, and can provide advice and counsel. The main limitation is the time disposable for the project. This affects all applications of the platform and the depth on each topic.

1.6 Approach

In this project the group will analyse the problems and look at several ways to solve them. Then the group will test and compare the different approaches and from the result decide which solutions that solves the problems best.

1.7 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 side of the bachelor thesis. The finished product is based on principles and methods from the theory mentioned in this chapter.

Chapter 3 - Method: Contains a description of the methodology and materials that were considered throughout the thesis. Several methods were noted for all tasks, with pros and cons for each. The decision for which method used are not noted in this chapter.

Chapter 4 - Result: Contains a description of the finished prototype, which includes the thruster allocation, dynamic positioning, autopilot and the active stabilisation of the vessel. All methods used in the project are defended through either testing, or from careful research. The explanation of the developed software and programs, as well as test results are presented at the end of this chapter.

Chapter 5 - Discussion: A summary of the objectives met, and what objectives were not completed. There is also a section of what the group would have done different, in hindsight of the project period.

Chapter 6 - Conclusions: This chapter present an overall conclusion from the project, and the final answer of the thesis are given.

Chapter 2

Theoretical basis

2.1 Buoyancy and stability

2.1.1 Buoyancy

Buoyancy is a force that effects all objects that are submerged in a liquid. The main principle of this force is described in Archimedes law. It states that the upward force that is exerted on a object submerged in a liquid. Is equal to the weight of the liquid displaced by the object. The force is parallel to the force from gravity, but has the opposite direction.

The magnitude of the force depends on the volume of the object and the type of liquid. Concerning the watercraft described in this paper, the watercraft will only be operated in salt and fresh water. The density of fresh water is 1kg pr litre, while salt water has 1.025kg pr litre. This means that the platform will have a larger buoyancy force in salt water than in fresh water.

The form of the object does not affect the buoyancy only volume. For a simple cuboid form the volume is given by

$$V = W \times L \times D \quad (2.1)$$

Multiplying the volume with the density of the liquid and gravity gives the buoyancy force.

$$F_{buoyancy} = V \times \rho \times g \quad (2.2)$$

The object will float as long as the weight of the object is smaller than $F_{buoyancy}$.

As with gravity, buoyancy has a point where the combined forces of all submerged objects works through. This is called centre of buoyancy and has the notation B. [11]

2.1.2 Angle of list

The angle of list is the average angle a ship have over a period of time. Must not be confused with the roll angle, which is the angle at any given moment. The list angle changes when the centre of gravity shifts. The ship will stay at this angle until the centre of gravity is moved or countered. This happens because the buoyancy point will change according to the angle and hull form. When the new centre of gravity aligns with the buoyancy point, the forces become equal and the vessel will stop leaning over and hold the current angle. [11]

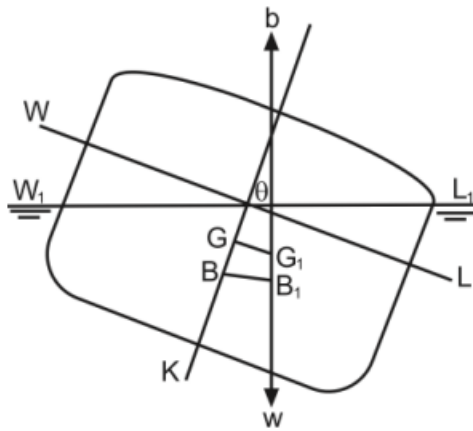


Figure 2.1: Angle of list.

2.1.3 Stability

Stability is a ships ability to right it self. A ship can also have a stable angle of list, meaning that the angle do not change. If a force has made the ship to roll, it will return to the same angle of list.

Looking at the forces righting the vessel, it is a combination of gravity and the buoyancy force. When these two forces are aligned, the forces cancel each other, and the ship will hold the current angle. [11]

Metacentre

The Meta centre is the point the buoyancy force crosses the vessels centre line. At small angles the vessel will rotate around this point. In this case a small angle is less then 10° . At larger angles the meta centre will start to move, and the stability will no longer be linear. [11]

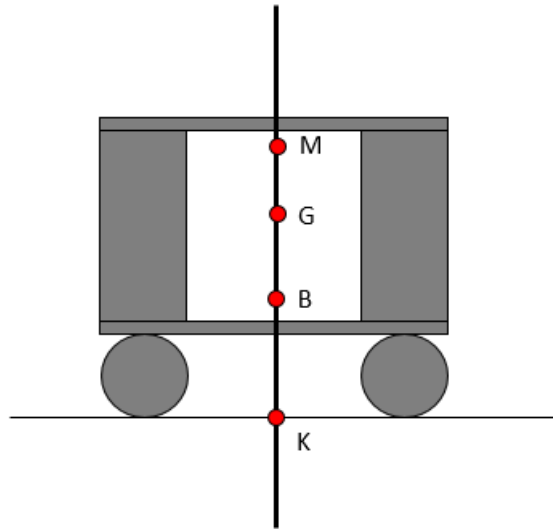


Figure 2.2: Pressure centres.

Centre of Gravity

Centre of gravity is the average location of the weight of an object and has the notation G . See figure 2.2

Keel point

The keel-point is the point where the horizontal line from G passes the lowest point of the vessel. See figure 2.2

GM

GM is the distance between the centre of gravity and the meta centre. The value of GM determines the stability of the vessel. $GM < 0$ the vessel is unstable, $GM = 0$ vessel is marginally stable and $GM > 0$ the vessel is stable.

BM

The distance between the buoyancy point and the metacentre.

GZ

GZ is the righting arm. The arm is defined at the horizontal distance between centre of gravity and the buoyancy point. This is the force that will right the vessel. A large GZ indicates good stability, but if GZ becomes negative the ship is unstable. A negative GZ will topple the vessel even at flat sea.

2.1.4 Inertia

Inertia is an objects resistance to change its speed, direction or state of rest. [11]

Moment of inertia

Determines the torque needed to overcome the inertia, and make an object rotate around a rotational axis.

2nd moment of inertia

Also known as area moment of inertia. This is the inertia an object has as seen from a plane, of-centred from the object itself.

Steiner's theorem

Steiner's theorem is for finding the 2nd moment of inertia of objects comprised of several sub objects. And the axis is not aligned with the area of mass.

$$I_z = I_x + Ad^2$$

Where I_x the inertia of the object through its mass centre. A is the area of the object and d is the distance between the plane through the mass centre and the of-centred plane.

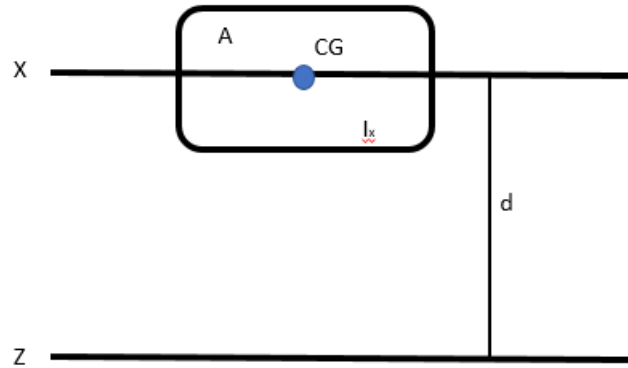


Figure 2.3: Steiner's theorem

2.2 Motion variables

When manoeuvring a marine craft, it experiences motions in 6 degrees of freedom. The 6 degrees of freedom can be represented by a set of independent displacements and rotations that describe the craft's position and rotation. Motions in the horizontal plane are *surge*(motion along the X-axis), *sway*(motion along the Y-axis) and *yaw*(rotation about the Z-Axis). The remaining three degrees of freedoms are *roll* (rotation about the X-axis), *pitch* (rotation about the Y-axis) and *heave*(vertical motion along Z-axis) [11].

Table 2.1: The notation of SNAME (1950) for marine vessels [11]

DOF		Forces and moments	Linear and angular velocities	Positions and Euler angles
1	motions in the x direction (surge)	X	u	x
2	motions in the y direction (sway)	Y	v	y
3	motions in the z direction (heave)	Z	w	z
4	rotation about the x axis (roll, heel)	K	p	ϕ
5	rotation about the y axis (pitch, trim)	M	q	θ
6	rotation about the z axis (yaw)	N	r	ψ

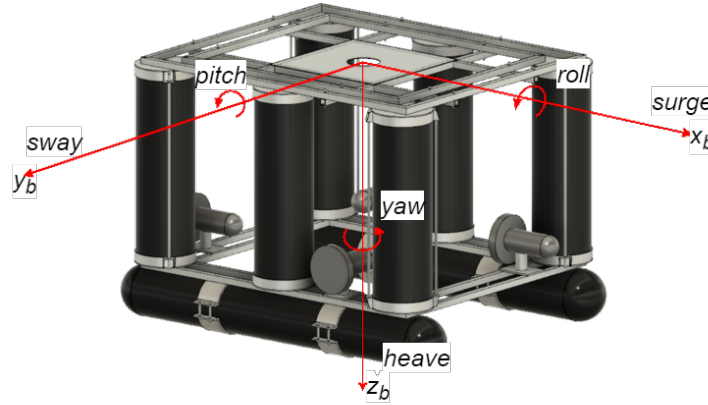


Figure 2.4: Body-fixed reference points.

2.3 Reference Frames

When analysing guidance systems, and the motion of marine crafts, it's convenient to define several coordinate systems.

2.3.1 Earth-Centred Reference Frames

ECI

The Earth-centred inertial (ECI) frame $\{i\} = \{x_i, y_i, z_i\}$ is an inertial frame for terrestrial navigation, that is a nonaccelerating reference frame in which Newton's laws of motion apply.

ECEF

The Earth-centred Earth-fixed (ECEF) reference frame $\{e\} = \{x_e, y_e, z_e\}$ has its origin o_e fixed to the centre of the Earth but the axes rotate relative to the inertial frame ECI, which is fixed in space. The angular rate of rotation is $\omega_e = 7.2921 \times 10^{-5}$ rad/s. For marine craft moving at rel-

atively low speed, the Earth rotation can be neglected and hence $\{e\}$ can be considered to be inertial.

2.3.2 Geographic Reference Frames

NED

The North-East-Down (NED) coordinate system $\{n\} = \{x_n, y_n, z_n\}$ Where the X- and Y- axis define a tangent plane on the surface of the Earth moving with the craft. For this system the X-axis points towards true North, and the Y-axis points towards East. The Z-axis points downwards normal to the Earth's surface. Origin is located in the center of the craft. The location of $\{n\}$ relative to $\{e\}$ is determined by using two angles l and μ denoting the longitude and latitude, respectively. For a marine craft operating in a local area, with approximately constant longitude and latitude, can one assume that $\{n\}$ is inertial such that Newton's laws still apply.

BODY

The body-fixed reference frame $\{b\} = \{x_b, y_b, z_b\}$ is a moving coordinate frame that is fixed to the craft. The position and orientation of the craft are described relative to the inertial reference frame usually approximated $\{n\}$ for marine craft, while the linear and angular velocities are described as $\{b\}$. The origin O_b is usually chosen to coincide with a point midships in the water line. This point will be referred to as CO (see Figure 2.4) [11]

The reference frames are illustrated in figure (2.5), which is retrieved from [10].

- x_b - longitudinal axis (directed from aft to fore)
- y_b - transversal axis (directed to starboard)
- z_b - normal axis (directed from top to bottom)

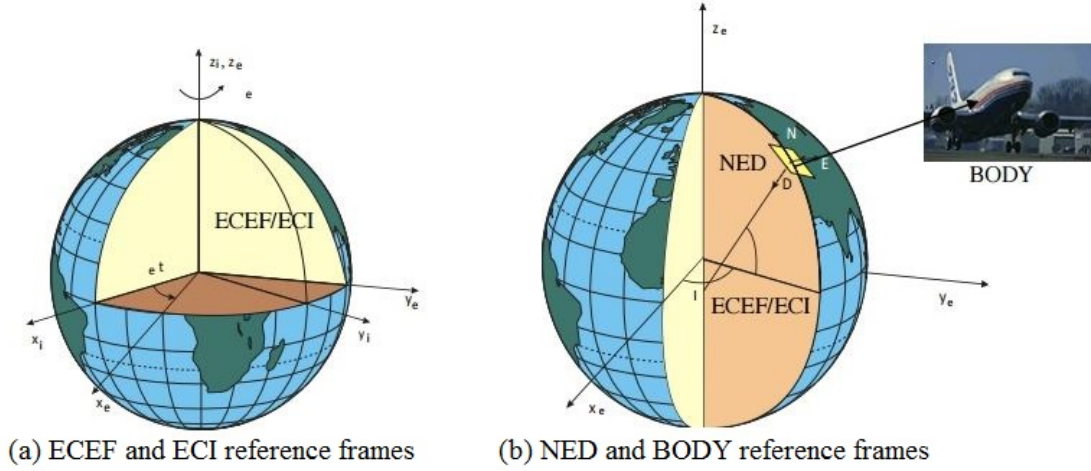


Figure 2.5: ECEF, ECI, NED and BODY reference frames

2.3.3 Conversion from geodetic reference frame to NED reference frame

The conversion between geodetic and NED reference frames, begins with finding small changes in latitude $d\mu$ and longitude dl by calculating the difference between the new values μ and l , and the initial reference values μ_0 and l_0 .

$$d\mu = \mu - \mu_0 \quad (2.3)$$

$$dl = l - l_0 \quad (2.4)$$

To convert geodetic coordinates to North-East coordinates, the radius of the curvatures in the prime vertical (R_N) and the meridian (R_M) are used. R_M and R_N are defined by the following relationships. [26]

$$R_N = \frac{R}{\sqrt{1 - (2f - f^2) \sin^2(\mu_0)}} \quad (2.5)$$

$$R_M = R_N \frac{1 - (2f - f^2)}{\sqrt{1 - (2f - f^2) \sin^2(\mu_0)}} \quad (2.6)$$

where R is the equatorial radius of the earth and f is it's flattening. Small changes in North (dN) and East (dE) are approximated from small changes in the North and East positions by:

$$dN = \frac{d\mu}{\arctan(\frac{1}{R_M})} \quad (2.7)$$

$$dE = \frac{dl}{\arctan(\frac{1}{R_N \cos \mu_0})} \quad (2.8)$$

2.4 Kinematics

It is customary to describe $R_b^n(\Theta_{nb})$ by three principal rotations about the z, y and x axes (zyx convention). Note that the order in which these rotations is carried out is not arbitrary. In guidance, navigation and control applications it is common to use the zyx convention from $\{n\}$ to $\{b\}$ specified in terms of the Euler angles ϕ , θ and ψ for the rotations.[11]

2.4.1 Principal rotations

Rotation

Euler angle rotation sequence (zyx convention). The vessel is rotated from $\{n\}$ to $\{b\}$ by using three principal rotations.[11]

This matrix is denoted $R_b^n(\Theta_{nb}) = R_b^n(\Theta_{nb})^T$. The matrix transpose implies that the same result is obtained by transforming a vector from $\{b\}$ to $\{n\}$, that is by reversing the order of the transformation. This rotation sequence is mathematically equivalent to

$$R_b^n(\Theta_{nb}) = R_{z,\psi} R_{y,\theta} R_{x,\phi} \quad (2.9)$$

and the inverse transformation is then written (zyx convention)[11]

$$R_b^n(\Theta_{nb})^{-1} = R_b^n(\Theta_{nb})^{-1} = R_{z,\psi}^T R_{y,\theta}^T R_{x,\phi}^T \quad (2.10)$$

The three rotation matrices $R_b^n(\Theta_{nb})$ is presented by rotating on each of the xyz axis. These three matrices can be merged to one rotation matrix.[11]

$$\begin{bmatrix} c\psi & -s\psi & 0 \\ s\psi & c\psi & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Principal rotation matrix for Z axis:

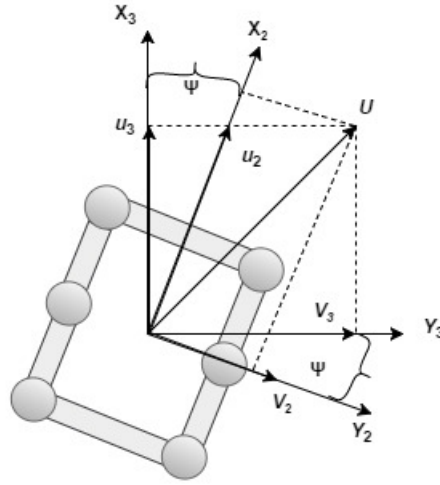


Figure 2.6: Rotation over yaw angle ψ

$$\begin{bmatrix} c\theta & 0 & s\theta \\ 0 & 1 & 0 \\ -s\theta & 0 & c\theta \end{bmatrix}$$

Principal rotation matrix for Y axis:

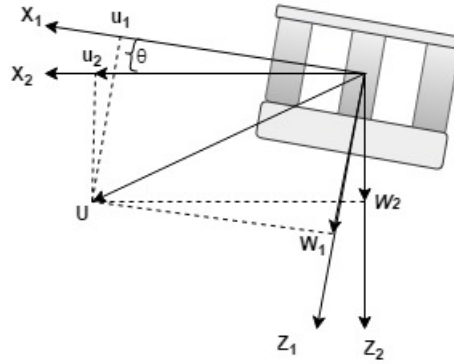


Figure 2.7: Rotation over pitch angle θ

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & c\phi & -s\phi \\ 0 & s\phi & c\phi \end{bmatrix}$$

Principal rotation matrix for X axis:

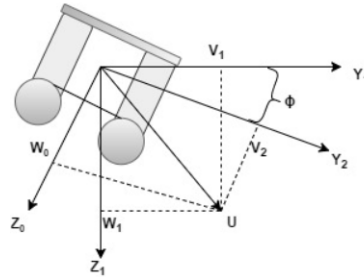


Figure 2.8: Rotation over pitch angle ϕ

Rotational matrix from BODY - coordinates to NED - coordinates yields:

$$R_b^n(\Theta_{nb}) = \begin{bmatrix} c\psi c\theta & -s\psi c\theta + c\psi s\theta s\phi & s\psi s\phi + c\psi c\phi s\theta \\ s\psi c\theta & c\psi c\phi + s\phi s\theta s\psi & -c\psi s\phi + s\theta s\psi c\phi \\ -s\theta & c\theta s\phi & c\theta c\phi \end{bmatrix} \quad (2.11)$$

2.5 Mathematical optimisation

Mathematical optimising is a method to determine a set of optimised minimum or maximum values for a set of variables.[16] A mathematical optimization problem, has the general form

$$\begin{aligned} &\text{minimise} && f_0(x) \\ &\text{subject to} && f_i(x) \leq b_i, \quad i = 1, \dots, m. \end{aligned} \quad (2.12)$$

Where $x = (x_1, x_2, \dots, x_n)$ is the optimisation variable of the problem, $f_0 : \mathbb{R}^n \rightarrow \mathbb{R}$ is the objective function, $f_i : \mathbb{R}^n \rightarrow \mathbb{R}, i = 1, \dots, m$, are the constraint functions and the constants b_1, \dots, b_m are the limits for the constraints.[5]

2.5.1 Convex Optimisation

A convex optimisation problem is a problem where the objective and constraint functions satisfy the inequality

$$f_i(\alpha x + \beta y) \leq \alpha f_i(x) + \beta f_i(y) \forall x, y \rightarrow \mathbb{R}^n \text{ and } \forall \alpha, \beta \rightarrow \mathbb{R} \text{ with } \alpha + \beta = 1, \alpha \geq 0, \beta \geq 0 \quad (2.13)$$

There are no general analytic formula to solve convex optimization problems, but there are very effective algorithms to solve them.

A special form of convex optimisation which can be relevant for this project is on the form

$$\begin{aligned}
 &\text{minimise} && f_0(x) \\
 &\text{subject to} && f_i(x) \leq b_i, \quad i = 1, \dots, m. \\
 &&& Ax = b
 \end{aligned} \tag{2.14}$$

Where A is a matrix $A \rightarrow \mathbb{R}^{q \times n}$ with rank $p < n$. Such problem is classified as a quadratic programming problem. And can be solved by inner points method, which is explained in detail in Convex Optimization, by Stephen Boyd and Lieven Vandenberghe, page 561 [5]

2.5.2 Linear Optimisation

Another important class of optimisation problems is linear optimisation. All the objective and all the constraint functions are linear:

$$\begin{aligned}
 &\text{minimise} && c^T x \\
 &\text{subject to} && a_i^T x \leq b_i, \quad i = 1, \dots, m.
 \end{aligned} \tag{2.15}$$

The vectors $c, a_1, \dots, a_m \in \mathbb{R}^n$ and scalars $b_1, \dots, b_m \in \mathbb{R}$ are parameters that specify the objective and constraint functions. As with convex problems, there are no simple analytic formula for a solution. A variety of very effective algorithms can be used for solving linear optimisation problems, such as the Dantzig's simplex method which is explained in [5]

2.6 Wireless Communication

Antenna

An antenna is a device used to transmit and receive electromagnetic waves. In a transmitting antenna, high frequency, alternating currents are transformed into electromagnetic waves, which travels through space at the speed of light. Radio waves from the transmitting antenna will then induce electrical currents and voltages in a receiving antenna. [15]

There are many types of antennas, and they can be classified in many ways.

1. Shapes or geometries

- (a) Wire antennas: dipole, loop, helix
- (b) Aperture antennas: horn, slot
- (c) Printed antennas: patch, printed dipole, spiral

2. Gain

- (a) High gain: dish
- (b) Medium gain: horn
- (c) Low gain: dipole, loop, slot, patch

3. Beam shapes

- (a) Omnidirectional: dipole, monopole
- (b) Pencil beam: dish
- (c) Fan beam: array

Monopole and dipole antennas are great for broadcasting wireless signals due to their omnidirectional radiation pattern, while yagi and parabolic dish antennas are pencil and fan beam antennas which are more sensitive to alignment, but provide better range properties.[7] An omnidirectional radiation pattern is illustrated in figure 2.9 and a omnidirectional radiation with more gain is illustrated in figure 2.10. The illustrations 2.9 and 2.10 are retrieved from [36]

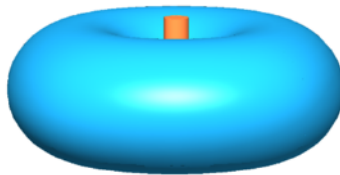


Figure 2.9: Omnidirectional radiation pattern

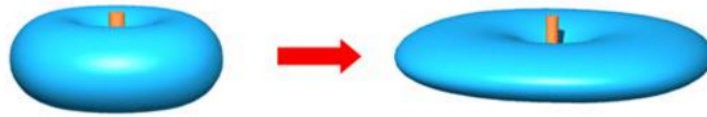


Figure 2.10: Omnidirectional radiation pattern with antenna gain

Wireless range

The theoretical range between a transmitter and a receiver of WI-FI and other radio signals, illustrated in Figure 2.11, can be calculated by the Friis transmission equation. 2.22

Where P_t is the output power that is fed to the transmitting antenna G_t . On the other end the signal is picked up by a receiving antenna with gain G_r . The received power is P_r and the distance is R . By assuming there is no atmospheric loss, polarisation mismatch. Impedance mismatch at the antenna feeds, misalignment and obstructions. The antennas are operating in the far-field regions. A far-field region is the region far away from the transmitting antenna, where the transmitting signals of the antenna resemble a spherical wave fronts coming from a point. The formula is derived from several equations, retrieved from the book *RF and Microwave Wireless Systems*, by KAI CHANG section 8.2[7].

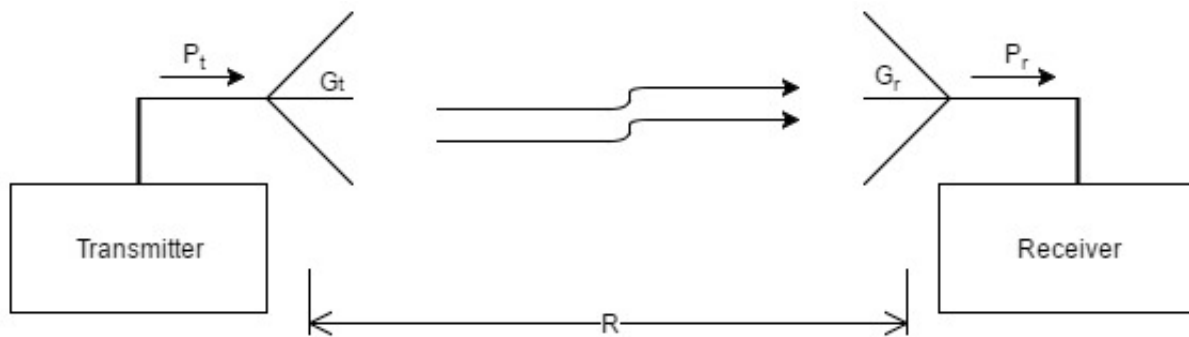


Figure 2.11: Simplified wireless communication system

The power density p of a receiving antenna for an omnidirectional transmitting antenna is given by

$$p = \frac{P_t}{4\pi R^2} \quad (W/m^2) \quad (2.16)$$

Since an omnidirectional antenna is used, it has a transmitting gain G_t in the direction of the

receiving antenna, the power density is given by

$$p = \frac{P_t}{4\pi R^2} G_t \quad (W/m^2) \quad (2.17)$$

The received power is equal to the power density multiplied by the effective area A_{er} of any antenna, where A_{er} for any receiving antenna can be expressed as

$$A_{er} = \frac{G\lambda_0^2}{4\pi} \quad (2.18)$$

$$P_r = \frac{P_t G_t}{4\pi R^2} A_{er} \quad (W) \quad (2.19)$$

The effective area is related to the antenna gain by the following expression:

$$G_r = \frac{4\pi}{\lambda_0^2} A_{er} \quad (2.20)$$

$$A_{er} = \frac{G_r \lambda_0^2}{4\pi} \quad (2.21)$$

By substituting equation 2.21 into equation 2.19 we get the expression 2.22

$$P_r = \frac{P_t G_t G_r \lambda_0^2}{(4\pi R)^2} \quad (2.22)$$

known as the Friis transmission formula. If $P_r = S_{s,min}$, the minimum signal required for the system, we have the maximum range R given by:

$$R = \left[\frac{P_t G_t G_r \lambda_0^2}{(4\pi)^2 S_{i,min}} \right]^{1/2} \quad (2.23)$$

One can also include various factors such as misalignment, polarisation mismatch, impedance mismatch, and atmospheric loss as a variable L_{sys} .

Range derived with the factor for different types of loss:

$$R = \left[\frac{P_t G_t G_r \lambda_0^2}{(4\pi)^2 S_{i,min} L_{sys}} \right]^{1/2} \quad (2.24)$$

2.7 Dynamic Positioning

Dynamical positioning is used to keep a vessel such as a boat, rig or drone to stay put at a fixed position. In the maritime industry DP is controlled by thrusters, which makes it easier to keep stationed both at shallow waters and far out on the ocean without anchor handling. The DP system takes sensor data from currents and waves into consideration, thus it can predict the direction of the drift and counteract it with delegation of thruster power and thruster direction.[34]

2.7.1 Classification

Based on IMO - International Maritime Organization publication 645 the Classification Societies [24] have issued rules for dynamically positioned ships described as Class 1, Class 2 and Class 3.

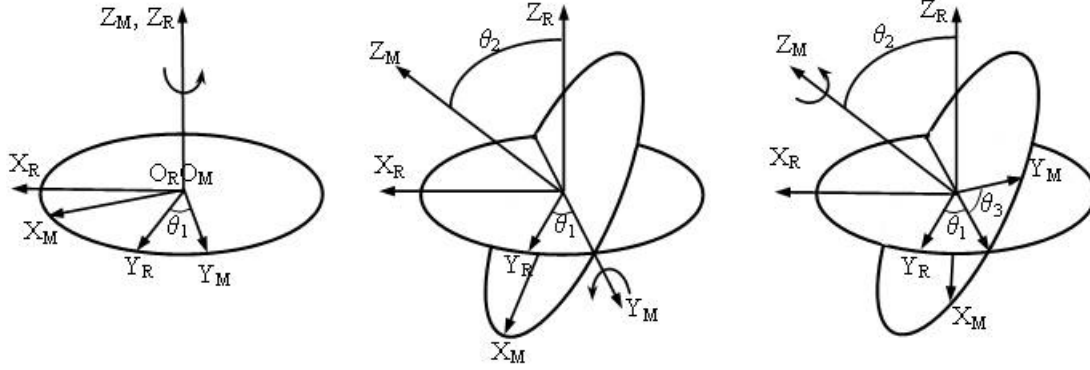
Equipment Class 1 has no redundancy. Loss of position may occur in the event of a single fault

Equipment Class 2 has redundancy so that no single fault in an active system will cause the system to fail. Loss of position should not occur from a single fault of an active component or system such as generators, thruster, switchboards, remote controlled valves etc. But may occur after failure of a static component such as cables, pipes, manual valves etc.

Equipment Class 3 which also has to withstand fire or flood in any one compartment without the system failing. Loss of position should not occur from any single failure including a completely burnt fire sub division or flooded watertight compartment

2.8 Euler angles

According to Euler's rotation theorem, any rotation may be described using three angles. If the rotations are written in terms of rotation matrices **D**, **C**, and **B**, then a general rotation **A** can be written as [27]



$$\mathbf{A} = \mathbf{B} * \mathbf{C} * \mathbf{D}$$

The three angles giving the three rotation matrices are called Euler angles. There are several conventions for Euler angles, depending on the axes about which the rotations are carried out. Write the matrix \mathbf{A} as [27]

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix}$$

The so-called "x-convention," illustrated above, is the most common definition. In this convention, the rotation given by Euler angles (ϕ, θ, ψ) , [27]

1. the first rotation is by an angle ϕ about the z-axis using \mathbf{D} ,
2. the second rotation is by an angle θ in $[0, \pi]$ about the former x-axis (*now* x') using \mathbf{C} , and
3. the third rotation is by an angle ψ about the former z-axis (*now* z') using \mathbf{B} .

2.9 Haversine formula

In navigation, a widely used formula of trigonometry is used to calculate the distance between two points on a sphere. This formula is called the haversine formula [31] and is given by:

$$hav\left(\frac{d}{r}\right) = hav(\varphi_2 - \varphi_1) + \cos(\varphi_2)\cos(\varphi_1)hav(\lambda_2 - \lambda_1) \quad (2.25)$$

- *hav* is the *haversine function* give by:

$$hav(\theta) = \sin^2\left(\frac{\theta}{2}\right) = \frac{1 - \cos(\theta)}{2} \quad (2.26)$$

- d is the distance between two points
- r is the radius of the sphere
- φ_1, φ_2 is latitude of point 1 and 2, in
- λ_1, λ_2 is longitude of point 1 and 2, in radians

2.10 PID Controller

In industrial control systems the PID controller [14] (Proportional - Integral - Derivative controller) is a widely used feedback system used to calculate the error between a wanted value and the actual value. In a time domain, the output signal can be described as:

$$y(t) = k_p \times x(t) + k_I \times \int x(t) dt + k_D \times \frac{d_x(t)}{dt} \quad (2.27)$$

The constants K_p , K_I and K_D are adjusted accordingly to reach the wanted value from the feedback system. In the Laplace domain, the transfer function of a PID controller is written as:

$$\frac{Y(s)}{X(s)} = K_p + \frac{K_I}{s} + K_D s = \frac{K_p s + K_I + K_D s^2}{s} = \frac{K_D \left(s^2 + \frac{K_p}{K_D} s + \frac{K_I}{K_D}\right)}{s} \quad (2.28)$$

The proportional component, weighted by k_p , is the gain factor. k_p determines the overall loop gain, and a large loop gain leads to low sensitivity, small steady-state error, and good disturbance rejection. Unfortunately, k_p usually has some upper limits, either physical limits or stability and overshoot limits.

The integral component is responsible for eliminating the steady-state error. We know that an

integrator can only reach equilibrium if its input is zero. If the input of the PID controller is the control deviation error, the integrator component will continue changing the corrective action until error = 0. Since the integrator component contains energy storage, it can lead to undesirable transient responses and even instability. Because of this, k_I are in most cases kept relatively small.

The derivative component helps providing a rapid transient response. If a transient disturbance occur, the first derivative is huge and causes a correspondingly strong control action. The derivative component also improves relative stability.

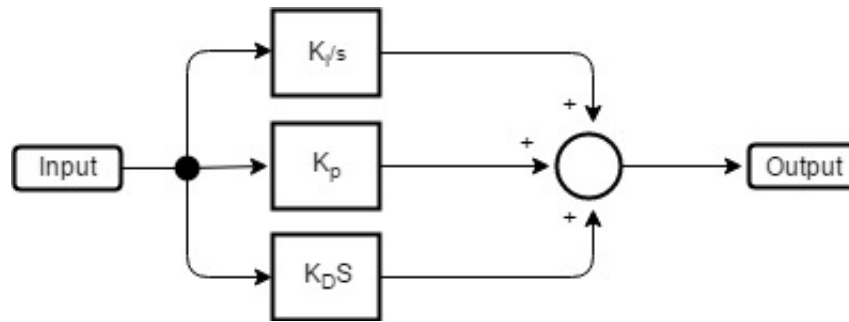


Figure 2.12: Block diagram of a PID controller

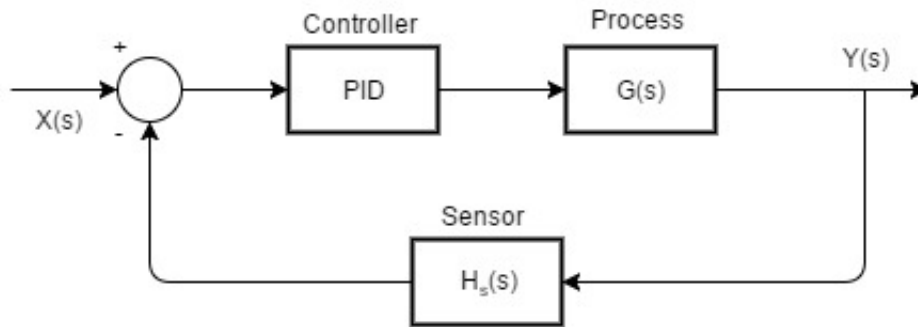


Figure 2.13: Block diagram of a feedback system with process $G(s)$, PID controller and a sensor with transfer function $H_s(s)$

2.10.1 Ziegler-Nichols method

The Ziegler-Nicholas method [19] is a step by step method to tune a PID. First set all values to zero. The proportional gain is then increased from zero to it comes to the ultimate gain. The ultimate gain is when the output has a stable oscillation. This has the notation K_u and the oscil-

lation period is T_u . K_p , K_i and K_d is then calculated from a table.

Table 2.2: Ziegler-Nichols method

	K_p	T_i	T_d
P-regulator	$0,5K_u$	∞	0
PI-regulator	$0,45K_u$	$T_u/1,2$	0
PID-regulator	$0,6K_u$	$T_u/2$	$T_u/8 = T_i/4$

2.11 Kalman filter

Is a filter algorithm co-designed by Rudolf E. Kalman and thus was given his name. The filter makes an estimate of the real value that is more correct than the real value. The input may have noise and or other errors. Such as the accelerometer that is very sensitive, and therefore the output will tend to vary a lot. The Kalman filter [21] measures the signal over time and calculates a estimate of the signal as a new output. The filter only work with a signal that has statistical noise and other inaccuracies. It is widely used in just about all disciplines from science to economics.

2.12 Linux

Linux is an open source operating system, developed by the Finnish student Linus Torvalds during the mid 90s as a hobby. Linux was originally developed for the Intel x86architecture. The popularity of Linux has grown proportional to it's functionality, which have lead to faster development of the operating system. Today, Linux is most versatile operating system. Linux is used on platforms such as mini computers, personal computers, super computers, servers, as well as the core of the Android operating system used on smartphones.

2.13 Android Application Programming

Android Studio [1] is the official integrated development environment (IDE) for Android platform development. Based on JetBrains' IntelliJ IDEA software, Android Studio is designed specifically for Android development. It is available for download on Windows, Mac OS X and Linux, and replaced Eclipse Android Development Tools (ADT) as Google's primary IDE for native Android application development.

2.14 Java Programming

Java is a general-purpose, class-based object-oriented computer programming language developed by James Gosling and several other software developers at Sun Microsystems.

2.14.1 Thread

A thread [2] is a thread of execution in a program. The Java Virtual Machine allows an application to have multiple threads of execution running concurrently. Every thread has a priority. Threads with higher priority are executed in preference to threads with lower priority.

2.14.2 Concurrent Programming in Java

A concurrent Java program is a program made up of two or more threads that cooperate to a common goal.

In a concurrent program, multiple threads can share CPU access, and which thread that gets CPU time is decided by a Scheduler. A thread can have access to any object in the program, and to prevent data corruption thread safety is important. This is done by using semaphores or synchronised access to parts of the code. This prevents multiple threads from reading or writing to a shared resource. Concurrent programs allow to leverage multiple cores of a modern CPU.[37]

2.15 GPS

GPS, Global Positioning System is a satellite based radio navigation system owned by the U.S. Government, and run by the American Department of Defence. The GPS system function by measuring the travel time of the signal from the satellite to a GPS receiver. The receiver's position and time is computed from set of equations with four unknown variables. In order to solve the equations, the receiver need at least GPS signals from four GPS satellites. By using just one receiver, it is realistic to achieve an accuracy of 5 to 10m [9].

To improve the accuracy of the location and time further, a technology called Differential GPS (DGPS) can be used. This technology is based on using a known position on earth to supplement and correct the inaccuracy of the GPS signal. With this correction of the inaccuracy, it is possible to determine the position of the GPS receiver within 1.5 meters. In Europe, this technology is called EGNOS (European Geostationary Overlay Service). The EGNOS Open Service has been available since 2009, and the EGNOS Safety of Live Service was officially declared available in March 2011. [3]

The development of the GPS started in the late 1960s, and it was designed as a military system. The GPS was declared operational in 1993, and it consist of 24 satellites orbiting the earth at approximately 20.2 kilometre. Today more than 90% of the user base are civilians. [9]

2.16 Inertial Measurement Unit

An inertial measurement unit is an electronic device, capable of measuring the specific forces applied to a body, and reports them. The device is a combination of accelerometers, gyrometers an sometimes magnetometer. [23]

2.17 Communications Protocol

A communications Protocol defines the format and the order of messages exchanged between two or more communicating entities in a network, aswell as actions taken on the transmission and/or receipt of a message or other events. [25]

2.17.1 OSI Model

The Open Systems Interconnection(OSI) model is a seven-layer model for network communication. Illustrated in figure 2.14.Each layer offer different services by either performing actions or using the services from the layer directly below it. [25]

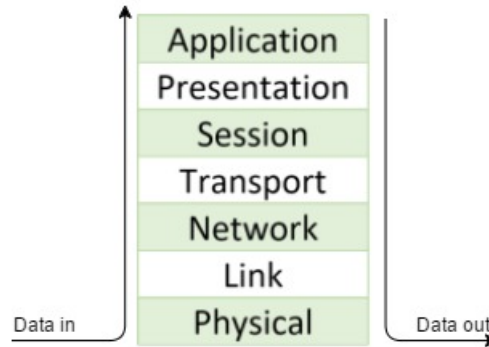


Figure 2.14: OSI-model

2.17.2 TCP

The Transmission Control Protocol(TCP) operate in the network layer, and together with the Internet Protocol(IP) they make up two of the most important protocols in the Internet. Thus the Internet's protocols are collectively known as TCP/IP. TCP is a control procedure between sender and receiver of data packets in a network, and ensure the transmitted datagram is delivered without packet loss and in correct order. In order to ensure no packet loss, TCP use a "three-way-handshake". A Three way handshake is a procedure where the client sends a small TCP segment to the server, the server acknowledges(ACK) and responds with a small TCP segment, and, finally the client acknowledges back to the to the server. Combined with the last ACK from the client to the server, the client also requests the datagram of interest. The TCP connection remains open until the client choose to close it. [25] Internally in the transport layer, the data packets are transmitted accordingly to the control procedure Internet Protocol(IP). [6]

2.17.3 UDP

The UDP protocol provides for communication that is not guaranteed between two applications on the network. UDP is not connection-based like TCP. Rather, it sends independent packets of data, called datagrams, from one application to another. Sending datagrams is much like sending a letter through the postal service: The order of delivery is not important and is not guaranteed, and each message is independent of any other. [8]

2.17.4 Internet Protocol

Internet Protocol(IP) operate in the network layer in the OSI-model 2.14. The IP protocol defines fields in the datagram as well as how end systems in a network act on these fields. There are two versions of the IP protocol, IPv4 and IPv6, and every component with a network layer must run the IP protocol. The IP service model is a “best-effort delivery service”, which means IP make its “best-effort” to deliver segments between communication hosts, but it makes no guarantees. IPv4 is the most common and uses a 32-bit address space, and the successor, IPv6 uses a 128-bit address space. The reason for the change in address type was that the pool of unallocated IPv4 addresses were running out. [25]

2.17.5 Socket

Definition: A socket is one endpoint of a two-way communication link between two programs running on the network. A socket is bound to a port number so that the TCP layer can identify the application that data is destined to be sent to. [29]

Normally, a server runs on a specific computer and has a socket that is bound to a specific port number. The server just waits, listening to the socket for a client to make a connection request.

On the client-side: The client knows the hostname of the machine on which the server is running and the port number on which the server is listening. To make a connection request, the client tries to rendezvous with the server on the server's machine and port. The client also needs to identify itself to the server so it binds to a local port number that it will use during this con-

nection. This is usually assigned by the system.

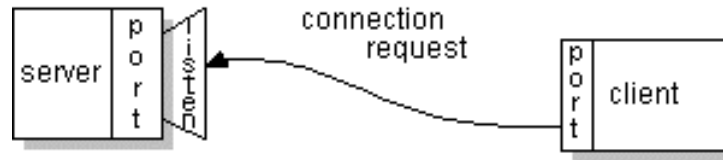


Figure 2.15

If everything goes well, the server accepts the connection. Upon acceptance, the server gets a new socket bound to the same local port and also has its remote endpoint set to the address and port of the client. It needs a new socket so that it can continue to listen to the original socket for connection requests while tending to the needs of the connected client.

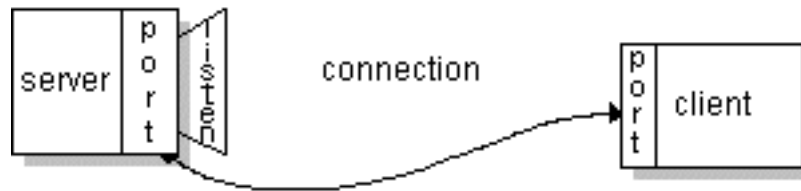


Figure 2.16

On the client side, if the connection is accepted, a socket is successfully created and the client can use the socket to communicate with the server. The client and server can now communicate by writing to or reading from their sockets.

2.17.6 USB

Universal Serial Bus(USB) is a standard for data transfer between computers and peripheral units. USB replaced the majority of serial and parallel ports on computers, and simplified the installation process on new peripheral units. USB 2.0, redefined in 2000 can reach data transfer rates up to 480 Mbps, while USB 3.0 defined in 2008 has data transfer rates 10 times faster. [32]

2.17.7 Wireless LAN

Wireless LAN(Local Area Network) more known as WiFi. WiFi is based on IEEE(Institute of Electrical and Electronic Engineers) 802.11 technology, and operates in the frequency range from

2.4-2.485 GHz and from 5.1-5.8 GHz. There are several 802.11 standards for WiFi. The standards 802.11b/a/g operates in the 2.4GHz band and the 802.11n/ac standard can operate in both, 2.4 GHz and 5GHz frequency bands. A Wireless LAN should also have some sort of encryption to ensure some level of data privacy. WEP (Wired Equivalent Protection) is ranked as the lowest type of protection and WPA2 (Wi-Fi Protected Access II + AES (Advanced Encryption Standard) as the highest, where AES is an encryption algorithm. The first IEEE specification was released in 1997 and had a air data rate of maximum 2 Mbps. Today air data rates are rated to more than 6 Gbps. [25]

2.17.8 I2C

The Inter-integrated Circuit (I2C) [13] is a multi-master, multi-slave, serial computer bus invented in order to simplify the board schematics, thanks to the fact that it needs two wires only (apart from the GND) to do its job. It's widely used in embedded computers to connect on-board sensors/actuators to the main CPU.

Despite the fact that the I2C bus is a multi-master, a typical configuration is a single master device (the CPU) connected to several slave devices (the sensors/actuators); for the USB bus, the master directs all the transfers. However, a main difference should be outlined: an I2C device can have a dedicated interrupt line connected to the CPU that can be used to signal that a message must be read by the master (in the USB bus, the interrupt messages go over the bus too!). So, a simple I2C connection needs two wires while they only, in case of interrupt lines, need three or more lines.

2.17.9 SPI

The Serial Peripheral Interface bus (SPI) is a serial communication specification mostly used for short distance communication between a microcontroller unit and peripheral devices. The interface includes Master and Slave mode, bi-directional mode, slave select output, mode fault error flag with CPU interrupt capability, double-buffered data register, serial clock with programmable polarity and phase, control of SPI operation during wait mode.

The signal pins on the SPI module is defined as follows:

MOSI (Master Output Slave Input):

This pin is used to transmit data out of the SPI module when it is configured as a Master and receive data when configured as Slave.

MISO (Master Input Slave Output):

This pin is used to transmit data out of the SPI module when configured as a Slave and receive data when configured as Master.

SS (Slave Select):

This pin is used to output the select signal from the SPI module to another peripheral with which a data transfer is to take place when its configured as a Master, and its used as an input to receive the Slave select signal when the SPI is configured as Slave.

SCK/SCLK (Serial Clock):

This pin is used to output the clock with respect to which the SPI transfers data or receive clock in case of Slave [18]

Chapter 3

Materials and methods

3.1 Project Organisation

The group consists of a permanent Project Leader, and a rotation of the secretary role between the remaining group members. Decisions concerning the project has been taken in plenary sessions, where every group member has shared their opinion. The group has had internal meetings when necessary, for discussing problems. Meetings with the supervisors has been held approximately every 14 days. A Gantt form has been the basis of the project planning as well as the SCRUM managing tool Jira, a lean software development method.

To maintain a great dynamics in the group, it was agreed to specify certain guidelines, such as when and where to show up in the mornings, and what to do if one got stuck on a topic. The project is partly separated in to several subjects. Each member was given the responsibility for their own subject. By doing this, everyone is on point for making their part of the project as good as possible.

3.2 Data

3.2.1 NMEA 0183 Standard

National Marine Electronics Association has defined a standard data format for marine equipment, such as GPS, AIS, Sonar and Autopilot etc. All the positioning data for the platform is read from a GPS as NMEA messages. There are many different types of NMEA messages, \$GPGGA, \$GPRMC, \$GPGLL to list a few. \$GPGGA, \$GPRMC, \$GPGLL have a couple of common data fields, such as longitude, latitude, time and a checksum. The remaining fields differ. Every NMEA message start with a \$, and the data fields are separated by a comma. An example is illustrated in figure 3.1, and the fields are explained in table 3.1 and 3.2.

```
$GPRMC,105435.00,A,6228.32447,N,00614.02623,E,0.145,,230217,,,A*7D
$GPGGA,105435.00,6228.32447,N,00614.02623,E,1,06,1.38,22.5,M,43.4,M,,*6E
```

Figure 3.1: NMEA messages, GPRMC and GPGGA

Table 3.1: \$GPGGA

Field	Name	Example Data	Description
1	Sentence Identifier	\$GPGGA	Global Positioning System Fix Data
2	Time	105435.00	Fix taken at 10:54:35 UTC
3	Latitude	6228.32447,N	Latitude 62 deg 28.32447' N
4	Longitude	00614.02623,E	Longitude 6 deg 1.402623' E
5	Fix Quality:	1	0 = Invalid, 1 = GPS, 2 = DGPS
6	Number of Satellites	06	6 Satellites are in View
7	Horizontal Dilution of Precision	1.38	Relative accuracy of horizontal position
8	Altitude	22.5,M	22.5 Meters above average sea level
9	Height of geoid above WGS84 ellipsoid	43.4,M	43.4 Meters
10	Time since last DGPS update	Null	No last update
11	DGPS reference station id	Null	No station id
12	Checksum	*6E	Used to check for transmission errors

Table 3.2: \$GPRMC

Field	Name	Example Data	Description
0	Sentence Identifier	\$GPRMC	Recommended minimum specific GPS/Transit data
1	Time	15435.00	Fix taken at 10:54:35 UTC
2	Status	A	Status A=active or V=void
3	Latitude	6228.32447,N	Latitude 62 deg 28.32447' N
4	Longitude	00614.02623,E	Longitude 6 deg 1.402623' E
5	Speed over the ground in knots	0.145	Speed over the ground in knots
6	Angle	null(false)	Track angle in degrees(true)
7	Date	230217	Date fix taken 23.02.17
8	Magnetic variation	Null	Magnetic variation in degrees
9	Checksum	*7D	Used to check for transmission errors

3.3 Control device

The group chose to use an app based control system, because devices such as tablets and smart phones are becoming more and more used as control and monitoring units. A tablet or smart phone are also more convenient to carry around. They are also smaller and more compact than for instance a computer.

3.4 GUI

An important part of the platform's control system is to have a user friendly GUI which makes it easy to master the fundamentals of controlling it. The design process started with several simple drawings at the web- based sketching program draw.io. Using such a tool makes it easier to predict what code is needed in the application to make the wanted functionality in the app. The control system should contain a Manual Mode, and Autopilot.

3.4.1 GUI design

Initial designs

The designs, see figures(3.2, 3.3, 3.4 and 3.5), must take to consideration which data to send and receive. The output of the Manual Mode should be data that can easily be transmitted and put to use in the end station, which in this case is the motor controllers on the platform. The buttons and slide bars shown in the GUI designs all send different data values to the server. This trigs the programmed thruster methods, and thus, the platform can move in the wanted direction. The Manual Mode does not need any input data to be functional, but some data feed on GPS data and connection status should be implemented.

The design for the autopilot is based on the Google Map API that is included in Android Studio. Some text fields for monitoring position, thruster power, connection status and heading should be displayed. Also, a table with the waypoints on the map must be implemented.

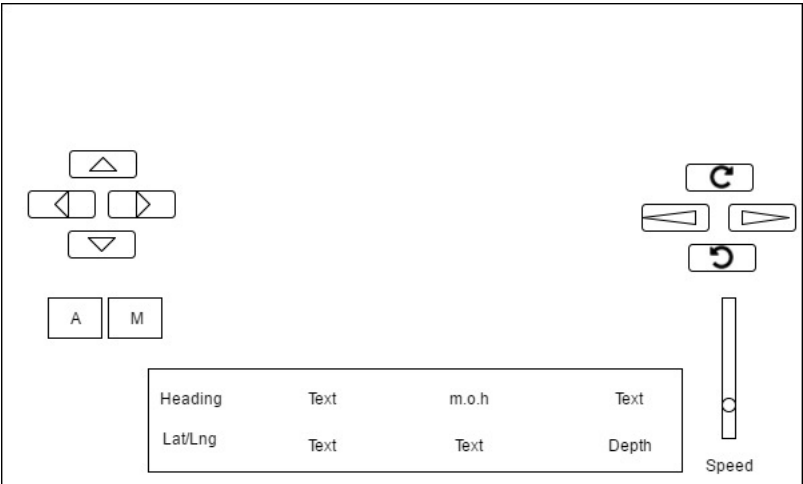


Figure 3.2: Client design 1 Manual Mode

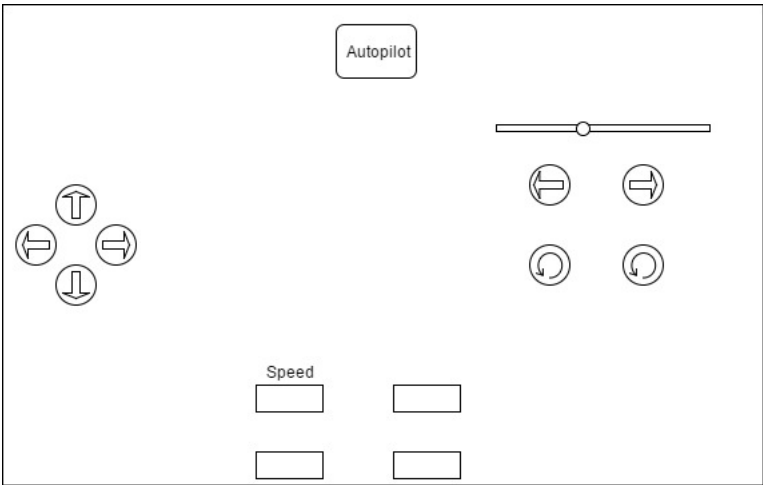


Figure 3.3: Client design 2 Manual Mode

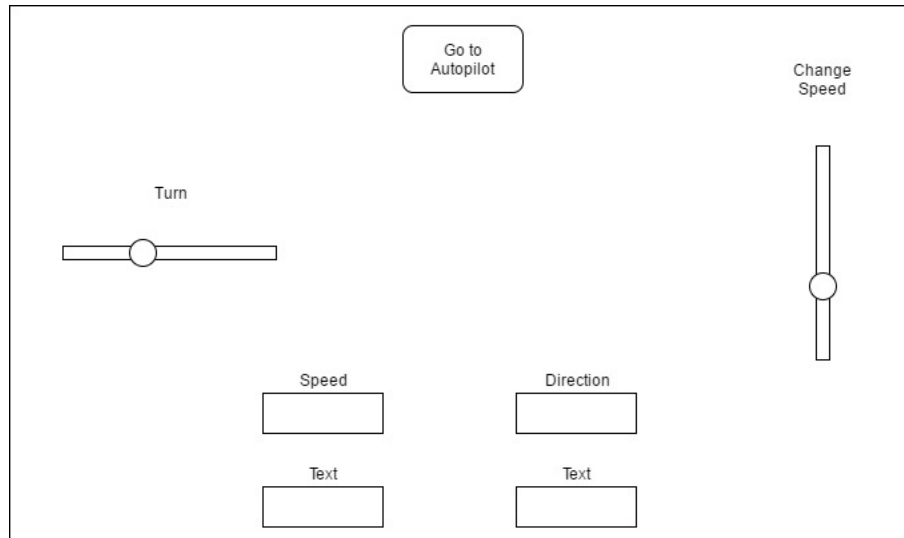


Figure 3.4: Client design 3 Manual Mode

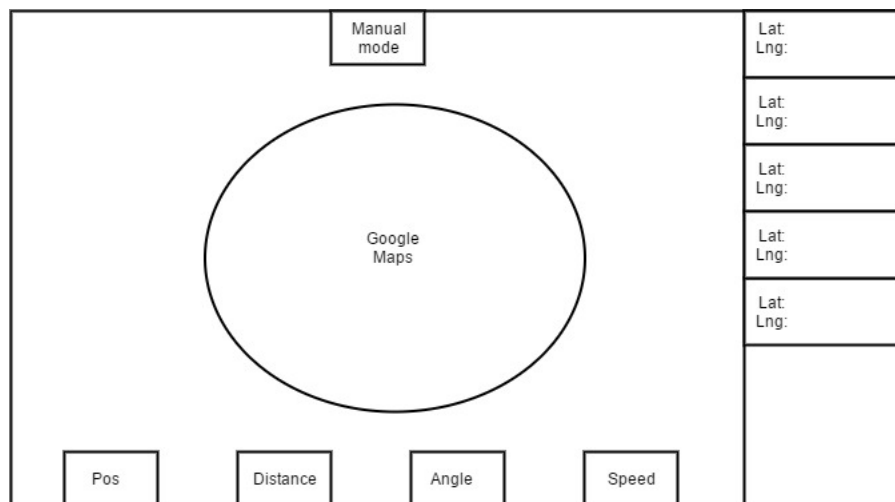


Figure 3.5: Client design 1 Auto Pilot

Manual mode designs has no limitations, and can have whatever design the group sees best. The Autopilot GUI on the other hand, must have the Google Maps API fragment implemented, which narrows down the design options. Therefore, the group has only one draft of the Autopilot GUI.

3.5 Platform design and modelling

This section contains a description of the alternatives considered for designing the platform such as materials and tools for design purpose.

3.5.1 Materials

Platform tubes

The materials used on the platform has a big impact for the behaviour and overall durability. Since the relative design is already made to look like a platform, some sort of pipes had to be used. The cheapest option would be to use PVC plumbing tubes, they also has a low weight and great buoyancy. However, there are some downsides to using these, they are hard to weld (if needed later on), and if left in the cold for too long (for example in seawater) they may crack on impact since they don't withstand the low temperatures very well.

Another option would be aluminium tubes, they are relatively light weight and are very durable in sea. There are some drawbacks to using aluminium as well, if struck or hit by a object at sea, the material may bend and cause leak. Even so, the preferred material is plastic due to its weight and strength ratio compared to metal.

The third option was PE plastic tubes. Polyethylene is both durable and light weight. The material is also often used in marine equipment such as fish farms, due to the material properties of polyethylene. The material withstands the rough conditions very well and low temperatures are not an issue, the weight of PE is 0.96kg/litre and compared to seawater which is about 1.025kg/litre so it even has a slight buoyancy. It is also a more flexible material so if struck or is hit by something, the walls would bounce back to its original form. [30]

Platform frame

Since the structure is made to withstand rough conditions, the frame also has to be stiff enough to hold everything together. One way is to use thick aluminium plates that doesn't bend very easily but the weight of it would be too much.

Another way is to use two thin plates of aluminium with some sort of trusses designed in between for stiffness.

A third option is to use square aluminium tubes and weld them together in the corners, this way

both stiffness and weight would be met in a reasonable way.

A fourth option is to use extruded aluminium profiles, as they are rigid, have a light weight design, and provides possibilities to screw the pieces together instead of welding them.

3.5.2 CAD tools

Since the platform has a somewhat complex design and is relatively large, building and rebuilding the whole thing several times until reaching desired behaviour seemed like a waste of time and resources. Using CAD tools for design was decided at an early stage.

The software used for drawing the platform was Autodesk Fusion 360. This is a cloud-based platform that works on both Mac and PC, the work done is at all times synchronised on the account which prevent any data loss and provides simplicity for changing work location. Assuming there is a computer with Fusion 360 installed, all files are available. There is also an application for mobile devices, both Android and iOS, in which you can display all your work and even do minor changes to the drawing.

The reason for using this software compared to for example Siemens NX or Solidworks, is that none of the participants of the thesis had experience with 3D modelling and Fusion 360 seemed to be the easier one for learning to use.

3.6 Stability and buoyancy calculation

There are several ways to calculate the buoyancy and stability of a vessel. The two main ways to do this is manual calculations or computer aided calculation.

Manual calculation

By using the basic formulas for stability and buoyancy. Buoyancy and stability of a vessel can be calculated with relative ease. The challenge is that any change of weight or angles, forces the calculation of a whole new model. This means that to calculate all the attitudes a vessel can have, requires a large number of calculations. Another issue is calculating the effect of waves, which requires very much calculation and is therefore not a practical solution.

Computer aided calculation

Modern computer programs can make all these calculations, and simulate the vessels movement. Most of these programs are made to calculate for ships, and are not that suitable for calculations of platforms. These programs are very specific, and the models needs to be sketched in the specific program.

3.7 Stability system

One of the features of this platform is its stabilisation system, a system that automatic stabilise the platform to level after adding a payload.

To correct an list angle due to an off centred payload, we need to move weight from the heavy side to the opposite side. Centre of gravity needs to be moved back in the centre of the platform. The most common way is to have a closed loop system were water is pumped from one side to another to correct the offset weight. Because of the limited payload, a closed loop system would only add more weight to the platform. The method chosen is a system which fill and empty the tanks from the surrounding water. This also means that dept adjustment can be made if necessary.

3.7.1 Stabilisation method

Weight, power consumption, speed, controllability, these aspects has to be taken into consideration due to the limited power and payload limit on the platform. Two methods were analysed and tested before any implementation.

Stabilise by compressed air:

An air compressor sends compressed air to an array of solenoid-valves in which each valve distributes air to its designated cylinder. The air inside the chamber will build up pressure and force the water out a hole in the bottom of the cylinder. By opening the valves, the weight of the platform would press the air back out and letting the water flow back inside. An alternative to letting the air back out would be to use a pneumatic vacuum valve to help draw the air back out. A disadvantage of using compressed air to remove water is that the air continues to expand after

the valve is shut, the result could be difficult to control in an exact manner.

Stabilise by water pumps:

Two different systems using water are considered. An enclosed, and an open system. Using an enclosed system would imply moving water between the vertical columns, which removes the possibilities of raising and lowering the draft of the platform. Using an open system however, gives the possibilities of pumping water in- and out of the vertical columns. This would allow the platform to change its draft within the buoyancy limits of the design. An advantage of using water over air, is that a water based balance system would be easier to control because water is incompressible under normal conditions.

3.7.2 Control system

Since the stability system changes the centre of gravity, it can not only stabilise the platform but also destabilise the platform. Therefore it is important that the system is designed as safe as possible. There are two main ways to control a system like this, with a main computer system or a separate system with it's own computing power. The stability system needs to operate even if the main system goes down. A server with many different tasks and that handles communication, has a grater likelihood than a stand alone system to fail. It is vulnerable for errors, and may freeze or shut down. If this happens, the system might give wrong input to the stability system, and the platform could potentially capsize or sink.

3.7.3 Sensors

To know the attitude of the platform, some sort of sensor input is needed. Information needed are pitch, roll, heading and draft. Two different systems were tested for accuracy and reliability. One system with inertial measurement units (IMU) (3.19), and another using pressure sensors (3.20).

water level sensor The attitude can be measured using pressure sensors with one is measuring the outside pressure of each column of the platform. From the height of the water on

one corner compared to another the angle would be calculated. This approach however can not measure the heading of the platform.

IMU sensor The inertial measurement unit has 9 degrees of freedom, measuring all angles as well as the heading.

3.8 Autopilot

The section of autopilot contains the methodology of using GPS for manoeuvring the vessel.

3.8.1 Sensor

For coordinate input to the system, a GPS module is needed. To create an autopilot system, the system needs the current location as a coordinate, the vessel's current heading and a waypoint to navigate to. While the GPS provide both coordinates and heading, the heading is only accurate while the GPS module is moving in a specific direction and speed. The vessel is not intended to move very fast, which means the heading needed has to be more accurate than the GPS can provide at slow speed. As an addition to a GPS, IMUs are often used. An IMU can provide a fairly accurate heading and movement calculation due to its internal magnetometer. Another advantage of using an IMU along with the GPS, is that the IMU can provide a continuous estimation of the vessel's movement, even if the GPS lose connection to the satellites.

3.8.2 Testing the autopilot

For testing the autopilot, some sort of test rig would be preferred, seeing that the platform is meant to move only in water. The rig developed for testing the way-point follower, consisting of an Arduino Uno with a breadboard and several LEDs. The Arduino simulates the Thrust controller, and the LEDs are used as indicators of in which direction the vessel is moving. In front there are four yellow lights that represent forward, backwards, right and left. In the centre, two blue LEDs are located to indicate rotational movement and two red LEDs in the bottom for side-ways movement, see figure 3.6.

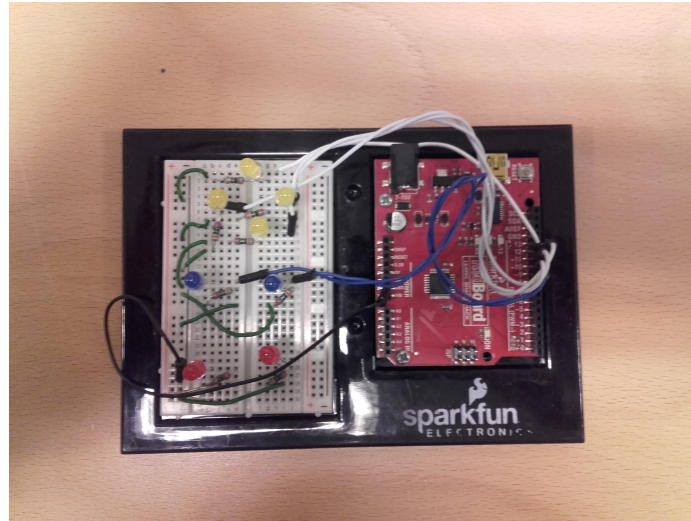


Figure 3.6: Arduino test rig

3.8.3 Collision avoidance

Proximity sensor

Since the platform is designed to be autonomous, it has to recognise objects in its path. There are several methods to do this. A common solution is to use a camera and image processing, along with machine learning. This method is however very complex and time consuming. Another method is to use a proximity laser sensor, that measure distances horizontally in 360 degrees. This method turned out to be the most reasonable one to use on the platform. A proximity sensor called Sweep was ordered to use on the platform. The Sweep sensor is a low-cost device with a 360 degrees horizontal searching area, and can detect obstacles out to 40m. The drawback of this sensor was that it was a Kickstarter project, with release date in mid April. The sensor did not arrive in time for the project, however the idea of how to use it were made.

Object detection

The Sweep sensor comes with an Arduino library which detects objects and outputs its distance and angle relative to the sensor itself. While knowing the objects distance, angle and the sensors sampling rate, one can calculate the objects travelling speed and travelling direction.

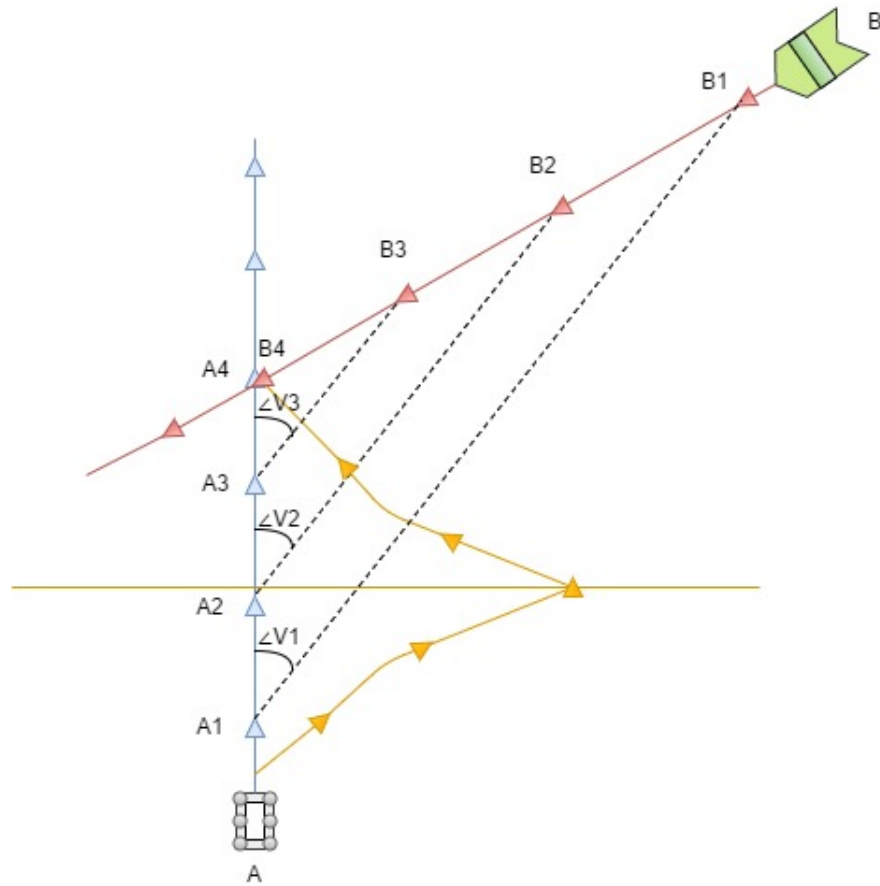


Figure 3.7: Illustration of object detection

Figure 3.7 illustrates the concept of knowing when the platform is on a collision course with an object. The blue line indicates the platform's travelling route and the red line is the other object's travelling route. If the angle ν_1 towards the object is equal to ν_2 towards the object, it indicates a collision course. With knowledge of if and when the two are colliding, the software could automatically set a new waypoint for preventing the incident. The yellow line illustrates the new waypoint for the platform to follow.

3.9 Communication

The choice of communication to the platform is based on several criteria for functionality and range.

- Wireless communication
- Compatibility among equipment

- Range of approximately 200m
- Enough bandwidth to transmit sensor data and camera feed
- Weight
- Robust and Flexible
- Expansion possibilities

To meet the specified criteria, several options to WiFi were considered, such as transceivers that operate in a lower frequency band and implementation of communication over mobile network. Using USB transceivers for a lower frequency band would increase the range, but limit the bandwidth. By utilising the mobile 3G or 4G network one can communicate with the marine craft anywhere where it has mobile coverage, and thereby eliminate the range concern while retain great bandwidth.

3.10 Video stream

A USB web-cam is mounted in front of the vessel. The server captures camera frames, and send them as a stream to the client device using the UDP protocol. In the manual control mode, the video stream will be an operating tool, and function as the "eyes" of the operator due to the manoeuvring difficulties over long distances, taking the desired range of the wireless network into consideration. See the range specification in section [3.9](#)

3.11 Software

3.11.1 NetBeans IDE 8.2

NetBeans IDE 8.2 a free, open source software development environment used to program the java server software running on the Odroid XU4. NetBeans is mainly developed for Java programming, and supports several types of version control systems. NetBeans is compatible with every OS that support JVM.

3.11.2 Android Studio

Android Studio is the official IDE for Android development, and is based on IntelliJ IDEA. Android studio is used to develop the client application running on an Android tablet. Android Studio's project structure and Gradle-based builds provide flexibility when generating APK for all android device types. Android studio is available for Windows, Linux and Mac, and supports JVM.

3.11.3 Arduino IDE

The open source Arduino software is used to program the Arduino/Genuino micro-controllers. By connecting the micro-controllers to a computer with USB, one can upload the software written in a simplified version of the C-language.

3.11.4 inSSIDer Home

inSSIDer Home is a WiFi troubleshooting and optimisation tool that provides a simple graphical view of WiFi information and signal strength of every network within reach. [28]

3.12 Software development

3.12.1 Overview

The programming language used in this project is Java and C. These programming languages are similar in many aspects, and relatively easy to use together. As the project depends on real time data, a necessary feature for the code language is the possibility of concurrent programming, and Java provide this option.

The Java code is written on two different platforms, NetBeans IDE and Android Studio. The reason for this is that the Android Studio gives very useful API's to create the user interface on the client side. In NetBeans, the whole server software is developed. It runs on an Odroid XU4 mini computer. The code written in C, is developed in Arduino IDE. The web-based hosting service Bitbucket and the version control system GIT, were used as platforms for the Java code devel-

opment. The group started the development with making a overview of wanted features using A3 forms, GANT diagram and JIRA Rapid Board. JIRA is a tool that makes it very easy to plan "sprints" of 14 days intervals, where everyone can update and monitor progress of the "to - do" list. Issues to be solved follows:

- Develop code to handle active stabilisation
- Develop code to handle communication between client and server
- Develop code to handle sensor data
- Develop code to handle thruster power

It is crucial for a program of this size that the objects and data are thread safe. To handle this issue, semaphores are used as "permissions" to access the data storage classes, and thus, it can be controlled that only one class has access at a time.

3.12.2 Libraries

The following external libraries are used in the project:

- RXTX
- JOptimizer
- Apache Commons Math
- JSON
- OpenCV

The JSON library is used to create a object that can hold all the values we want to send across- and between the client and server. The methods in this library makes it easy to store and fetch data using "key" words for the different data stored in the object.

To handle the math for thruster allocation, JOptimizer and Apache Commons Math is used.

RXTX is a library for serial communication used to communicate over USB to the Arduinos and the GPS

. OpenCV is used to handle the image capture from the webcam on the server side.

3.13 Materials

3.13.1 Lenovo Tab 2 A10

In this project, the client program for manual control and autopilot runs on a Lenovo Tab 2 A10.

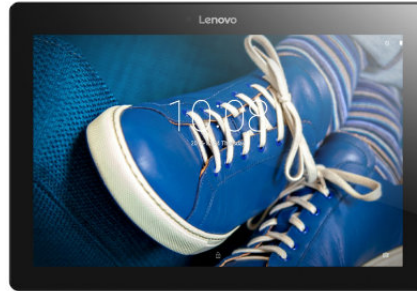


Figure 3.8: Lenovo Tab 2 A10

3.13.2 Odroid XU4

The server side of the control program runs on a Odroid XU4 and is placed on the platform itself. With a Octa Core CPU and 2 Gb of RAM it runs the the multi threaded server program seamlessly.

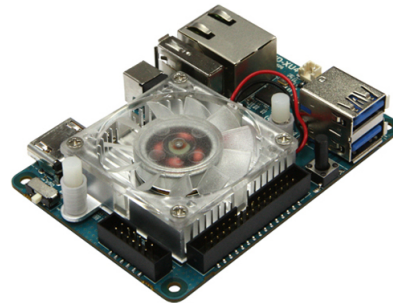


Figure 3.9: Odroids XU4

3.13.3 Linksys TL-WN722N V2

TL-WN722N V2 is a USB network adapter used on the Odroid XU4 to communicate through WiFi with a client. It support data rates up to 150Mbps, and has a transmit power for 2.4GHz is 20dBm, and a reception Sensitivity for 2.4GHz at 11n HT20 is -68 dBm. TL-WN722N V2 is based on the IEEE 802.11n standard. It has a 4dBi antenna connected with an SMA connector which makes it possible to change antenna.

3.13.4 Linksys Archer c5 v2

The router used in this project is a Linksys Archer c5 v2 WiFi router. It is the connection between the client and server for data transmitting. It supports the IEEE 802.11ac/n/a 5GHz and IEEE 802.11b/g/n 2.4GHz standards, and support data rates up to 300Mbps on the 2.4GHz band and 867Mbps at 5GHz band. The transmit power for 2.4GHz is 20dBm, and the reception Sensitivity for 2.4GHz at 11n HT20 is -73 dBm.



Figure 3.10: Linksys Archer c5 v2

3.13.5 Arduino Uno

Arduino Uno is a microcontroller board based on the ATmega328P. It has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, a 16 MHz quartz crystal, a USB connection, a power jack, an ICSP header and a reset button. The platform has two of the Arduino Uno, one for controlling the thrusters, and one for controlling the lidar stabilizing gimbal.

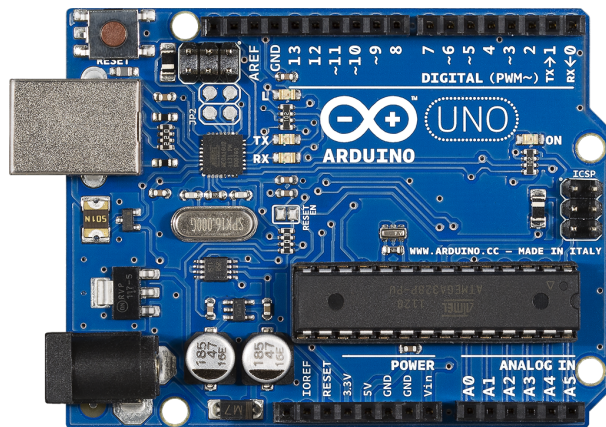


Figure 3.11: Arduino Uno

3.13.6 Arduino Mega

The Arduino Mega is a microcontroller board based on the ATmega2560. It has 54 digital input/output pins (of which 15 can be used as PWM outputs), 16 analog inputs, 4 UARTs (hardware serial ports), a 16 MHz crystal oscillator, a USB connection, a power jack, an ICSP header, and a reset button. Two Arduino Megs are used, one for reading IMUs and the other to control the thrusters.

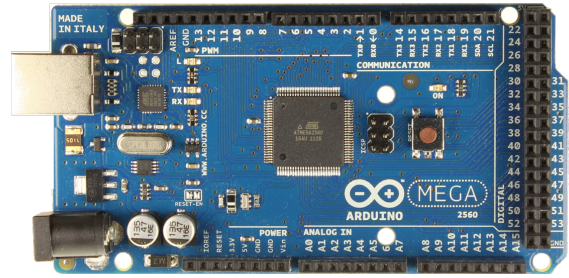


Figure 3.12: Arduino Mega

3.13.7 Haswing 20

The Haswing 20 is an electrical outboard motor rated for 9kg of thrust and consume 17Amp at max speed. The thruster motors are cut off its shaft, and new mounts are made for them specifically for the platform. The thrusters are very cheap compared to the amount of thrust they produce and the fact that they already are waterproof. An alternative thruster considered was the BlueRobotics T200 thruster, but they are much more expensive and produce a lot less thrust.



Figure 3.13: Haswing 20

3.13.8 Bilge pump

A bilge pump rated at 32 l/min at 12 volts and consume 2.5A. Weighs approx. 300g each. The platform has eight of these for the active stabilisation system.



Figure 3.14: Biltrema bilge pump

3.13.9 Relay module

Sainsmart 4-channel relay module, each channel rated for 10A. Two modules are used for controlling the eight bilge pumps in the stabilisation system. The modules are digitally controlled by one of the Arduino Mega boards.

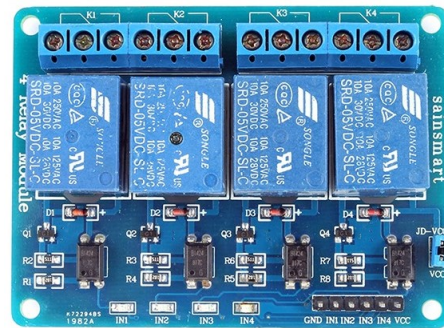


Figure 3.15: Sainsmart 4-channel relay module

3.13.10 Pololu motor controller

The Pololu Simple Motor Controllers are versatile, general-purpose motor controllers for brushed, DC motors. They have a wide operating range of up to 5.5-40V and the ability to deliver up to several hundred Watts in a small form factor make these controllers suitable for many motor control applications. The platform use these for controlling its thrusters, they communicate via serial through an Arduino Uno and give the motors 3200 steps of speed in each direction.

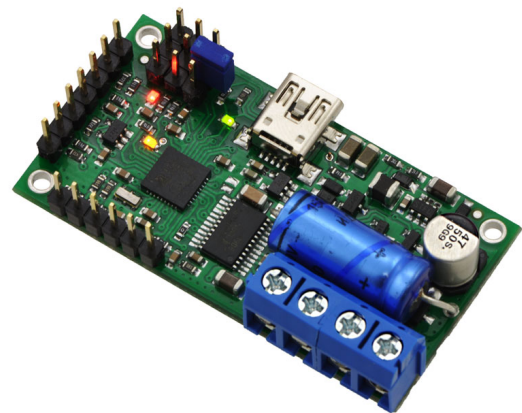


Figure 3.16: Pololu simple motor controller

3.13.11 CP1232 Battery

For powering the platform, it has two battery packs with five CP1232 batteries in each. The batteries are 12V and are rated for 3.5Ah each, giving the total of all ten batteries 35Ah. They are rechargeable batteries and are lead-lead dioxide systems. The dilute sulfuric acid electrolyte is absorbed by separators and thus immobilized. Should the battery be accidentally overcharged producing hydrogen and oxygen, special one-way valves allow the gases to escape thus avoiding excessive pressure build-up. Otherwise, the battery is completely sealed and is, therefore, maintenance-free, leak proof and usable in any position. [4]



Figure 3.17: CP1232 Battery

3.13.12 Multiplexer

Even though the number of devices you can connect to a bus is very large, (for example 127 for 7-bit), it is still limited by the fact that none of the devices can have the same address. The TCA9548A multiplexer is needed for the platform to read all four of its IMU's at the same time since they all have the same address. The TCA9548A device has eight bidirectional translating switches that can be controlled through the I^2C bus. The SCL/SDA upstream pair fans out to eight downstream pairs, or channels. Any individual SCn/SDn channel or combination of channels can be selected, determined by the contents of the programmable control register. [35]

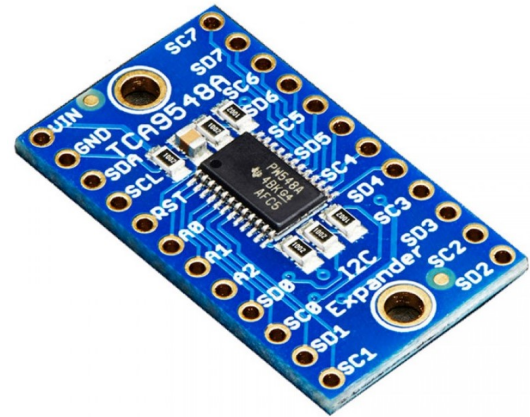


Figure 3.18: TCA9548A

3.13.13 9DoF IMU

For navigation and stabilisation functionality, an IMU (Inertial Measurement Unit) is needed. The platform is installed with four IMU's, one on each corner, for enhanced accuracy. The sensors has a 3-axis accelerometer, 3-axis gyrometer and a 3-axis magnetometer, which defines 9 degrees of freedom. One of the platforms sensors are a 10DoF, this has barometric/temperature sensor as well as the other three. This functionality is not used on the platform. [35]



Figure 3.19: Adafruit 10-DoF

3.13.14 Pressure sensor

To measure water level the MPX2010dp is used. The sensor is a piezoresistive pressure sensor. It is self compensated between 0 and 85 degrees Centigrade. It has good accuracy and have a linear voltage output proportional to the applied pressure. [33]



Figure 3.20: MPX2010DP

3.13.15 Check valve

For stopping water of flowing the wrong direction in the stabilisation system, each pump is installed with a check-valve. The valves are from Bilterma, a low-cost solution for the issue.



Figure 3.21: Bilterma check valve

3.13.16 GPS module

The Ublock vk-162 is used to as the GPS receiver on the vessel. It has an update frequency of 10Hz, and supports DGPS(WAAS, EGNOS and MSAS). The average accuracy in the 2D plane of 3.5m, and average startup time is 32 second for both warm and cold start, and 1 second for hot-start. The VK-162 supports the following operating systems, Windows, Android and linux.



Figure 3.22: ublock vk-162

Chapter 4

Result

4.1 Designing the platform

This section presents a walk through of designing and building the prototype as well as decisions made for choosing methods and equipment.

During the design part of the project, a focus was to keep the material cost as low as possible and the durability of the structure as high as possible. These two arguments counteracts each other in some level so they had to meet half way. The durability of the platform is emphasised first of all due to the rough conditions at sea but also for withstanding transportation and handling.

During the design and modelling, the latest model would continuously be tested for its weight, volume and centre of mass through a mathematical calculation of stability and buoyancy.

As seen in the models, polyethylene tubes were decided to use from the very beginning. The material properties of PE for the tubes and aluminium for the frame was used in the calculations for each model.

The following subsections displays all the models that were drawn.

4.1.1 Four legged version

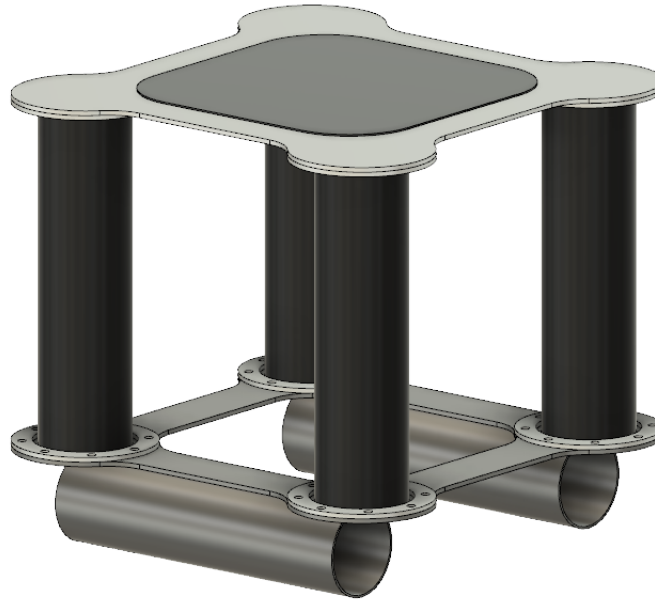


Figure 4.1: Four legged version

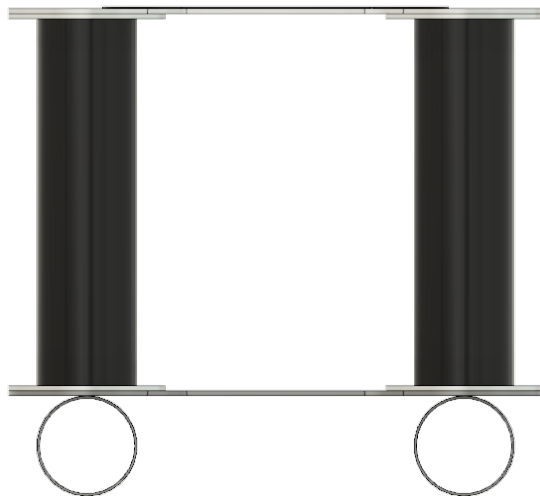


Figure 4.2: Four leg Bow

The four leg idea, figure 4.1 and 4.2, was brought up early on because of its simple design, based on the original "oil rig" look. The vertical pipes would produce the needed buoyancy, and the bottom horizontal pipes would contain battery packages. The batteries would be placed in the bottom pontoons to increase the stability by lowering the centre of gravity. The model is a working concept, but the pipes has to be scaled up to reach the payload and stability criteria.

4.1.2 Hexagon model 1

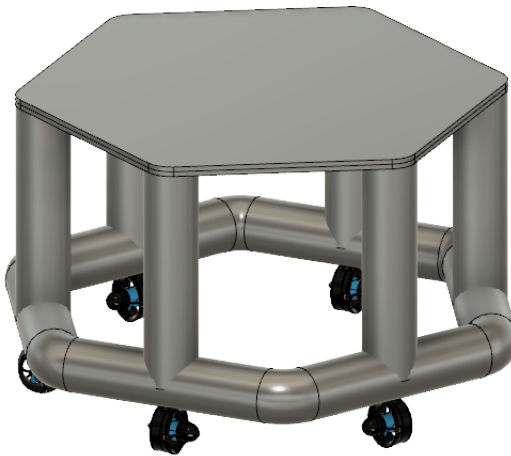


Figure 4.3: Hexagon model 1

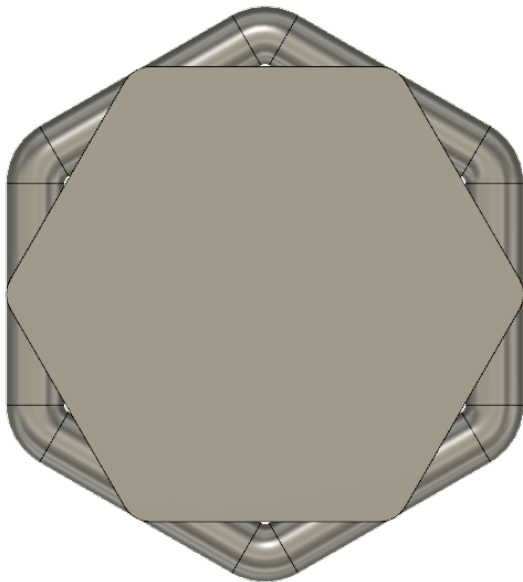


Figure 4.4: Hexagon model 1 Top

To increase the stability and buoyancy from the previous design, the six legged version was created. Figure 4.4 and 4.6. With six legs, buoyancy and stability were increased. The reason for the hexagon shape is a theory of having the longest possible distance from centre platform and out to the pontoon at all times, making it as difficult as possible for it to capsize.

4.1.3 Hexagon model 2

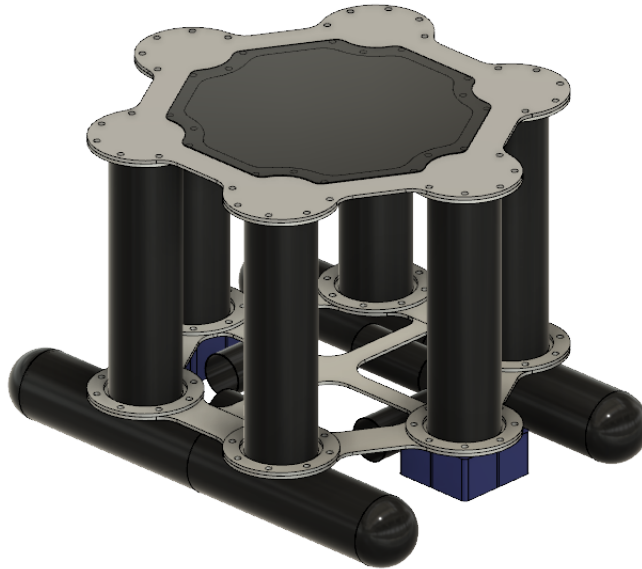
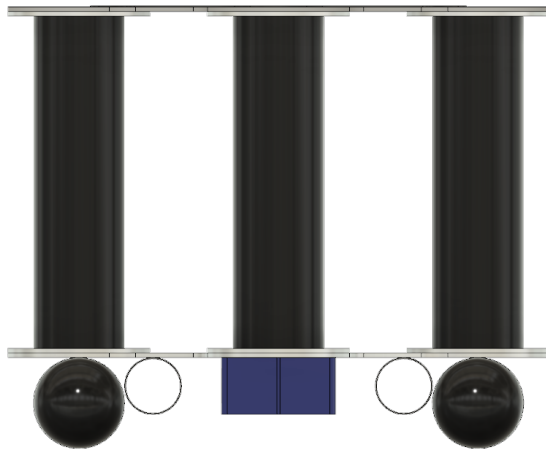


Figure 4.5: Hexagon model 2



A more detailed version of the Hexagon model was made. See figure 4.5 and 4.7. The vertical columns are 60cm tall in this version.

Figure 4.6: Hexagon model 2 Bow

4.1.4 Hexagon model 3

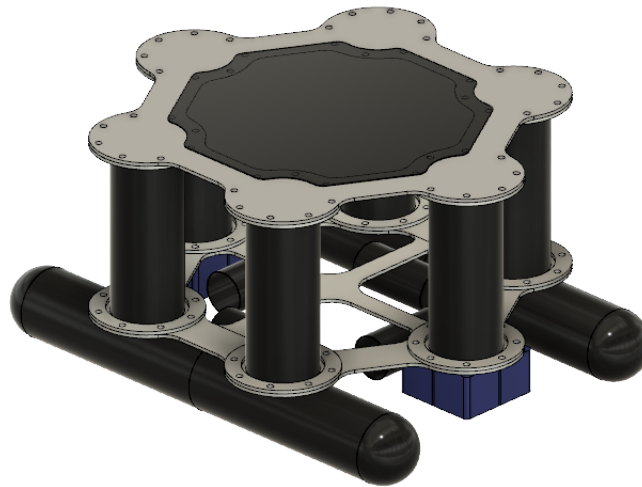


Figure 4.7: Hexagon model 3

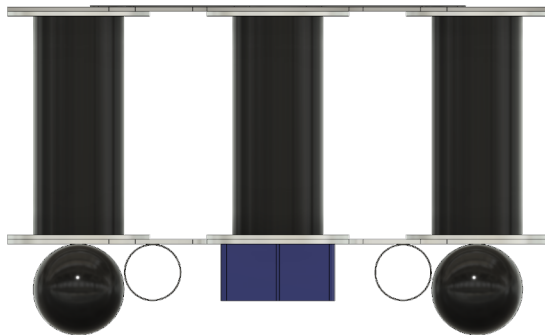


Figure 4.8: Hexagon model 3 Bow

A shorter version of the hexagon model. See figure 4.8 and 4.9. While the buoyancy was decreased, the stability would increase with a lower center of mass. The vertical columns are 40cm tall in this version.

4.1.5 Hexagon model 4

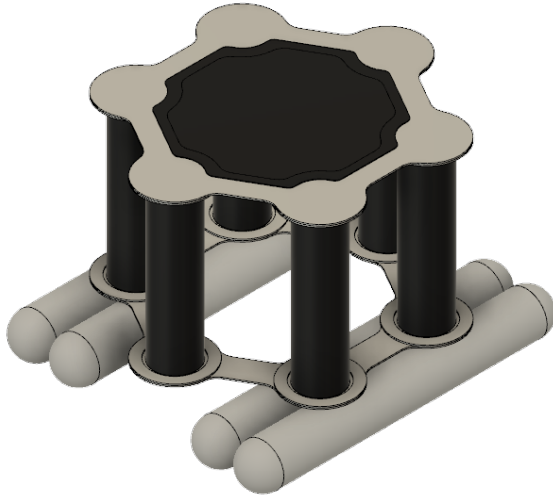


Figure 4.9: Hexagon model 4

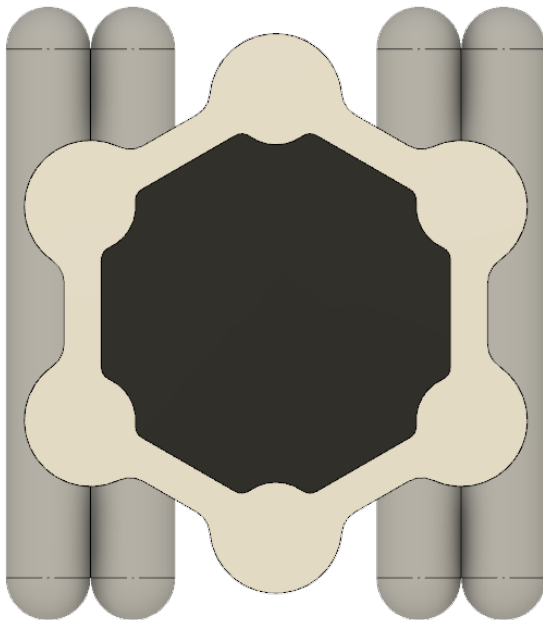


Figure 4.10: Hexagon model 4 Top

While running calculations of stability, another idea was to have two "modes" for it to run in. One called "transport mode" where the whole platform would rise above the water surface, floating only on the four bottom pontoons. This would increase the GM, giving the platform to be more seaworthy and behave similar to a catamaran. In the other mode, called "sensor mode". The platform would lower itself to where the waterline would meet the middle of the vertical columns. This would increase its stability, which would result in better accuracy when using sonar or other equipment. Calculating the stability and buoyancy for transport mode turned out to be somewhat difficult reason being the horizontal pipes in the water surface due to its cylindrical shape. See figure 4.9 and 4.10.

4.1.6 Octagon model

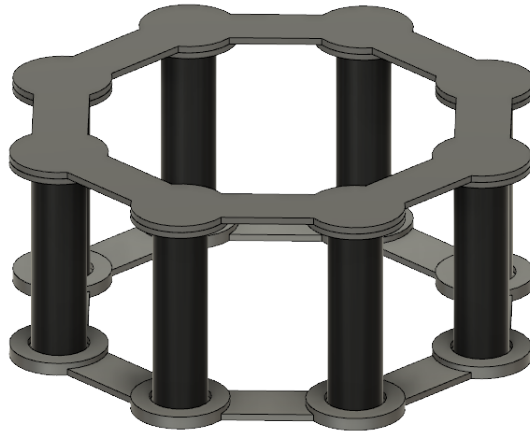


Figure 4.11: Octagon model

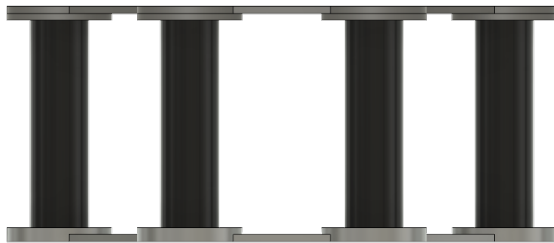


Figure 4.12: Octagon model Bow

An octagon shaped model were started but left aside early on. Figure 4.11 and 4.12. There were a couple of reasons for this, while the size of the platform was an issue and the number of columns were increased, they all joined in on creating a bigger surface area for waves to hit in the ocean. Secondly, the complexity of the model increased with the number of columns, which was unwanted. Even though the models were fine and seemed stable in the calculations, all of the hexagon and octagon models were eventually discarded due to the poor hydrodynamics.

4.1.7 Catamaran model

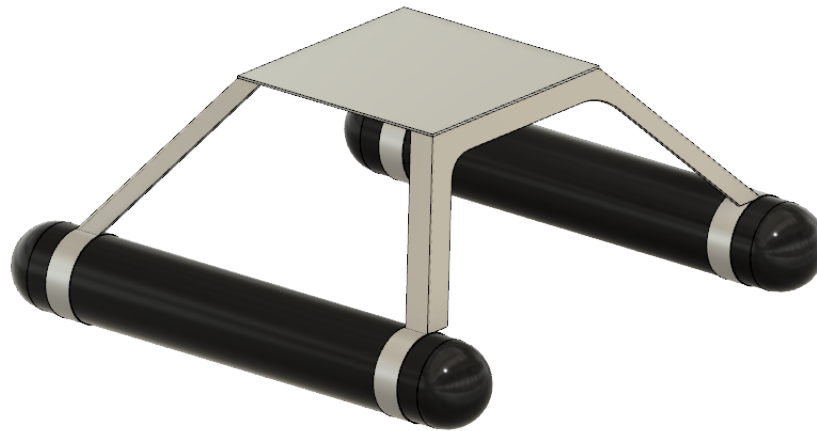


Figure 4.13: Catamaran model

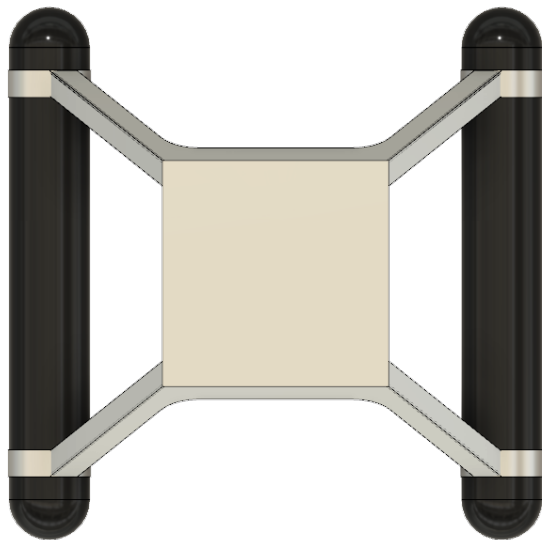


Figure 4.14: Catamaran model Top

A catamaran inspired model was started as a light weight alternative for the platform. See figure 4.13 and ???. This model could not be semi-submerged, but had to float on the surface like a regular boat. The idea was left aside because it didn't meet the requirements of the platform's stability and payload.

4.1.8 Rectangular model 1

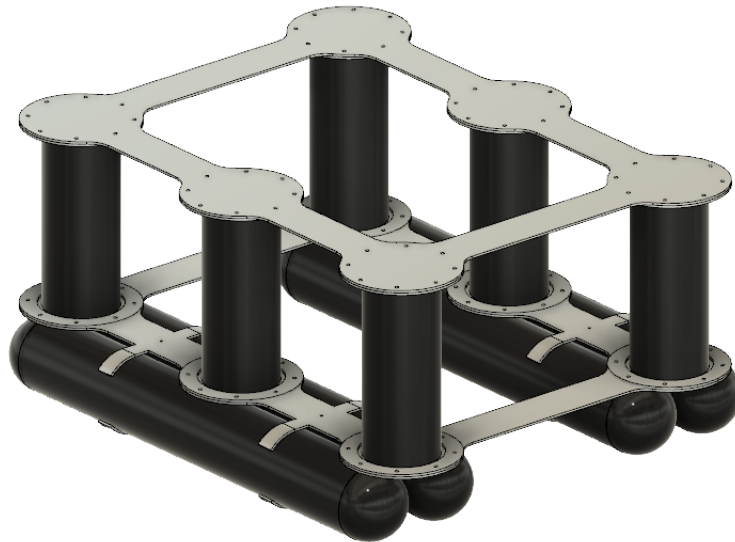


Figure 4.15: Rectangular model 1

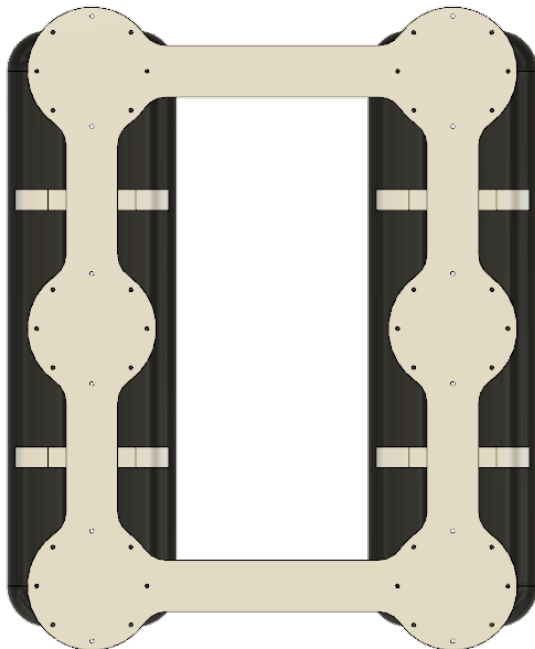


Figure 4.16: Rectangular model 1 Top

This is a rectangular model, see figure 4.15 and ??, with the same theoretical buoyancy of the hexagon model but slightly increased stability due to it being longer. The model was designed in parallel with the hexagon model, therefore the concept of having two operating states were still implemented.

4.1.9 Simulation-sketches

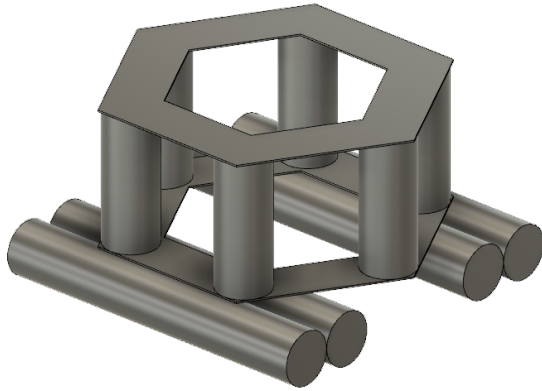


Figure 4.17: Simple hexagon

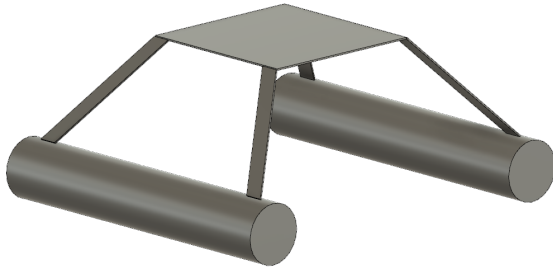


Figure 4.18: Simple catamaran

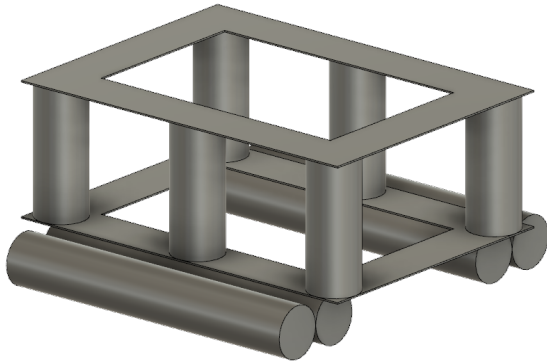


Figure 4.19: Simple rectangular

As an interesting approach, the models were to be tested in a physics simulator to visualise the behaviour and not only the calculations. For simulation purpose, the models doesn't have to be exact, the simulator only need the physical properties of the model such as centre of mass, buoyancy and weight, everything else is calculated by the physics engine. Since the simulator only need these parameters, the design is less important for the simulation. A simple design made it easier for the visualisation software to run the simulation. Se figure 4.17, 4.18 and 4.19.

4.2 Data collection and calculation

During the design stage, a minimum payload of 15 kg was set to handle a winch and a ROV. Several design ideas was made for researching their stability and how they are affected by the platform geometry.

Three main concepts were chosen, a Catamaran version, a Hexagon shaped version and a rectangular version. The Catamaran was discarded due to low payload, and that it was heavily

affected by waves. To find the stability of the two remaining models, some testing was done using different computer programs. A lesson learned is that these programs need a skilled user to produce a satisfactory result. The decision was made to use the manual calculation method, since this provided better control over the result. In consultation with professor Henrique M. Gaspar, an excel spreadsheet was made to calculate the stability and buoyancy.

The heading of the spreadsheet contain information of the platform, such as the number of columns and dimensions. It also need specified water density, and the weight of the plastic pipes that is used.

While designing the platform and choosing materials, the weight and volume of each components were known at all times. Some of the small parts like nuts, bolts and brackets were only estimated, and added as a constant to the calculations. For simplifying the model, the weight measurements for each part are combined in one component. For instance, the weight of one column is considered with nuts, bolts and support rods, and then all the main components were added together in the spreadsheet. One component is added as a point object, which means that you add the centre of gravity of the object in the x, y and z axis, and then specify the weight. From this one can calculate the centre of gravity of the platform as a whole.

The spreadsheet has a section were the volume of each of the main components are listed, and the buoyancy is calculated. This calculation uses the weight of the platform and payload to calculate the draft of the platform, it also separate between watertight and volumes that can be filled with water. The draft are given in both meters and percent.

Next is the calculation of the moment of inertia. Here Steiner's theorem (2.1.4) is used. The calculation is done in three stages. First the inertia of one column is calculated, then the transferred inertia is calculated. And then the two are combined. This is done for longitudinal inertia and transverse inertia.

In the section "Stability semi submerged" KB, KG, BM and GM (2.1.3) is calculated. To visualise the stability, a calculation of GZ for different angels is made. For a semi-submerge platform, the GZ is nearly linear for angles up to 17 degrees. After that this method of calculation is not precise enough. Because of that we do not have GZ for angles over 17 degrees. The GZ values are showed in a diagram, here the differences in stability between the longitudinal- and transverse stability are easy to see. The stability calculation only take in to account, movement in the z-axis

of CG. Therefore there is a last section for calculating the angle of list.

With the help of this spreadsheet, the two remaining platform designs were tested and both designs were stable, and had good buoyancy. The Hexagon shaped platform had equal stability in any direction. But the rectangular platform has better stability transverse than longitudinal. This because it has three columns that adds buoyancy and thus inertia in the transverse, and only two longitudinal. This was unexpected, we believed that the longer moment arm of the longitudinal columns would compensate for this. Tests in the wave tank confirmed the calculations.

Since the stability of both the hexagon and rectangular was satisfying, the decision of choosing one design over another, was not made on the basis of stability or buoyancy.

4.3 Choosing design

To make a correct decision about design. Pros and cons are listed in the table [4.1](#)

Design	Advantage	Disadvantage
Four legged version	Simple design	Pipes has to be scaled up to a point where the model is larger then the other models to provide the same buoyancy. It is preferred to increase the number of columns instead of the diameter of them.
Hexagon and Octagon versions	The design provides extra stability by having several arms reaching out in more than four directions.	While the design provides extra stability, the arms also prevents some waterflow in the front and rear part of the platform. Also, the design is found to be more complex than necessary.
Catamaran version	Lightweight design, for a small scale use this could be a plausible design	The model would not be semi-submersible which means, the waves would influence the vessel more than the other models.
Rectangular version	The model provides both buoyancy and stability. It has a natural front and rear, considering the placement of the vertical columns. Since the model has six legs, the pipes doesn't have to be as big as in the four legged one. Only two columns are visible in front and rear, so the water would easily flow through the vessel.	None significant.

Table 4.1: Model comparison

From the result of the table, the decision was made to continue with the rectangular shape.

4.4 Final design

With the general design locked, the group started on detail drawing and specifications of the final prototype. A detailed CAD drawing was made to reveal as many mechanical difficulties as possible, as well as having the model on the drawing board. The method made the actual build of the platform itself an easy job with very little difficulties.

4.4.1 Rectangular model 2

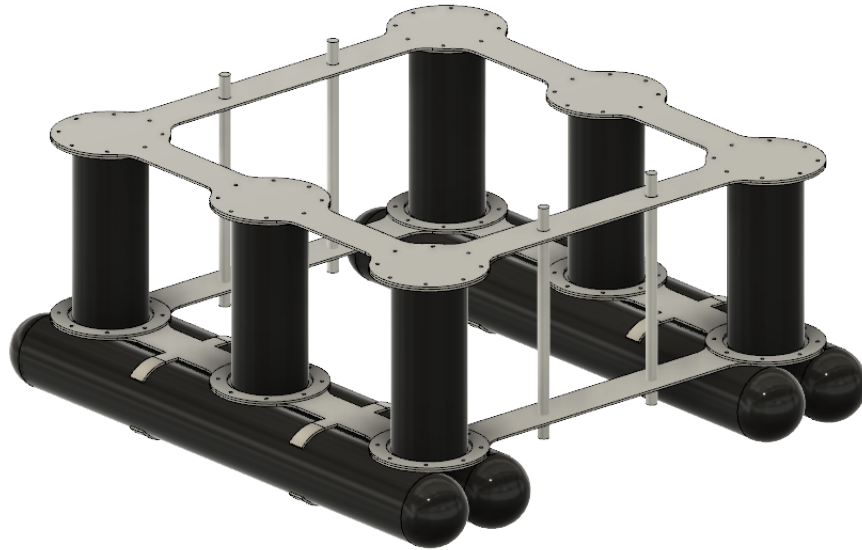


Figure 4.20: Rectangular model 2

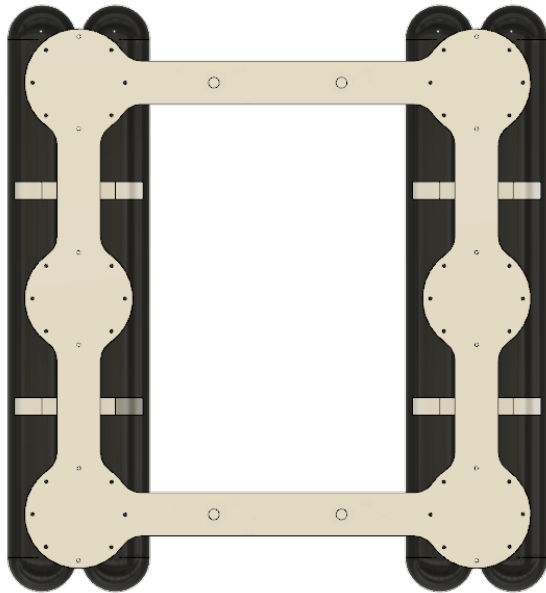


Figure 4.21: Rectangular model 2 Top

While sticking to the design of having a rectangular model, the question of where and how to place the thrusters was considered. In this model the idea was to have the thrusters in front and rear, mounted to the end of the rods. Doing this, the thrusters could rotate 360 degrees individually like an Azimuth. See figure [4.20](#) and [4.21](#).

4.4.2 Rectangular model 3

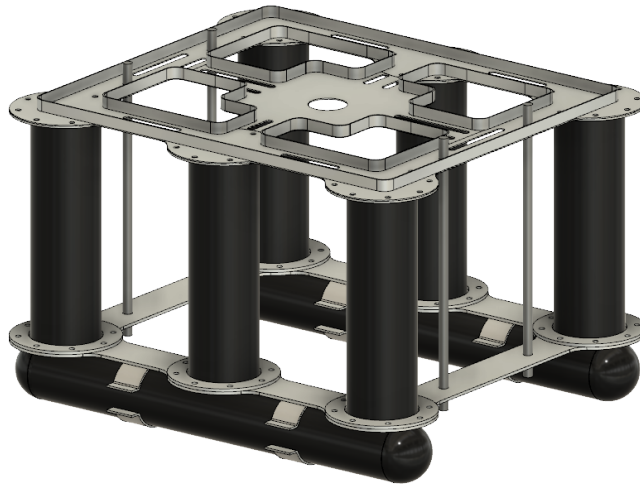


Figure 4.22: Rectangular model 3

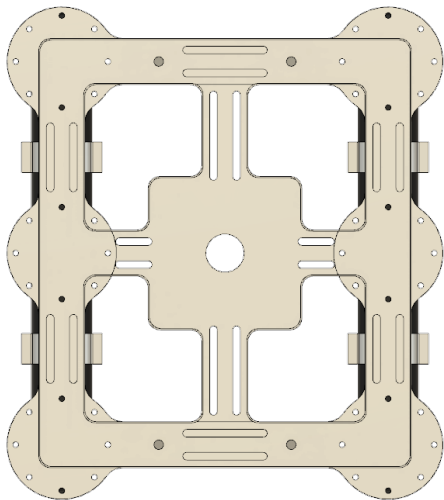


Figure 4.23: Rectangular model 3 Top

While working on the rectangular model, the buoyancy and stability was found to be OK, however the structural integrity of the platform was not yet very good. Using only thin sheets of aluminium, the platform wouldn't be as rigid as necessary for the environment. This model, see figure 4.22 and ??, was created for the purpose of stiffening the top section, aluminium plate rods would be bent and welded to the top frame. While the design would definitely do its job, it was a little overkill and making it only more complex than necessary. Another problem with this design, was that while the thrusters were turned 90 degrees and pointed straight into the bottom pontoons, the thrust would cancel out by hitting the platform itself. Along with the thrust problem, the concept would be too complicated to implement on this project within the given deadline.

4.4.3 Rectangular model 4

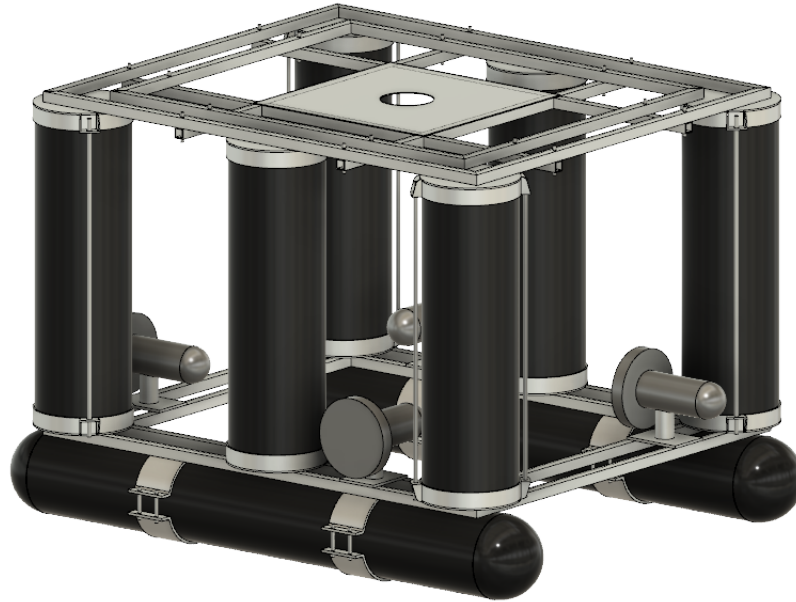


Figure 4.24: Rectangular model 4

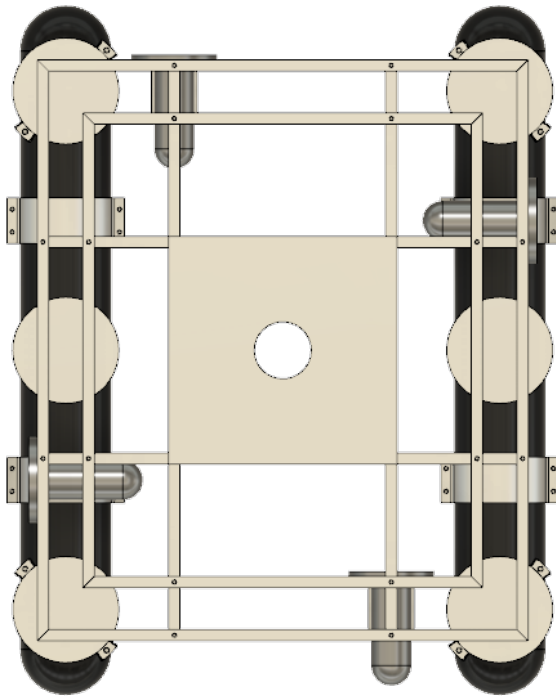


Figure 4.25: Rectangular model 4 Top

The fourth and final design, see figure 4.24 and ??, where thruster mounts and location were changed. The structural integrity of the platform were maintained by using aluminium profiles instead of a complex design. The vertical columns are mounted with end caps and the whole platform is held together by threaded rods. The thrusters are located in way that keeps the platform directional, which means it has bow, stern, starboard and port sides. Even though it has a naturally "forward" manoeuvre, it can move in any direction.

4.4.4 Weighing of parts

All the parts of the main structure of the platform was weighed (fig4.26) before assembly. Some changes were made while ordering parts and building, which caused the weight to deviate from the table below 4.2.



Figure 4.26: Weighing of parts

Bottom frame, with battery pontoons, end caps and stainless rods, W/O batteries	24.2 Kg
Top frame	8.8 Kg
One vertical pontoon with four stainless rods with nuts and washers	5.14 Kg
Six vertical pontoons with 24 stainless rods with nuts and washers	30.84 Kg
One thruster	1.25 Kg
Four thrusters	5 Kg
Two packs of five batteries	16.784 Kg
Eight water pumps	2.4 Kg
Two boxes for electrical installation	4.3 Kg
Total	92.324 Kg

Table 4.2: Overview



Figure 4.27: Battery packs are weighed

Table 4.2 shows the weight of the individual parts of the platform, this is however only the main parts of the platform. Cables, valves, electrical components and lights were not weighed before mounting.

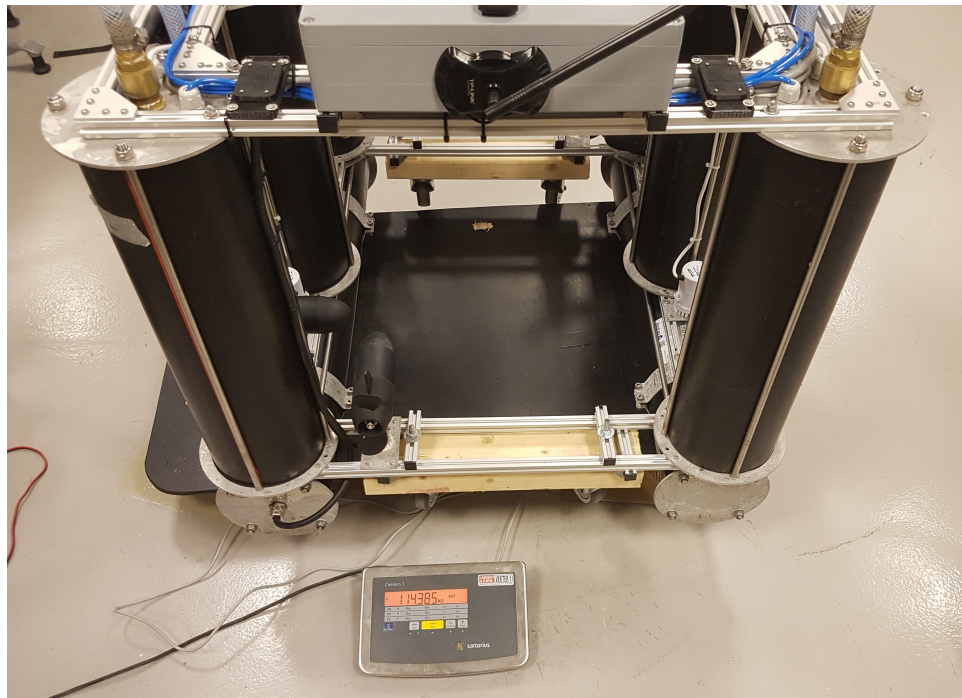


Figure 4.28: Finished platform weighed

After completing the prototype, the platform was weighed and found to be at total 114.385 Kg. See figure 4.28

4.4.5 Buoyancy test in water

The platform was tested in segments, the first test was just after completing the general build, only the pipes, frame and thrusters was mounted and the battery pontoons were open in both ends. See figure 4.29.

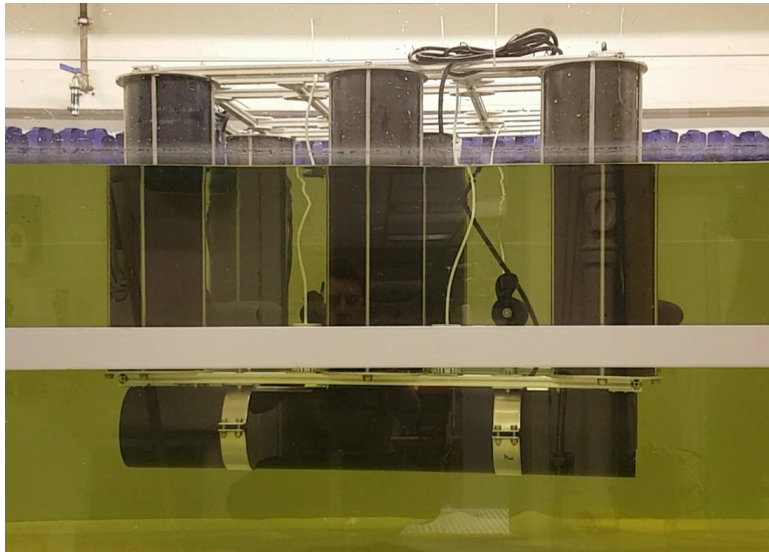


Figure 4.29: Finished platform weighed

Even though there were no equipment mounted, the platform's draft is little low. The reason for this is the open battery pontoons. Without closing the pipes, there is far less buoyancy in them. The total weight of the platform at this stage is 60 Kg.



Figure 4.30: Finished platform weighed

In the second test, the battery pontoons were enclosed, and had the batterypacks inside. See figure 4.30. The result is that the platform got an increased buoyancy, causing a higher draft. The platform had at this stage a total weight of 86,824 Kg.

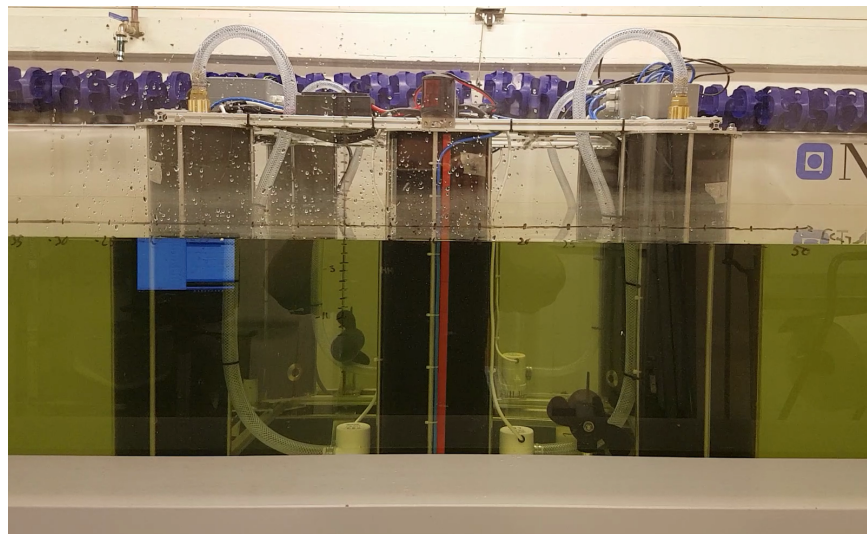


Figure 4.31: Finished platform weighed

In the third test, the platform was complete with all the equipment mounted. The weight of the platform was measured to 114Kg, and the draft of the platform has 4.31, 114 Kg, plus some water for stabilising the platform to level.

4.5 Stabilisation system

This section presents the analysis made for all the methods considered using in the stabilisation system.

4.5.1 Water vs air as control medium

Control by air

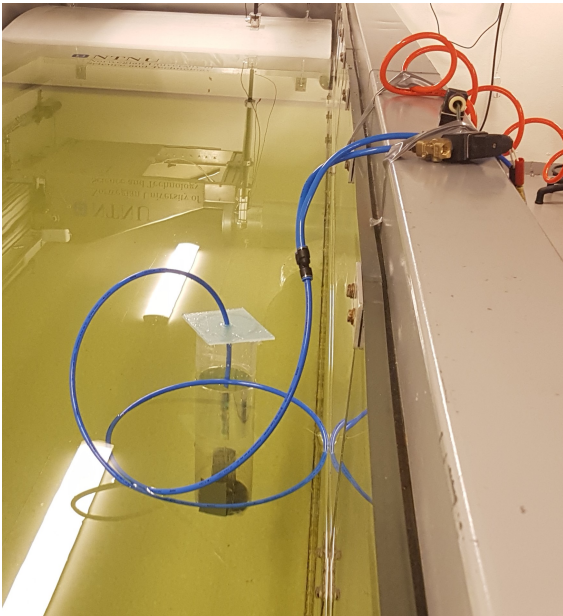


Figure 4.32: Test with air

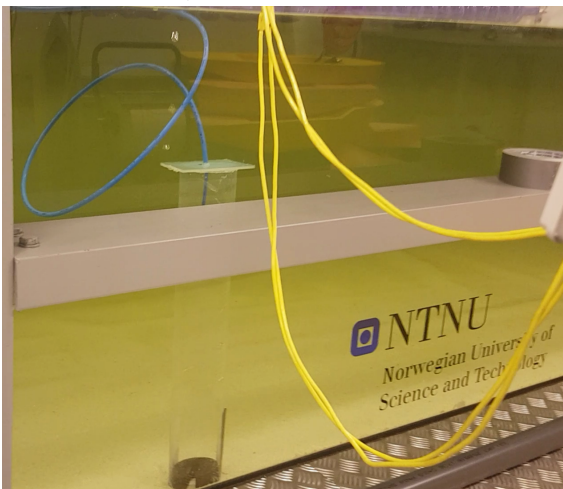


Figure 4.33: Test with air

The first method of control medium tested, was air. An acrylic tube with a heavy load at the bottom end, pneumatic hose for the air, pneumatic solenoid valve for controlling on and off, compressed air, and a power supply for the solenoid valve is mounted and used as a test rig. The first image [4.32](#) shows how the tube standing vertically in the water due to the payload at the bottom.

Using compressed air for moving ballast, the tank doesn't have to stay above the surface for being able to rise. See figure [4.33](#)

Control by water pumps

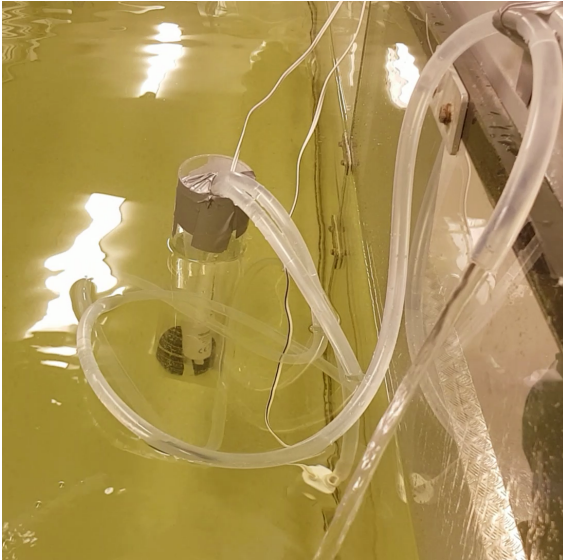


Figure 4.34: Test with water pumps

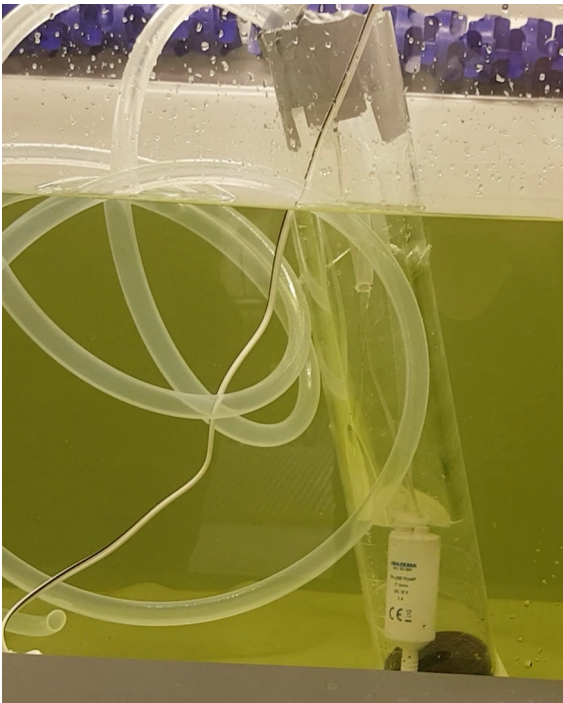


Figure 4.35: Test with water pumps

To control the water with pumps, a two-directional pump or just using one for each direction is required. For this test there were used one pump for filling and another one for emptying the cylinder. At first, the results from the test seemed fairly close to the one using compressed air. Both managed to rise and lower the cylinder with ease. See figure [4.34](#) and [4.35](#).

While the method of using compressed air was versatile and had the possibility of rising the cylinder from beneath the surface, this method has no possibility for doing such thing. If the cylinder were to fill itself and sink below the surface, there is no way to get it back up. If the method were to be implemented to the platform, some sort of fail-safe had to be implemented so if the stability system failed, the platform wouldn't sink. After analysing both methods, the data was collected and presented to the following table.[4.3](#)

Water	Air
Power consumption	
Bilge pump with a capacity of 32 l/min 2.5 Amp at 12 Volts Total: 2.5 Amp at 12 Volt	Air compressor with capacity of 12 l/min 6 Amp at 12 Volts Solenoid air valves 1 Amp at 12 Volts Since the air is only pumped into the cylinders and not out, we can halve the consumption of the air compressor. Total: 4 Amp at 12 Volt
Weight	
Each bilge pump has a weight of 300g and for the total system it's needed eight. $8 \times 300g = 2400g$ Water hoses and the actual water in them adds to this. Total: 2400g	The air compressor weighs 2500g and each of the solenoid valves weighs 300g For the total system it's needed eight solenoid valves. $8 \times 300g = 2400g$ $2400g + 2500g = 4900g$ Air hoses add to this weight. Total: 4900g
Controlling ability	
Water is not compressible, easier to calculate actual weight of each cylinder. Since the water is pumped both in and out, the amount of water pumped is equal both in and out per time unit.	Air is compressible, making it more challenging to regulate the amount of lift in the cylinders. For lowering the cylinders, the weight of the platform is the only thing pressing the air out of the cylinder, which means the amount of air in and out is not equal per time unit.
Speed	
At 32 l/min the water pump speed is sufficient for the application	Compressed air is very fast but only as long as there is air inside the reservoir, the compressor has to keep up with the air consumption and to match 32 l/min the weight of the equipment increases.

Table 4.3: Stability system, water and air comparison

At first glance, the two methods preformed equally. The comparison of the methods indicates that using water pumps is simpler, uses less power and the components of water pumps weigh less than the components of using the air method.

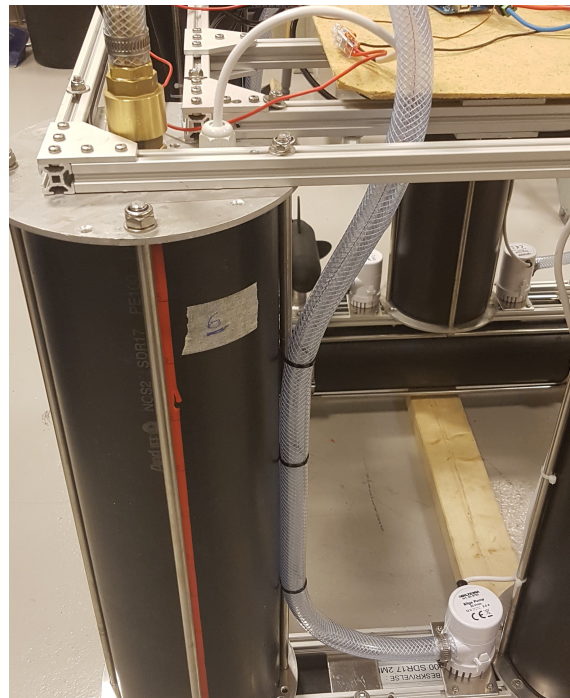
4.6 Mounting the stability system

4.6.1 Water pumps and hardware

Each pump is installed with a check valve to prevent water to flow in the wrong direction. Without these valves, the chambers would be filled with water by gravity alone. The outside check-valves prevent water to run back down the hose while the pumps are on standby, this way the the pump doesn't have to refill the hose on every start-up. The inside valves keep the water from flowing back in the cylinder through the pump. See figure 4.36



(a) Inside pump with direct outlet (seen from below)



(b) Outside pump with hose going through top

Figure 4.36: Placement of the water pumps

Since the cylinders are air tight, the water pumps would create either pressure or vacuum depending on which way the water flows. The pressure would eventually build up to where the

water pumps stall and the pressure sensors would react to the change in pressure. Therefore a ventilation hole was necessary. The hole would have to be big enough to match the specification of the pumps (32 l/min), in such way that pressure nor vacuum would influence the sensors.

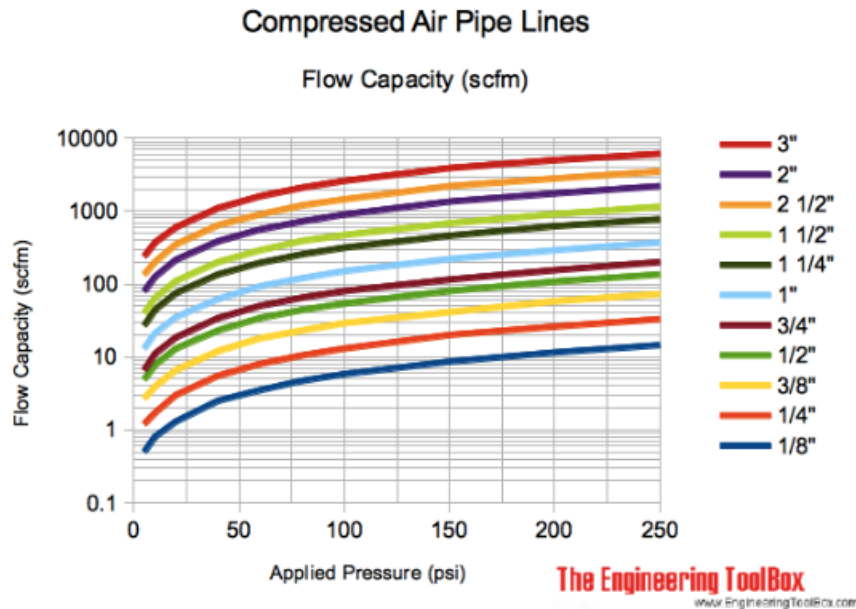


Figure 4.37: Flow capacity to pressure table

The graph in figure 4.37 indicates needed dimensions for given pressure and air flow. One standard cubic feet of gas per minute equals 28.32 litres per minute, so in this case where the pumps capacity is 32 l/min it's a little over one (scfm), and the air pressure inside the cylinder is ideally 1 bar, or equal to atmospheric pressure. Considering this information, the value read from the table is 1/4 inch hole is sufficient to prevent unwanted pressure change. 1/4 inch equals to 6,35mm so the hole was rounded up to a diameter of 7mm. Se figure 4.38

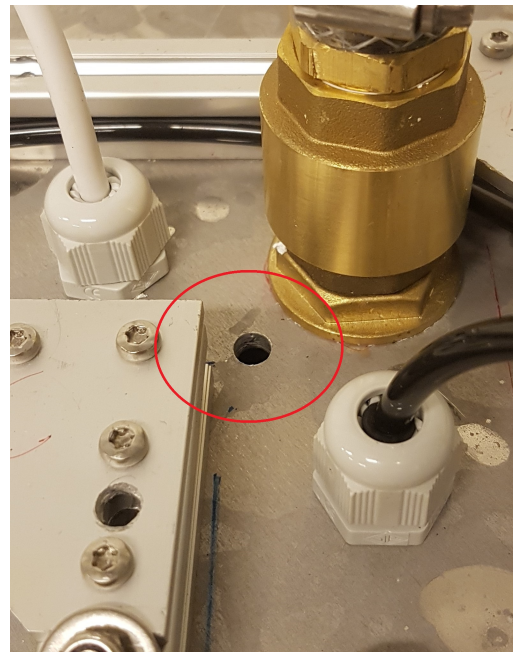


Figure 4.38: Ventilation hole

4.6.2 Water level sensors

Pressure sensors are used for measuring the water level inside each column. The sensors are placed in the electrical box on top of the platform (figure 4.39) with a hose each reaching down to the bottom of each column. The sensors used are the MPX2010DP [33] sensors which measures the pressure difference between two inputs. In this case, one is placed in the bottom of the columns while the other is not connected, so the measured values are the difference between the water level and the atmospheric pressure. The sensor output is in a mV signal, so it has to be amplified to be readable. Each sensor has its own instrumental amplifier, AD620, with a gain of 100, and outputs a voltage depending on its input voltage. Since the amplifiers' output is an analog voltage signal, the microcontroller has to convert the signals through an ADC (Analog to Digital Converter). While building this array of amplifiers, the idea of using a Raspberry Pi for the stabilisation system were considered, and since the Pi doesn't have an in-build ADC like the Arduino has, an external ADC was implemented to the drawings. The array of amplifiers also consisted of one LM358N operational amplifier, all references from the array connects to the op-amp and creates a common reference to the ADC. The op-amp is powered from a buffered voltage divider, the basic content of the wiring scheme is borrowed from Analog Devices AD620 datasheet. Later on when deciding on using the Arduino instead of the Pi, both the ADC and the LM358N was removed. [17]

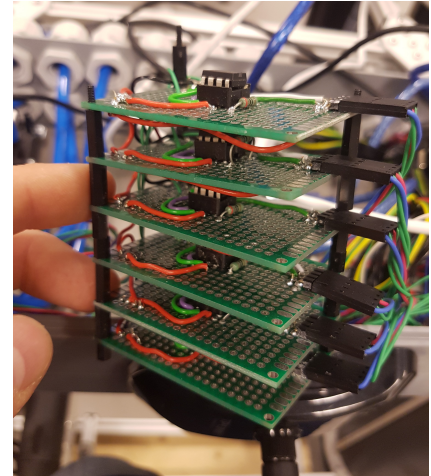


Figure 4.39: Amplifier array

4.6.3 IMU

Adafruit's 9-DoF sensor is used for measuring the platforms attitude. By using four of them, enhances stability and accuracy of the measurements compared to using just one. Since the platform has four of them and they all communicate via I^2C to the Arduino, a multiplexer was needed to separate their addresses. This way the Arduino can read all of them, one by one, almost simultaneously.

The 9-DoF sensors distributes 3-axis readings from gyrometer, accelerometer and magnetometer. The sensors are used for reading the platform's angle of list as input to the stability system, and it's heading as input for the autopilot. The placement of the sensors are on the corners of the platform. The reason for having four is actually the placement options, since they are located symmetrically they generate an average center at the platforms exact center. The mounting brackets for the IMUs seen in figure 4.41 are drawn in the same software as the platform and 3D-printed in PLA.

4.6.4 Control system

To make the control system as safe as possible, the stability system is made as a stand alone system. Therefore the decision was made to use a method called cyber-physical systems, where software and hardware are closely linked, and computing are decentralised. In this way the stability system is not reliable on the server to operate, and a failure in any other system would not affect the stability of the platform.

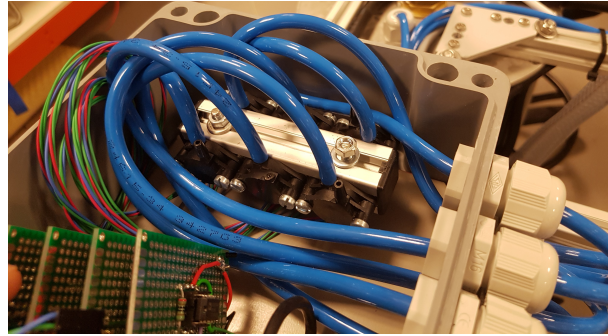


Figure 4.40: Array of six MPX2010DP

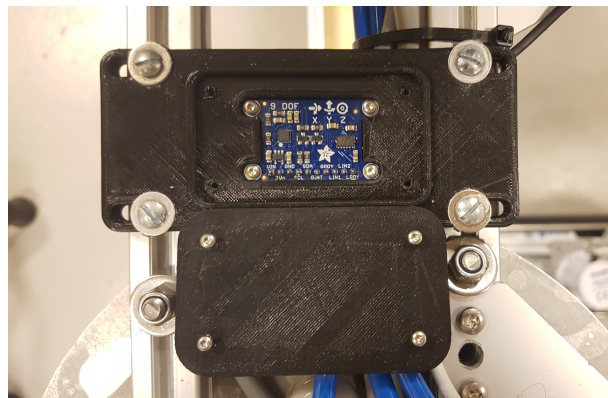


Figure 4.41: Inertial measurement unit

All components of the stability system is connected to a microcontroller, which controls the system and sends the required information to the server. The Arduino Mega is chosen for this task because it has an easy setup, and for its number of I/Os. Another advantage of using Arduino is the simple operating system with less things possible of failing. The other alternatives like Raspberry Pi and Beagle Bone, were rejected because they are much too powerful and unnecessary complex for its purpose.

Sensors used are IMUs and pressure sensors. Four IMUs are used to get a more stable and accurate measurement. They are mounted on each corner of the platform, and the average value of all four IMUs are calculated. This increases the accuracy and the average value has its pressure point in the centre of the platform. IMUs are prone to noise, and because of this the signals are run through a Kalman filter. In waves, the constant rolling will make the stability system to run at all times. This system is not designed to stop rolling, therefore a low pass filter is added to cancel this motion. Then the system only sees the list angle, which it can correct.

The pressure sensors are used to measure the draft of the platform and the height of the water inside the columns. The sensors are calibrated to show the depth in cm.

4.6.5 Stabilisation software

The software makes all the calculations and filtering. The only hardware components for the system is the sensors, relay modules and the microcontroller. In the software, a setpoint for the different motions are set. It states what is the wanted roll, pitch and draft of the platform. It tests if the filtered values exceeds the setpoint plus a tolerance. The tolerance will give the maximum list the platform will tolerate before it starts to correct. These values can be adjusted individually to set the wanted attitude of the platform. The roll and pitch have a default value of 0, for getting the platform level. The draft has to be set to adjust the current payload.

Calculating the output is done by a PID controller (chapter 2.12). It uses the Kalman filtered values as input. And since we have relays operating the water pumps, the PID output is used to calculate the time of the pumps being activated. The output value of the PID is between 0 and 100. This value is then scaled and used as the time the pumps is on before it turns off and a new on time is calculated. In this way the activation time becomes shorter and shorter the closer to the setpoint the platform gets, and the correction movement slows down preventing

an overshoot. To find the K_p , K_i and K_d , the Ziegler Nichols (2.2) method was used.

To control the draft, the pressure sensor at column 2 and 5 is used. The average of these two values are calculated. The draft is not run through a Kalman filter or a low pass filter. This was done because tests in the wave tank showed that the vertical movement was much slower than pitch and roll. Therefore there was no need for filtering or a PID. The draft has a PID allocated, but it is not used. With further testing in heavy waves and when large payloads are on board, the platform starts to heave and depth correction starts. To prevent this a low pass filter had to be implemented for the draft as well. When it comes to the need for a PID for the depth correction, the tests show that that is not necessary. The correction movement is so slow that the system is self controlled, and a PID would only make the correction even slower.

4.6.6 Stabilising time

Several tests of the stabilising system were executed in the wave tank. Due to depth limitations of the wave tank, the tests were limited on wave size, payload capacity and payload placement. The payload placement (Figure 4.42) and results are displayed in the tables below (Tables 4.4, 4.5, 4.6). The red numbers indicate test times where the platform touched the floor of the tank, and therefore might have slightly lower values. The "No Data" fields are tests that did not provide reasonable data due to the platform was resting heavily on the floor of the wave tank or high risk of causing damage to it because of oscillation caused by waves.

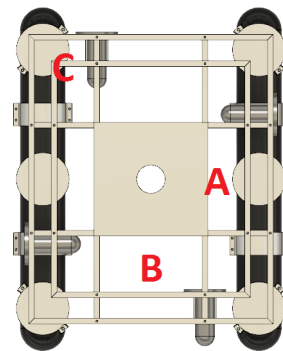


Figure 4.42: Payload placement

Table 4.4: Stabilization before draft compansation

Pos/Kg	3 Kg	5Kg	10Kg
A	6.1s	8.7s	14.9s
B	7.8s	No Data	No Data
C	13.4s	19.7s	No Data

Table 4.5: Stabilization and draft compansation

Pos/Kg	3 Kg	5Kg	10Kg
A	6.1s	8.7s	14.9s
B	7.8s	No Data	No Data
C	20.4s	20.7s	No Data

Table 4.6: Stabilization with fixed payload in position A with changes in Amplitude A and period time T on the waves

Pos/3Kg	A=150mm T=4s	A=150mm T=3s	A=200mm T=4s
A stabilization	6.0s	6.9s	5.3s
A draft correction	9.5s	6.9s	17.3s
B	No Data	No Data	No Data
C	No Data	No Data	No Data

From the results, the average time to stabilise before draft correction is 11.8 seconds, and with draft correction the average time were 15.2 seconds. The average stabilising time with draft during wave test is 31.1 seconds, and the total time to stabilise with draft correction regardless of environment is 23.3 seconds.

4.7 Thruster allocation

The thruster allocation- and configuration is based on the same algorithms used in the previous bachelor thesis *USV - Unmanned Surface Vessel* [22], but modified in therms to fit the design and thruster configuration of the platform.

For marine craft with n DOF it is necessary to distribute generalised control forces $\tau \in \mathbb{R}^n$ to the thrusters in X and Y direction in a 2 dimensional plane. The wanted forces are generated in the manual mode of the control system or in the autopilot/DP system on the server.

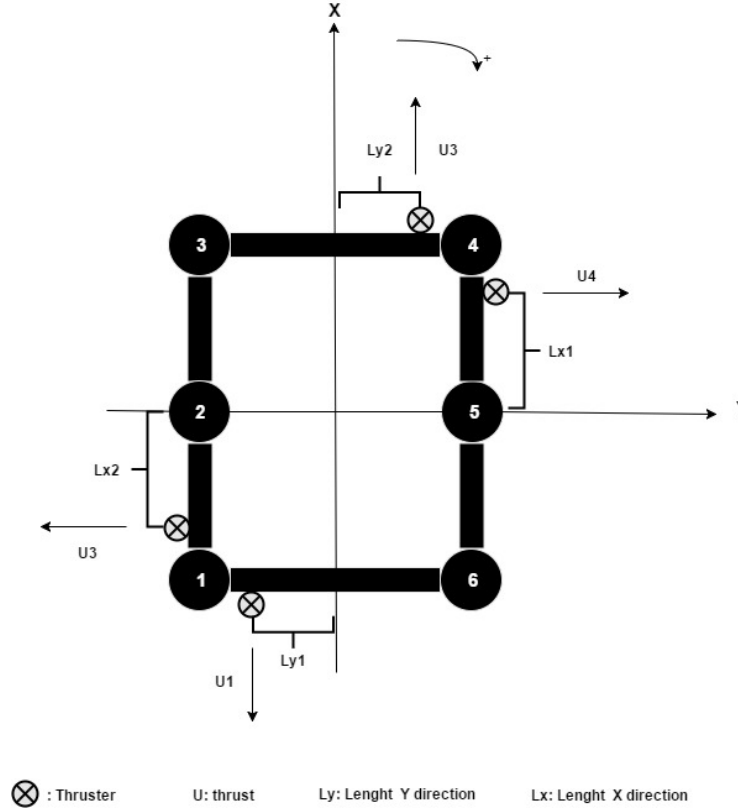


Figure 4.43: Thruster configuration

4.7.1 Thruster configuration

A force vector $\tau_{ref} = \begin{bmatrix} X & Y & N \end{bmatrix}^T$ is used as the input to the thruster allocation. X is the desired thrust in x-direction, Y is the desired thrust in y-direction and N is desired thrust about the z-axis. A positive momentum about the z-axis in a right-handed coordinate system will act clockwise, see figure(4.43). The control thrust from a single thruster is $F = u$. The thrust generated by each thruster can be denoted in a vector $\mathbf{u} = [u_1, u_2, u_3, u_4]$, while the thrust and momentum generated can be related to the control thrust τ_{ref} by the equation

$$\tau_{ref} = \mathbf{T}\mathbf{u} \quad (4.1)$$

Where \mathbf{T} is a matrix that describe the thruster configuration on the craft. [22]

For u_1 we have

$$\tau = \begin{bmatrix} 1 \\ 0 \\ L_{y1} \end{bmatrix} u_1 \quad (4.2)$$

For u_2 we have

$$\tau = \begin{bmatrix} 1 \\ 0 \\ -L_{y2} \end{bmatrix} u_2 \quad (4.3)$$

For u_3 we have

$$\tau = \begin{bmatrix} 1 \\ 0 \\ -L_{x1} \end{bmatrix} u_3 \quad (4.4)$$

For u_4 we have

$$\tau = \begin{bmatrix} 1 \\ 0 \\ L_{x2} \end{bmatrix} u_4 \quad (4.5)$$

which gives the following thruster configuration system

$$\tau_{ref} = \begin{bmatrix} X \\ Y \\ N \end{bmatrix} = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ L_{y1} & -L_{y2} & -L_{x1} & L_{x2} \end{bmatrix} = \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{bmatrix} \quad (4.6)$$

The thruster allocating consist of finding values that satisfy eq.(4.6). Measured distances perpendicular on the axis to the thrusters are as follows: $L_{x1} = L_{x2} = 0.45$ metres, and $L_{y1} = L_{y2} = 0.19$ metres, see fig(4.43). The theoretical centre of mass is retrieved from the cad drawing of the prototype without taking in account the weight of the pumps, thrusters and the electric system. CO is located where the X and Y axis cross, see fig(4.43).

4.7.2 Actuator Models

The control force due to a propeller, a rudder or a fin can be written (assuming linearity)

$$F = ku$$

Figure 4.44: Control force

where k is the force coefficient and u is the control input depending on the actuator considered; The linear model $F = ku$ can also be used to describe nonlinear monotonic control forces.

4.7.3 Solution by quadratic programming and JOptimizer

Taking the thrusters limitations in consideration, the optimisation problem has to be reformulated. The problem formulation used to solve the thruster allocation problem is based on the presented solution in section IV([12]), *Linear Quadratic Constrained Control Allocation*, The notation is from, *USV-UnmannedSurfaceVessel* [22] to match the values and variables used, and is as follows:

$$\begin{aligned} & \underset{u,s}{\text{minimise}} && u^T W u + s^T Q s \\ & \text{subject to} && T u = \tau_{ref} + s \\ & && u_{min} \leq u \leq u_{max} \end{aligned} \tag{4.7}$$

where s is a vector of slack variables that takes in consideration incidents of where τ_{ref} can't be reached by Tu . The condition $u_{min} \leq u \leq u_{max}$ ensures that the thrusters don't exceed their minimum(u_{min}) and maximum (u_{max}) values. By choosing the weighting matrix $Q \gg W > 0$, the slack variable should be close to zero, and an accurate generalised force Tu can be archived.[12]

The following explanation of the ThrustAllocator class is from (page 77-78,[22]) By defining

$$p = \begin{bmatrix} \tau_{ref}^T & u_{min}^T & u_{max}^T \end{bmatrix}$$

$$\text{and } z = \begin{bmatrix} u_{max}^T & s^T \end{bmatrix} \quad (4.8)$$

then the problem can be reformulated to the form as shown in eq. (2.14) in subsection 2.5.1

$$\begin{aligned} & \underset{z}{\text{minimise}} && z^T \Phi z \\ & \text{subject to} && A_1 z = C_1 p \\ & && A_2 z \leq C_2 p \end{aligned} \quad (4.9)$$

where

$$\begin{aligned} \Phi &= \begin{bmatrix} W & 0_{4 \times 3} \\ 0_{3 \times 4} & Q \end{bmatrix} \\ A_1 &= \begin{bmatrix} T & -I_{3 \times 3} \end{bmatrix} \\ C_1 &= \begin{bmatrix} I_{3 \times 3} & 0_{3 \times 8} \end{bmatrix} \\ A_2 &= \begin{bmatrix} -I_{4 \times 4} & 0_{4 \times 3} \\ I_{4 \times 4} & 0_{4 \times 3} \end{bmatrix} \\ C_2 &= \begin{bmatrix} 0_{4 \times 3} & -I_{4 \times 4} & 0_{4 \times 4} \\ 0_{4 \times 3} & 0_{4 \times 4} & I_{4 \times 4} \end{bmatrix} \end{aligned} \quad (4.10)$$

The open-source library JOptimizer for java is used to solve this optimisation problem. In the class ThrustAllocator these matrices are initiated in the constructor. Then a **PDQuadraticMultivariateRealFunction**-object, which is the objective function that is being minimised. This object then take the matrix Φ as a parameter in the constructor. An array **ConvexMultivariateRealFunction**-objects that set the constraint function is also initialised in the constructor. Finally an object of the class **JOptimizer** is instantiated. The JOptimizer object take care of the actual optimisation. This object use a primal-dual interior point algorithm to solve the quadratic programming problem[20]. A description of this algorithm can be found in [5]

To calculate the vector \mathbf{u} for a given τ_{ref} the method calculateOutput in the ThrustAllocator-class. This method returns an array of dimension 4 of the type double, where the elements in

the array represent the thrust each thruster should generate.

Actuator	u (control input)	α (control input)	f^\top (force vector)
Main propellers (longitudinal)	Pitch and rpm	–	$[F, 0, 0]$
Tunnel thrusters (transverse)	Pitch and rpm	–	$[0, F, 0]$
Azimuth (rotatable) thruster	Pitch and rpm	Angle	$[F \cos(\alpha), F \sin(\alpha), 0]$
Aft rudders	Angle	–	$[0, F, 0]$
Stabilizing fins	Angle	–	$[0, 0, F]$

Table 4.7: Definition of actuators and variables

4.8 Software

The software is programmed as a server-client system, where the platform functions as a server, and the control unit functions as a client. The software on the platform is written in Java and a simplified version of C. The platform is controlled by an application on an Android tablet. In the control application The operator can choose between two modes, Manual mode and Autopilot mode. The platform has two independent software systems. One system is used to control the balance system, running on a microcontroller. The other system is a concurrent system, reading GPS data, receiving and sending data over WiFi, and communicate with the micro controller that controls the thrusters as well as reading IMU data from the balance system. The third party software used to program the platform is Netbeans 8.2 and Arduino Studio. The android application is programmed in Android Studio.

4.8.1 Flow chart of the complete system

Figure 4.45 illustrates the data flow on- and between the client and the server.

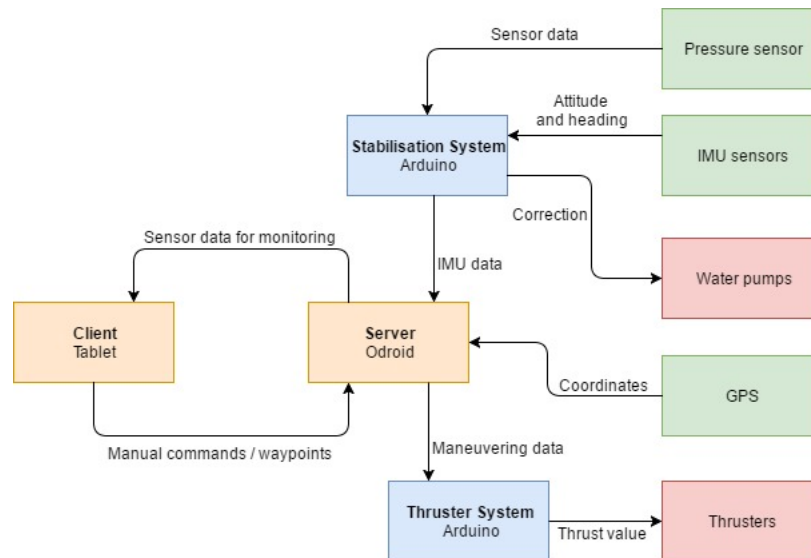


Figure 4.45: General flow chart for complete system

4.8.2 Server-Client

Manual Mode

The main purpose of the Manual Mode is to send parameters to the server for every action the operator performs. This is done by putting "key" String into a JSON Object with a value attached to each "key". Every time Manual mode is initiated, it starts with sending a JSON Object with "Mode", 1 to the server. This lets the server know that Manual mode is chosen, and the server start the corresponding thread for handling manual mode. From here on a constant TCP stream of data will be exchanged between the tablet and the platform, the client sending manoeuvring commands to server, and the server sending sensor data for the operator to monitor. When the operator is not pressing any of the buttons, the application constant sends zero power to all the thrusters, thus it will stand at rest. If the operator sets a thruster power value using the slide bar in the GUI, and then press a button, the `setOnTouchListener` method in Android Studio reads that the button is pressed and the application sends the new thruster values to the server, moving the vessel in the wanted direction. The Sending and receiving data is handled in real time by two separate Threads, this results in a fast and responsive way of communicating. The client also receives a UDP camera feed from the server with live camera feed to make it easier to operate the vessel over long distances.

The Manual mode application version created for this thesis has functionality with the possibilities to manoeuvre the platform in eight directions, activate the signal horn and lanterns, as well as displaying a camera feed received from the platform seamlessly. Alternatives are Joystick, Xbox control or a GUI interface on a remote computer

Autopilot

The Autopilot gets its functionality from the Google Maps API on Android Studio. The app locates the vessels position by reading the GPS data received on the TCP input stream, and zooms the map to that very position. In this map application, the operator can set waypoints at desired positions to make a path for the platform to follow. The communication between the client and the server works basically the same way as the manual mode. When Autopilot mode is se-

lected in the android application, it creates and sends a JSON Object with a String "key" and a value. Every time Autopilot is initiated, it starts with sending a JSON Object with "Mode", 2 to the server. This lets the server know that Autopilot is chosen, and the corresponding thread for handling autopilot mode is started. The application works by storing the positions put on the map into two separate JSON Array lists, where one handle the latitude coordinates, and the other the longitude coordinates. Both those lists are then stored into a single JSON Object before it is sent to the server. When this data is sent, the server receives and process the data, where coordinates are extracted in longitude and latitude pairs. Direction and distance to the waypoint is calculated by using GPS data combined with the received coordinate data and the platform will perform various calculations before the thrust is set. When the platform has reached its destination within 5 metres of a waypoint it is set to extract a new pair of coordinates and repeat the process. If a new pair of coordinates can't be found in the received path, it will try to hold the position of the last coordinate pair of the path.

The version of the Autopilot mode in the android application in this thesis sends and receives position data, and the operator can monitor the platform's position in real time. The signal horn and lanterns functionality is not implemented as a part of the autopilot due to priorities of features with more importance for the autopilot system. The same applies for the DP simulation on the application.

4.8.3 Float chart software

Figure 4.38 and 4.39 shows the data float in the server and client

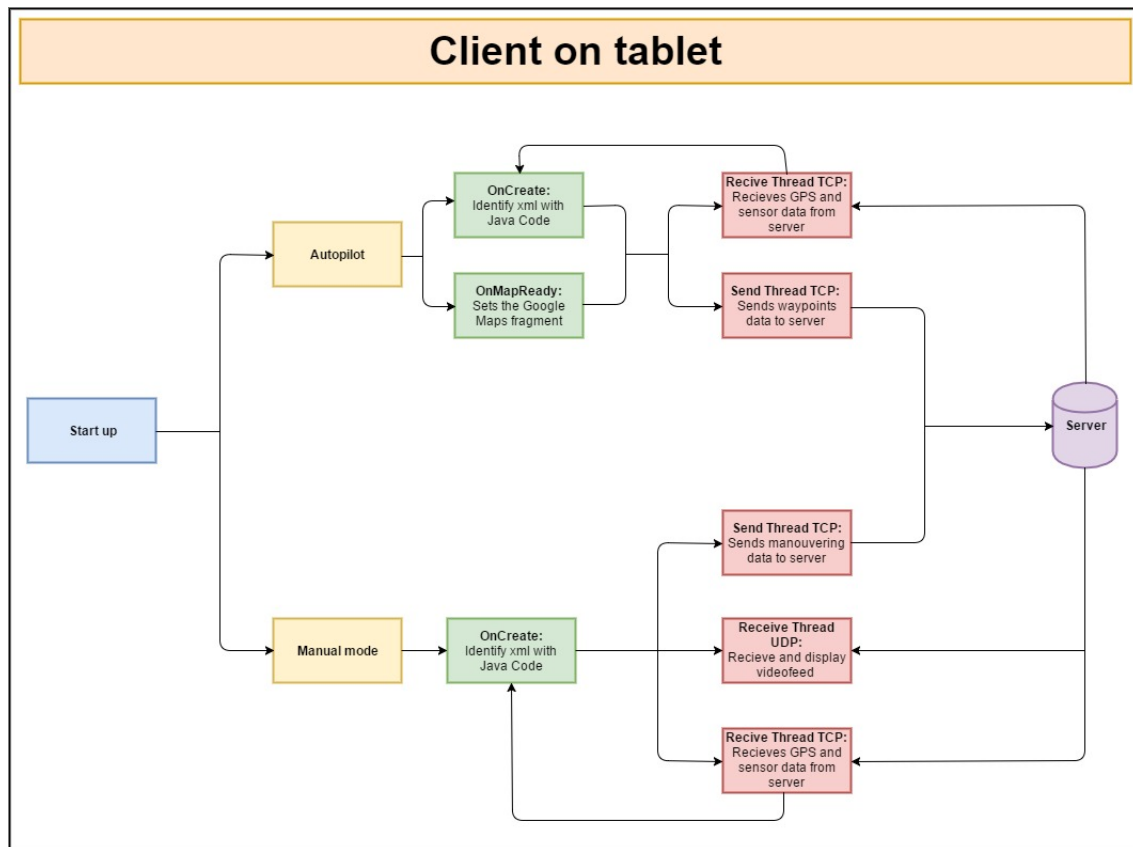


Figure 4.46: Float chart client

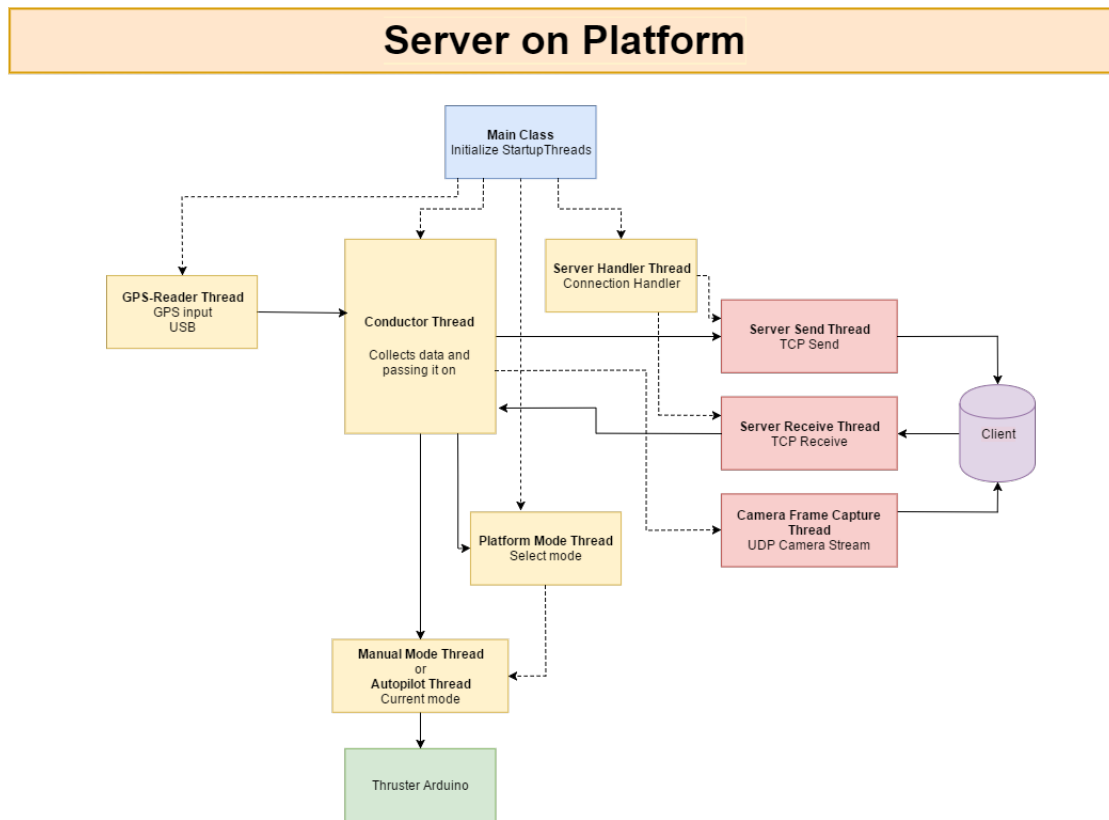


Figure 4.47: Float chart server

4.8.4 Class Diagram

Client

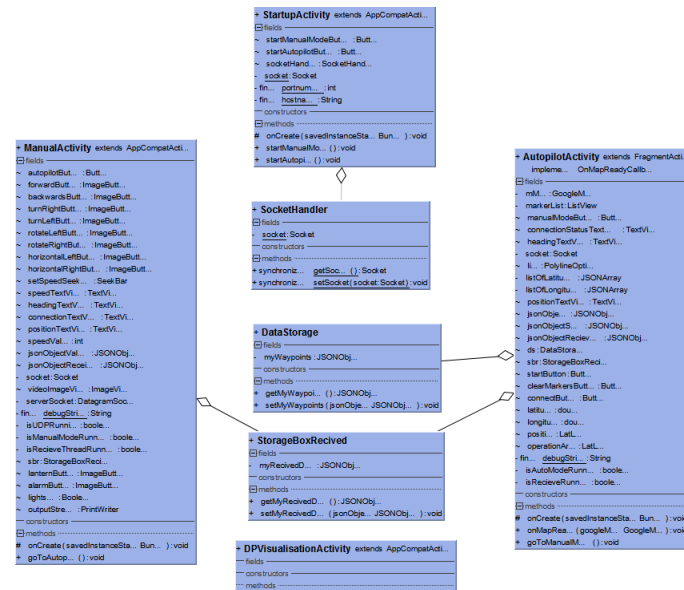


Figure 4.48: Clas Diagram Client

Server

4.8.5 Graphical User Interface



Figure 4.49: Mode GUI

This is the start up GUI, see figure 4.49, that meets the operator, and the client automatically connects to the server. From this screen, the operator get to choose between Manual mode and

Autopilot by tapping either one of the buttons.

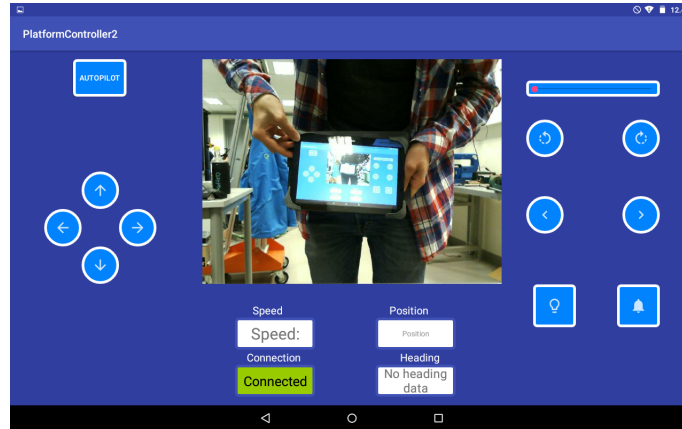


Figure 4.50: Manual mode GUI

Manual mode, illustrated in figure 4.50, let the operator control every movement of the vessel. A continuous stream of commands are sent to the server when the operator tap a button. On the left hand side the buttons moves the vessel forward, backwards, and turns it right and left. The sidebar in the right top corner sets the output thrust. The buttons with a circular arrow below the sidebar makes the vessel pivot, and the symbols < and > makes it go straight sideways. The squared buttons controls the lantern and the foghorn, while the rectangular text boxes are used to display sensor data from the vessel, and connection status.



Figure 4.51: Autopilot GUI

The Autopilot, illustrated in figure 4.51, gives a GUI that shows the operating area and displays the current position of the vessel. Markers can be set anywhere on the map by a touch

and hold action on the location where it's desired. The GPS coordinates will be listed in the blue margin to the right in the screen. A path can be created by placing multiple markers. By tapping the START button, the vessel will start moving towards the destination, while the green marker indicates the trajectory by updating the current position once a second. The rectangular text boxes are used to display sensor data from the vessel and connection status.

4.8.6 Server application

The program running on the Odroid XU4 is launched from a "main" class which. See illustration in figure 4.47. The main class has one purpose, and that is to start five of the total eight threads when the server starts. The "Server Handler Thread" handles the TCP socket connection with the client. If the client tries to connect, it accepts and starts a receiving and sending thread. The "Platform Mode Thread" will start corresponding "mode thread" on request from the client. Each thread has its own task. Whether it's reading a sensor, sending- and reading data over WiFi, or gathering it before processing.

4.8.7 Reading sensors on the platform

The GPS and Arduinos were connected to the Odroid by USB, and the RXTXcomm.jar library were used to communicate through serial. The Arduinos use the built in Serial library to communicate with the server application. In the server application the GPS is assigned a Thread that read the GPS input data at a constant time interval that is set to 200ms. To read the input data on the USB ports the method "getInputStream()" from the SerialPort class is used, and to send data over the USB the "getOutputStream()" method from the same class is used. SerialPort is a class found in the RXTXcomm library. Due to difficulties with the opencv2413 library on the Odroid, the Odroid were substituted by a computer running Windows 10 for several tests of the UDP stream in manual mode.

4.8.8 Sensor data processing on the platform

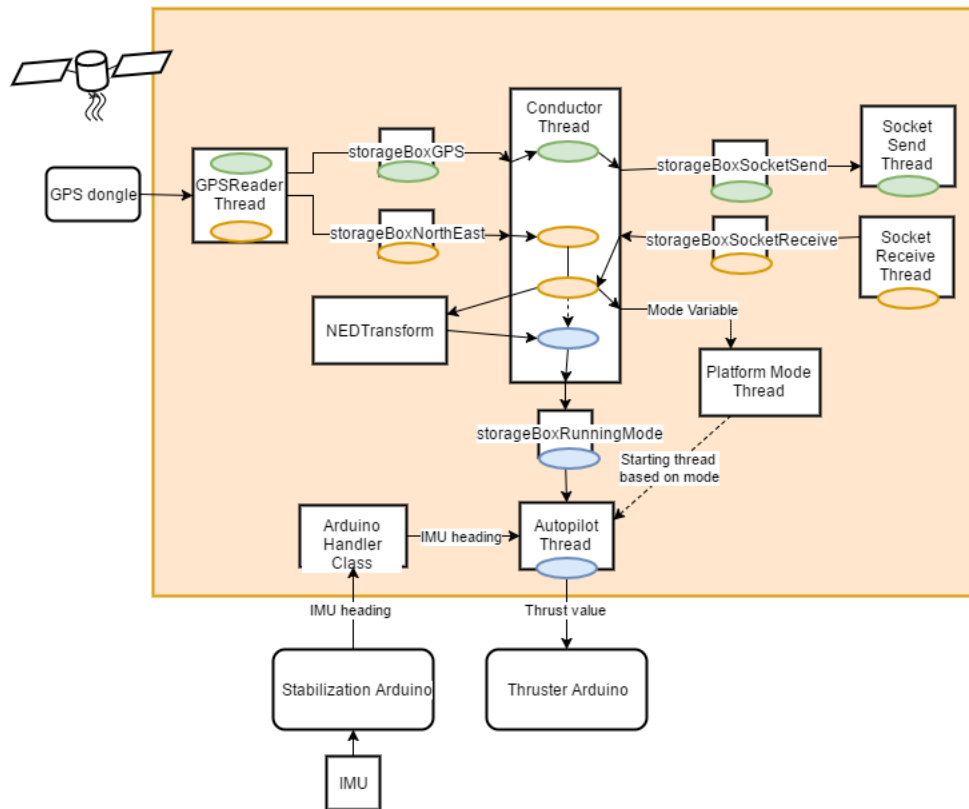


Figure 4.52: Flow chart sensor data

Communication between threads, illustrated in figure(4.52), are done by having shared resources, referred to as storage boxes, and Semaphores that are used to maintain a thread safe structure between them. NMEA sentences are read from the GPS dongle in the GPSReader thread, and latitude, longitude and speed are extracted from the NMEA sentence and stored as lat, lon and spd of the type double. Latitude and Longitude are transformed to radians before the GPSReader thread then put the variables lat, lon and spd into storageBoxGPS, and a JSONObject containing coordinate data as radians and degrees as well as speed into the storageBoxNorthEast object. Then the "conductor thread" will check if the storageBoxGPS and storageBoxNorthEast contain GPS data, and if they do, the conductor thread will retrieve information and store it as two different JSONObjects. The JSONObject with the information containing only GPS data as degrees will be passed toward the socketSend thread. The other, that contains the complete GPS data will be extended with waypoint data received from the socketReceive thread which reads a TCP

input stream from the client. The waypoint data is received as a JSONObject containing two JSONArrays and a "mode field", where one of the JSONArray lists contain Latitude the other Longitude coordinates. If the mode field equals "autopilot", represented by the int 2, The conductor thread will pass the received mode parameter to the Platform Mode thread, which then will start autopilot mode, and stop the manual mode if that current mode is already running. The received waypoint lists are extracted and the coordinates are converted to radians. Waypoints are set, and before the the position info is sent to the StorageBoxRunningMode object, which is shared between the conductor and autopilot thread, a method called "getFlatEarthCoordinates()" within the "posInfo()" method is called. getFlatEarthCoordinates() converts the geodetic position of the craft into the North-East frame, from chapter(2.3.3). The method for conversion between the two reference frames are shown below.

Listing 4.1: NEDTransform Java code

```
public JSONObject getFlatEarthCoordinates(JSONObject jsonFlatEarth) {
    // Tar ut enkelte variabler av JSONObjektet
    try {
        double latBody = jsonFlatEarth.getDouble("xyLatBody");
        double lonBody = jsonFlatEarth.getDouble("xyLonBody");
        double latRef = jsonFlatEarth.getDouble("xyLatWaypoint");
        double lonRef = jsonFlatEarth.getDouble("xyLonWaypoint");
        // *****

        double dMy = latBody - latRef;
        double dL = lonBody - lonRef;

        double rN = (R / (Math.sqrt(1 - (2 * f - f * f)
            * Math.pow(Math.sin(latRef), 2))));
        double rM = rN * ((1 - (2 * f - f * f)) / (1 - (2 * f - f * f)
            * Math.pow(Math.sin(latRef), 2)));
        double dN = (dMy / Math.atan(1 / rM));
        double dE = (dL / Math.atan(1 / (rN * Math.cos(latRef))));
```

```

        jsonFlatEarth.remove("latBody");
        jsonFlatEarth.remove("lonBody");
        jsonFlatEarth.remove("latRef");
        jsonFlatEarth.remove("lonRef");

        jsonFlatEarth.put("dNorth", dN);
        jsonFlatEarth.put("dEast", dE);
    } catch (JSONException ex) {
        Logger.getLogger(NEDTransform.class.getName()).log(Level.SEVERE, null, ex)
    }

    return jsonFlatEarth;
}

```

The input parameter is a JSONObject containing the geodetic position of the craft, and waypoints received from the client. latBody and lonBody is the crafts geodetic position in radians, latRef and lonRef is the geodetic reference coordinates of the waypoint. R is the equatorial radius of the earth, and f is its flattening. The method `Math.pow(Math.sin(latRef), 2)` returns the value of the first argument raised to the power of the second argument. The deviation in north and east is respectively dN and dE, which then replaces the old values of the JSONObject used in the method.

In the autopilot thread a distance check to the waypoint is done by calling the "haversine()" method, as shown below.

Listing 4.2: Haversine Java code

```

public static double haversine(double lat1, double lon1, double lat2, double lon2) {
    double dLat = Math.toRadians(lat2 - lat1);
    double dLon = Math.toRadians(lon2 - lon1);

```

```

lat1 = Math.toRadians(lat1);
lat2 = Math.toRadians(lat2);

double a = Math.pow(Math.sin(dLat / 2), 2)
           + Math.pow(Math.sin(dLon / 2), 2)
           * Math.cos(lat1) * Math.cos(lat2);

double c = 2 * Math.asin(Math.sqrt(a));
distanceToWaypoint = RADIUS * c;

return distanceToWaypoint;
}

```

The first two parameters, lat1 and lon1 are the crafts geodetic position, and the last two parameters lat2 and lon2 are the geodetic reference coordinates to the first coordinate pair of the waypoint lists received from the client, all coordinate parameters are in degrees. The methods `Math.pow(Math.sin(dLat / 2), 2)` and `Math.pow(Math.sin(dLon / 2), 2)`, returns the value of the first argument raised to the power of the second argument. RADIUS is the equatorial radius of the earth. The return value of the method is the distance to the waypoint in metres.

4.8.9 Autopilot and Dynamic positioning

When the platform is in autopilot mode, the Conductor thread read the GPS position that is already stored in the storageBoxNorthEast (see figure 4.52). Waypoint coordinate lists are retrieved from a JSONObject received from the client, and values are converted to N-and E- values in NED by utilising the NEDTransform class. A JSONObject, containing the transformed coordinates are put to the common object storageBoxRunningMode, between the Conductor thread and the Autopilot thread. The Autopilot thread starts by reading the IMU heading from an Arduino, before it retrieves the coordinate data from the storage box. The autopilot thread will check if it is within a range of two metres by utilising the *haversine* formula described in section 2.9 and as a Java implementation in 4.2, before calculate a heading and a distance to the waypoint. The PID controllers generate an output thrust on each thruster based on the coordinates and from the NEDTransformation, IMUheading and heading to the waypoint. The

platform will try to hold the position on the last waypoint.

4.8.10 PID control for thrust

A standard PID controller [2.12](#) is used. One for latitude, one for longitude and one for heading. Input for the latitude and the longitude PIDs are GPS- and waypoint coordinates transformed to radians. Because of very small numbers, the radian values are increased by a factor of a hundred on the PID input. The tuning of the PID is done in the constructor of the Autopilot class.

4.8.11 Thruster control

In autopilot mode the thrusters are controlled by the AutoPilot class, which is instantiated as a thread. After the PID controller has generated the output thrust, the thrust is transformed to BODY-coordinates, (see [2.3.2](#)), by using the rotation matrix [2.4.1](#) for rotation about the Z-axis. Then the a thruster allocation is executed by the method `calculateOutput(XYNtransformed)`, where XYNtransformed is the transformed thrustvector. The thrust for each thruster is set by calling the method `setThrustForAll(forceOutputNewton)` and `write thrust`, where `forceOutputNewton` is an array consisting of four double values, from an object made of the ThrustWriter class. The Manual mode also use an object of the ThrustWriter class to write the output thrust to the platform, but the output values are set in the client application.

Arduino thruster control

The Arduino controlling the thrusters read a char array on the input stream, sent from the ThrustWriter's `writeThrust()` method. The char array is parsed to a JSONObject, and values from the JSONObject is extracted- and casted as integers. The integers are then fed directly into the thruster speed methods, one for each thruster.

4.9 Results from tests at sea

Wheels were mounted on the platform to ease the transportation from the workshop, and to set it afloat. A sheet of Styrofoam was placed below deck to add extra buoyancy as a fail-safe effort,

because the potential the stabilisation system has to sink the platform if the software would freeze. This however was not a problem while testing seeing the system performed as it were supposed to.

4.9.1 Autopilot mode

During testing of the autopilot, the platform behaved different than expected. There were done several tests on land before the tests at sea. during the land tests, a waypoint was placed in the test area on the map, and the group pushed the platform in the thruster directions. The thrusters seemed to function properly. During the tests in the sea, the feedback from the platform movement exceed the expectations. The platform pivoted too quickly, and the PID would overcompensate the thrust in order to correct the heading. The PID were adjusted, but the platform kept pivoting to the left in a curved trajectory from where it started toward its waypoint.

4.9.2 Manual mode

The manual mode has been working while testing in the wave tank at school. What was noticed during the sea test however, is that one of the thrusters didn't run at all, the Arduino seemed to deliver the correct information to the motor controllers.

4.10 Results from wave test

After completing the platforms mechanical structure and electrical wiring, some tests were run in the water-tank at the university. The main goal of the tests was to observe how the platform behaves in waves, while the stabilisation system is active. During the tests, a gathering of data were done for each of the sensors on the platform and the results are presented in this section of the report.

4.10.1 Results from one vs four IMUs

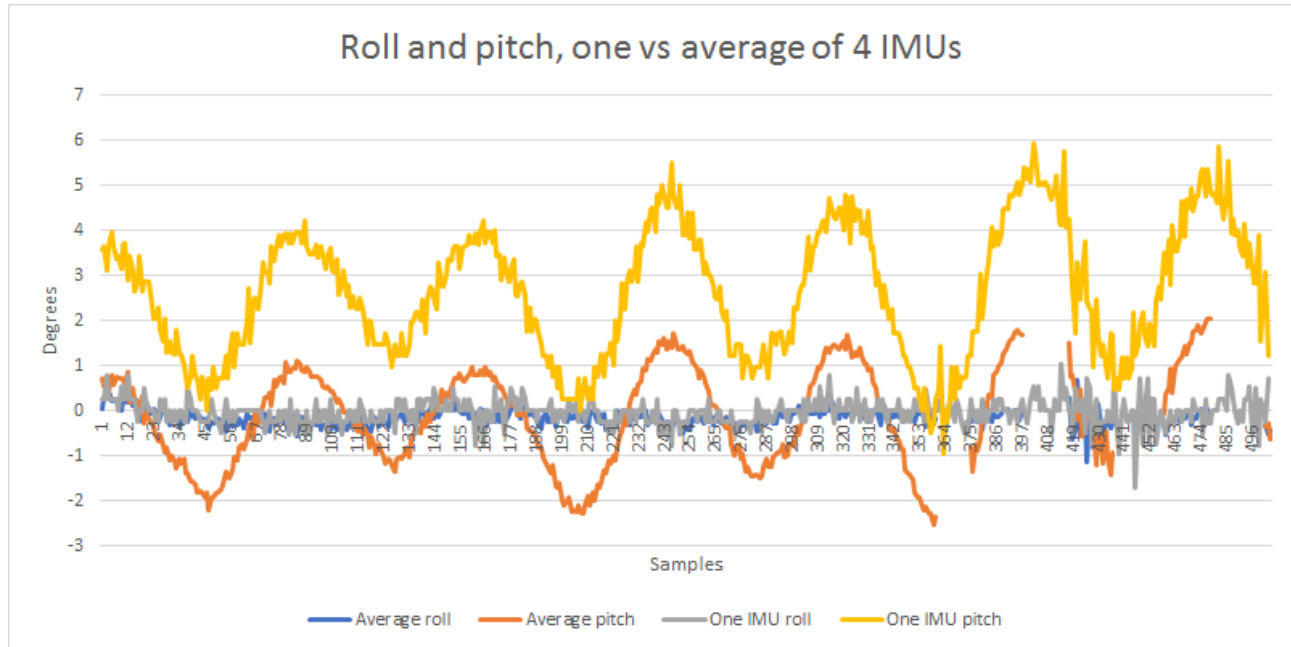


Figure 4.53: One vs Four IMU

In this specific test, illustrated with figure 4.53, the platform was placed lengthwise so the waves hit the front and a 5kg payload was placed on top. The amplitude of the wave was 150mm and a wave-period of 2 seconds. As the graph indicates, the signal difference between one and four IMUs is mainly the location around zero, but the average of four is also a bit more accurate. A reason for why the single IMU lays above the other may be that this sensor has a slight tilt forward compared to the others or that the platform itself is not completely level at all corners.

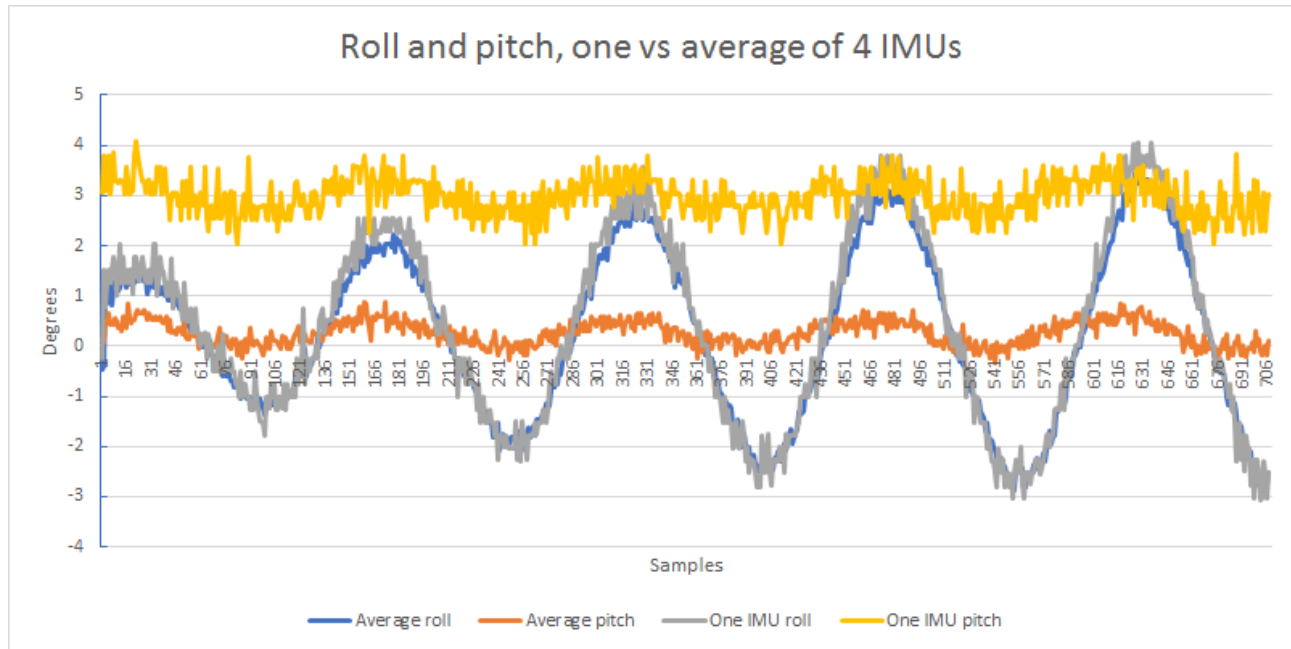


Figure 4.54: One vs Four IMU

In another test, illustrated in figure 4.54, the platform was turned 90 degrees so that the waves came in from the side. Wave amplitude 150mm, wave-period 4 seconds, 10kg payload. As stated above, the pitch from the one IMU seems to be offset by around 3 degrees and the roll is consistent around the average. What is learned from this is that using several sensors increases the accuracy overall, the signal is more stable and is stabilised relatively close to zero.

4.10.2 Kalman filter

Even though the signals were improved using four sensors, they were still too noisy to be used. Some filtration methods were tested beforehand but the Kalman filter seemed to give the best result.

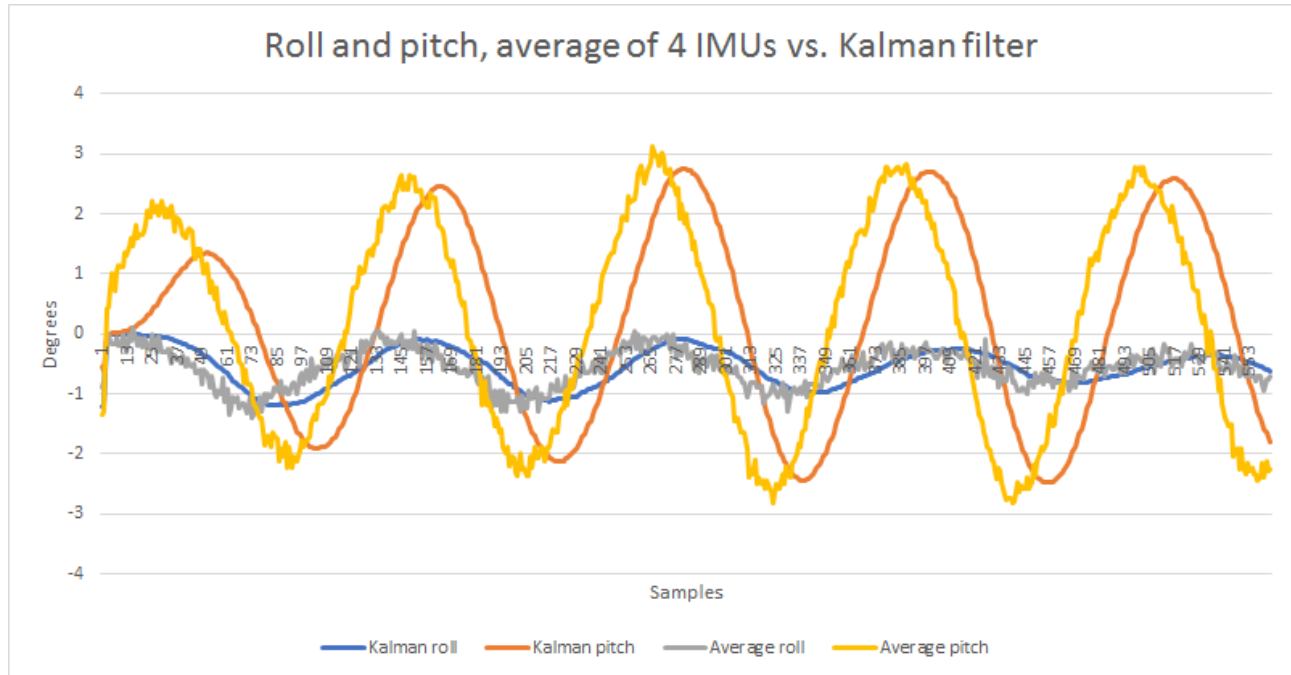


Figure 4.55: Kalman filter

As shown in figure 4.55, the Kalman filter calculates its input-values and predict the next step. For doing this it uses the sensors angular output as well as the angular acceleration. The filter cancels noise but leaves a slight delay, this however is very small and not at all noticeable in real time. This test was run with a wave amplitude of 150mm, wave-period of 3 seconds and no payload.

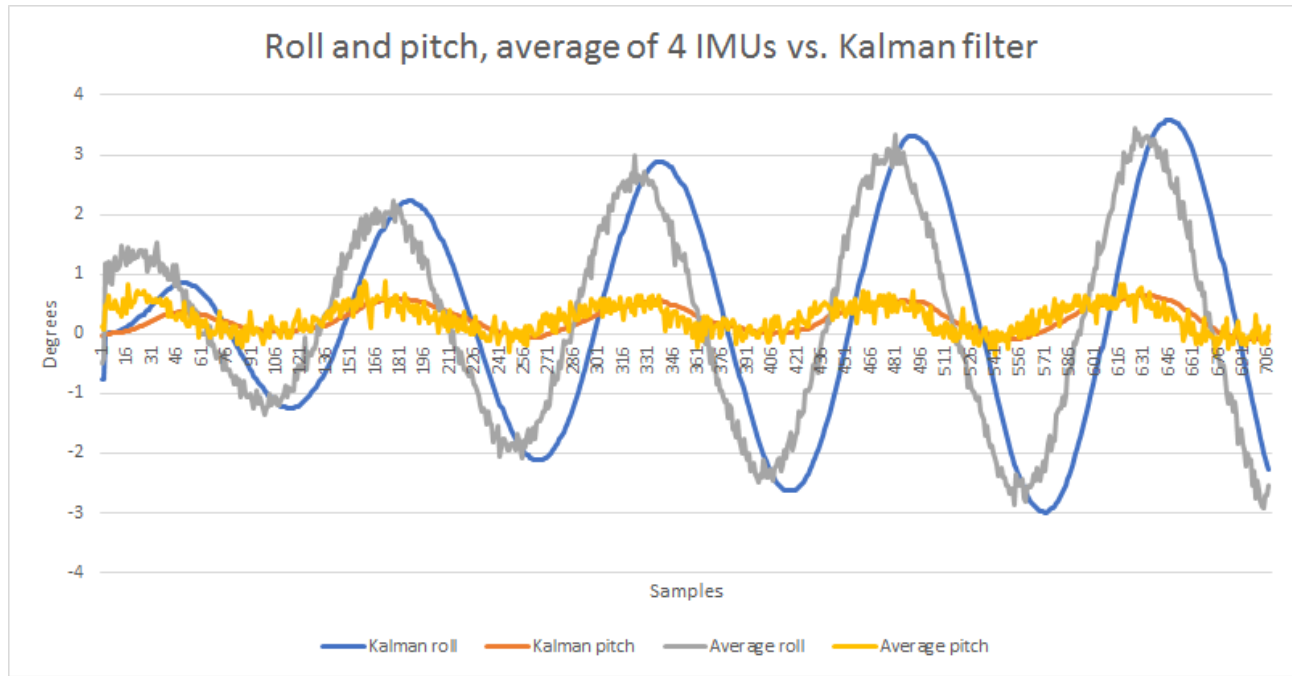


Figure 4.56: Kalman filter

Figure 4.56 shows another example of behaviour, the platform is turned 90 degrees with waves in from the side, wave amplitude of 150mm, wave-period of 4 seconds and 10kg payload. As the already accurate signals of the four IMUs as input, the Kalman filter generates smooth predictions of the signal. The results from Kalman filtration is a good real-time input for knowing the exact angle of the platform.

4.10.3 Low-pass filter

For the angular signals to be usable for the stability system, yet another filter is needed. If the Kalman values were to be used as input for the stability system, the platform would start to oscillate, creating an unstable system.

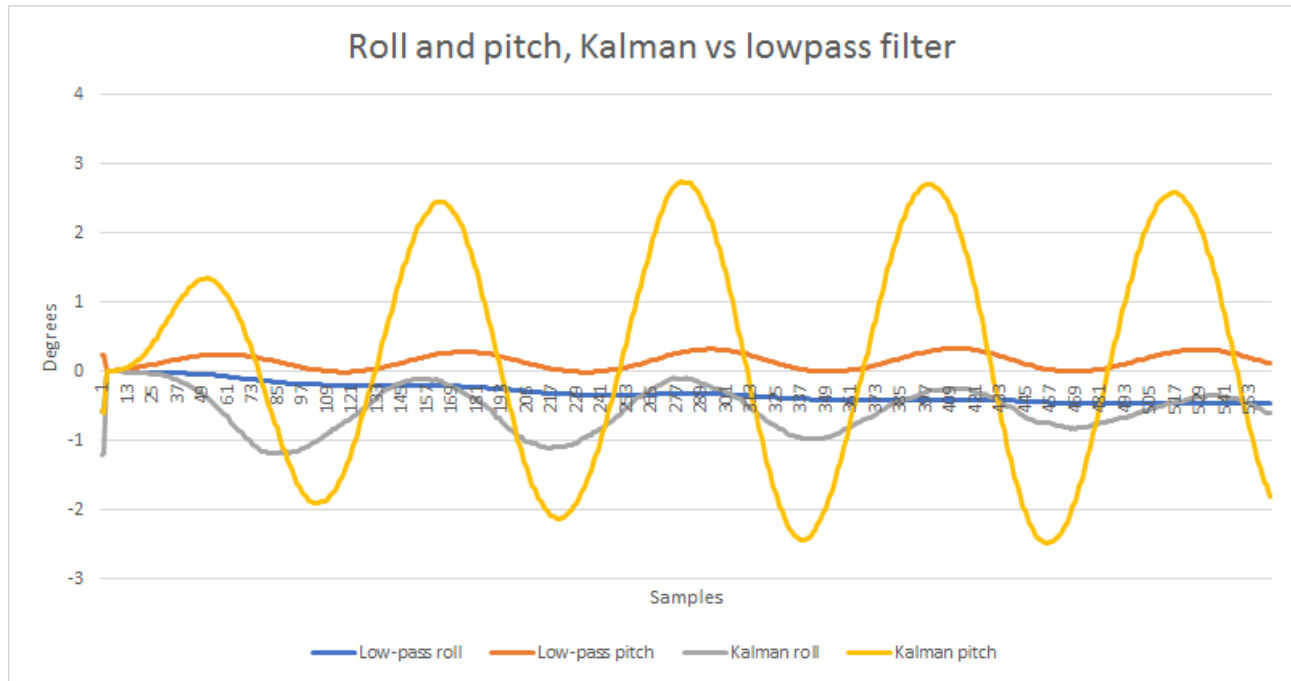


Figure 4.57: Low-pass filter

Wave amplitude: 150mm Wave-period: 3 seconds Payload: 0kg

The low-pass filter inputs the Kalman values and has a filter frequency of 0.02. As seen from graph 4.57, the low-pass filter is slow enough to cancel oscillations on the platform, making the stability-system only capable of correcting a consistent angle of list.

4.10.4 Platform movement from waves

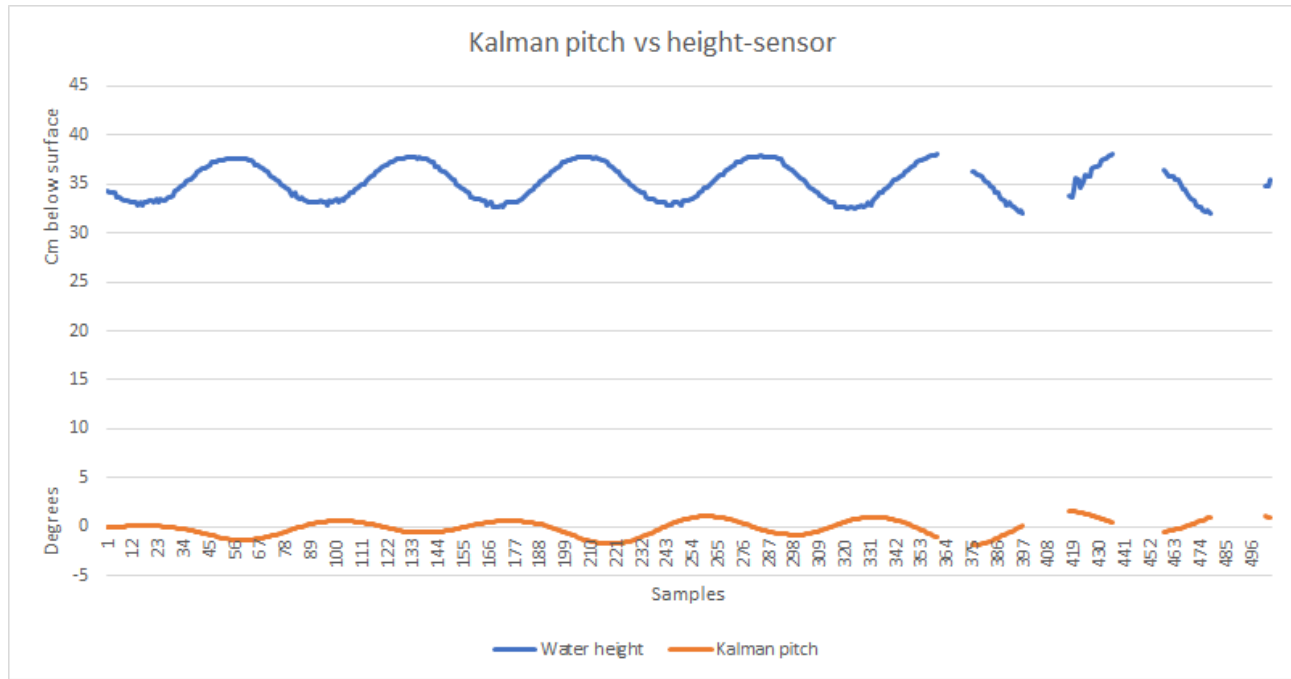


Figure 4.58: Low-pass filter

Wave amplitude: 150mm Wave-period: 2 seconds Payload: 5kg

Considering only the design of the platform, it is very resistant for creating oscillations. The top blue line in figure 4.58 is created by the platforms water height sensor, in this graph the line generates a good presentation of the waves. With this wave illustration you can see just how little the pitch is oscillating from it, only a couple of degrees as the waves are 300mm from top to bottom. There are some parts of the graph that is cut off and the reason for this is the system "thinks" that the platform is too high or too low in the water, and therefore starts a sequence for height compensation. As the platform could very well bounce up and down a bit, this is not a reasonable solution. This was first noticed while running these tests so later on there was implemented a low-pass filter for the water height as well as the pitch and roll.

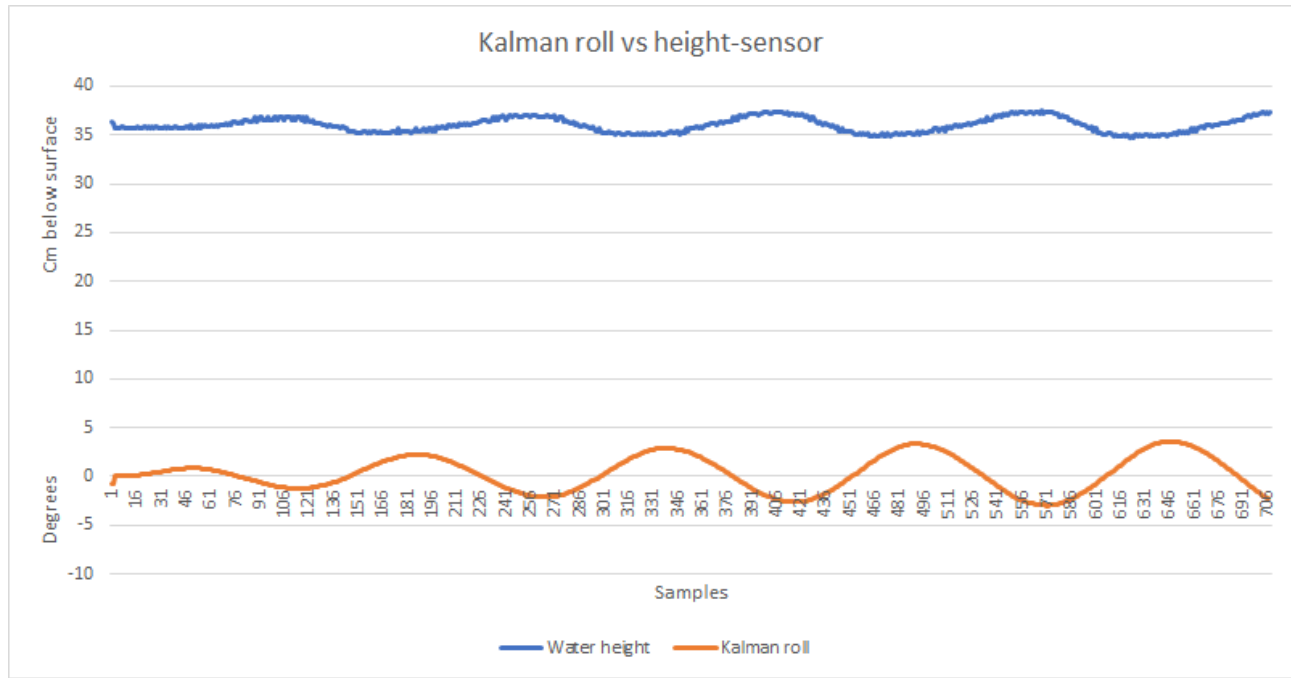


Figure 4.59: Low-pass filter

Turned 90 degrees, waves in from side. Wave amplitude: 150mm Wave-period: 4 seconds
 Payload: 10kg

At this wave setting, something interesting happens in graph 4.59, as the platform seems unstable and starts to oscillate. The small waves are creating large angular oscillations, in other means, the wave frequency hit the platforms resonance frequency with its given payload.

4.11 Wireless communication

The wireless system was tested on land between the router TP-Link Archer C5 v2 and two laptops. The laptop that was used to test network range, used a TL-WN722N USB network adapter, while on the other laptop the built in wireless network card was used.

By utilising Friis transmission formula from chapter 2.22, a range estimate was calculated by the formula derived for range without signal loss to the surroundings from, and with an estimate of 10 dB signal loss from (2.24).

Transforming the dB and dBm values to real vectors:

$$P_t = 20dBm = 10^2 W \quad P_r = S_{i,min} = -68dBm = 10^{-6.8}$$

$$G_t = 8dBi = 10^{0.8} \quad G_r = 8dBi = 10^{0.8}$$

$$f = 2.45GHz \quad c = 3 \times 10^8$$

$$\lambda_0 = \frac{c}{f} = \frac{3 \times 10^8}{2.45 \times 10^9} = 0.1224m$$

$$L_{sys} = 10dB = 10^1$$

From eq.(2.23)

$$\begin{aligned} R &= \left[\frac{P_t G_t G_r \lambda_0^2}{(4\pi)^2 S_{i,min}} \right]^{1/2} \\ &= \left[\frac{10^2 10^{0.8} 10^{0.8} 0.1224^2}{(4\pi)^2 10^{-6.8}} \right]^{1/2} \\ &= 1544m \end{aligned}$$

By adding an estimate of 10 dB signal loss due to antenna atmospheric loss, polarization mismatch, impedance mismatch at the antenna feeds, misalignment, and obstructions, the eq(2.24) gives following range estimate:

$$\begin{aligned}
 R &= \left[\frac{P_t G_t G_r \lambda_0^2}{(4\pi)^2 S_{i,min} L_{sys}} \right]^{1/2} \\
 &= \left[\frac{10^2 10^{0.8} 10^{0.8} 0.1224^2}{(4\pi)^2 10^{-6.8} 10^1} \right]^{1/2} \\
 &= 488m
 \end{aligned}$$

During the range test, the command window with the built in ping function to send data packets, and the third party software inSSIDer Home to read the wireless signal strength every 25m out to 225m. Then a final test at approximately 385m (Figure 4.60) were taken with several antenna combinations on the router and the laptop (Figure 4.61).

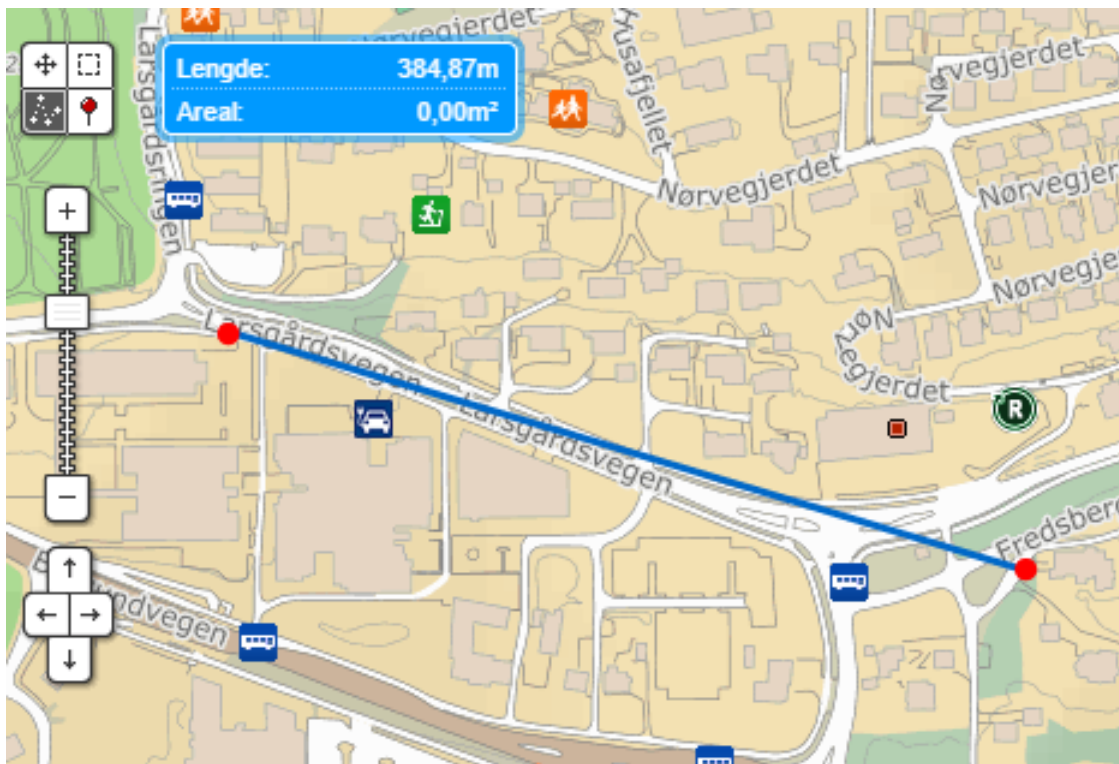


Figure 4.60: Wireless range max distance

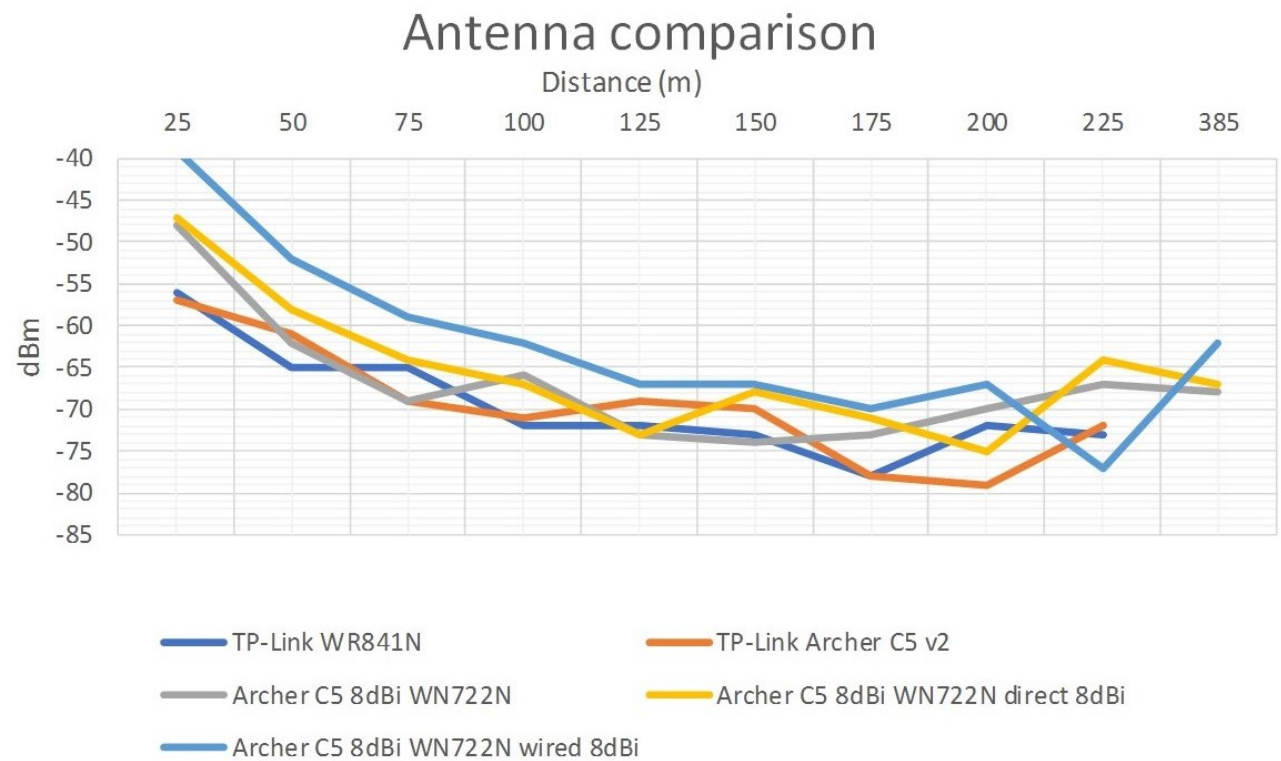


Figure 4.61: Wireless range comparison

Chapter 5

Discussion

5.1 Test results

5.1.1 Platform design and buoyancy

The platform performed very well in waves, it countered movement from waves exceptionally, however the buoyancy of the platform could definitely be better. It floats a bit low in the water due to its total weight. As mentioned in chapter [4.4.5](#), the weight of the platform at the second test was 86.824 Kg, which seemed to be ideal for a weight to buoyancy ratio. The weight of equipment such as electrical components and cables was greatly undervalued at the design part of the project, as the end result weighed 114 Kg. For getting closer to the ideal ratio, an option would be to increase the pipe diameter in the bottom two battery pontoons from 160mm to 180mm, this way you would benefit from having more buoyancy as well as possibility of adding more batteries. Since the increasing of buoyancy would be at the bottom part of the platform, the stability would also be influenced, but overall the platform is believed to perform better.

5.1.2 Stability system

The stability system on the platform performs as expected during the tests, the amount of time from a payload is placed and until the platform is level is well within the specified 30 seconds. As stated in chapter [4.6.5](#), the water pumps are driven directly from relays, which is either ON or OFF. The decision of using relays was only the fact that the parts were available at the time

being and were not reconsidered until the testing of the system. While the pumps only can be turned on or off, the only way to regulate the process is by regulating the amount of time they are turned on. For doing this the, system is integrated with a PID controller. After doing these tests, an idea of using the same motor controllers for the water pumps as used for the thrusters came to mind. These controllers regulate the voltage to the pumps with 3200 steps in each direction, even though the pumps only function in one direction, the regulation process would be much smoother. If these controllers were used along with the PID controller, the stabilisation system would have definitely performed smoother and perhaps even faster. An unknown factor to this theory is the amount of water delivered from the pump at given PWM. This would have to be tested and documented for use in the controlling.

5.1.3 Software solutions

Client

The client application running on Android was a interesting experience where the group learned a lot of the strengths of app programming and the possibilities of developing software on mobile units. One of the biggest advantages of developing the client in Android Studio is that there is a easy and lucid way of designing a GUI and implement it with the actual software code. More and more systems will have this kind of programming and it has been very useful to work with this software development. However, there where only a few members of the group that had a little experience with Android programming, and of course, this results in a lot of research to find good solutions for making the client work. Also, the class structure in the application suffers from this lack of experience. Separating each thread into individual classes is something that can and should be done, and thus improve cohesion and loose coupling, but the group decided to keep the first version of the application since it works and is crucial for testing the whole project.

Server The server application was programmed as a concurrent Java program. The reasons for this was because of the responsiveness, and utilisation of the hardware on the Odroid XU4. Semaphores were used as a Thread safe mechanism, which worked fine. An option would be to utilise synchronized blocks and methods. The code seemed to work very well on windows 10 and Linux 64-bit systems, but the Odroid's arm7l 32-bit architecture caused several problems

with certain libraries, such as RXTXcomm and openCV. The RXTXcomm problems where the operative system would assign the USB ports randomly were solved after the final test. But once again the ".so" files for the libraries could not be located when the java application was exported as a ".jar" file. Optimally a Bourne shell script should launch the server application on startup. The platform mode thread seem to not stop and start the threads as requested from the client.

5.1.4 Autopilot and Dynamic Positioning

To implement a functional and reliable autopilot to this vessel, it needs a sensor input for knowing where to go, and what to avoid. As discussed in chapter 3, the plan was to mount a 360 degree proximity sensor on the platform. The solution for how the platform handles this input has yet to be implemented to the software. Since this platform is designed to operate itself, it has to recognise foreign obstacles in the surface, such as smaller boats. Larger ships mounted with an AIS system could potentially be scanned and taken into consideration without actually seeing them. The current autopilot system was intended to navigate along a path created on the client application. During out test of this system, a problem seemed to occur over and over again which caused the rear thruster to stop. The autopilot will follow the path by correcting the error between its current position and the waypoints of the path in the order they are set in the client application. This causes the autopilot to function much like a DP system, except it does not provide any sort of mechanism to control the heading as it moves along a path, or trying to hold the last waypoint. A proper DP system would benefit this platform by keeping it on a certain location while performing a job, with a known and controllable bearing, and without drifting away. Even though this system is not yet implemented to the platform, it is believed to be a short way from the current autopilot to a functional dynamic positioning system.

As a first version of the autopilot, the group is satisfied with the functionality Android Studio has given with Google Maps API, and user friendly GUI options. This could be solved in several different ways, using a laptop and other API's for maps and monitoring. As mentioned, the remaining time was not enough to finish a DP simulation function on the client side, so the operator could monitor the vessel closing in on a single waypoint. All the data needed to make this simulation is available from the server side, and the group has a general idea how to implement its functionality.

5.1.5 Wireless Communication system

The wireless communication system exceeded the desired range specification for this prototype, but did not completely meet the estimated range. The reasons for the result's aberration to the estimated range are many, and mentioned in 2.6, but antenna miss alignment and obstacles were obvious sources to signal loss. The router use a 12v power supply, which made the solution very portable because of the easy access to 12v batteries. The TP-link Archer C5 v2 has a very intuitive interface with many customisation possibilities, and the TP-link WN722N is a great plug and play, high gain USB wireless adapter. Both TP-link devices have detachable antennas, which were replaced by omnidirectional indoor antennas with a gain of 8dBi. The WLAN solution worked for this project, but the antennas should be replaced with outdoor antennas for a more rigid solution.

5.1.6 Thruster feedback

While testing the platform in sea, one of the motor controllers stopped working, the only way to recognise the fault was visually as the propeller didn't move. A feedback system to the client would be a good addition to the system where the motor controller register any errors, and could distribute these to the server.

5.2 Stabilisation method

As the platform uses water pumps for controlling its ballast, there is no possibility of raising it from beneath the surface if the ballast tanks are completely filled with water. As seen in chapter 4.5.1, this would be possible if using compressed air. This method would need separate tanks for storing compressed air, and a compressor for filling them while at surface. If the method were to be used, the height of the platform could then be controlled by separate thrusters, or by having rotatable thrusters like some ROV's have today. This could even be controlled by a system which control the air inlet on and off, creating more or less buoyancy. This method would spend a lot of air, and since an air compressor only works above surface this would only work for as long as the air tanks are pressurised.

5.3 Version control Git

In a project of this scale, a version control of the software has been crucial. The possibility to create separate branches to test new functionality without altering the main code, is a safe and excellent way to develop the software. Without version control, the merging of codes from the group members would be a tedious and difficult task, and the safety of being able to return to previous versions of the software would not be possible.

5.4 Necessary improvements

5.4.1 Buoyancy

The prototype could definitely benefit from some additional buoyancy. As discussed in section [5.1.1](#), the simplest solution for handling this issue would be to replace the battery pontoons with larger ones, and swap the current batteries to lithium batteries.

5.4.2 Motor controllers

During the outdoor testing of the vessel, it was observed that one of the thrusters would shut down after some time. Since the Arduino sends the information it is supposed to, and it is always the same thruster which stops, there reason to believe that the problem is located in the motor controllers. This controller should be changed or inspected before further work on the platform.

5.4.3 Client - Server connection

The communication between client and server works without any trouble and connection is established every time. However, if the operator of the client application wants to switch operation mode, the server has an handling issue. To switch mode, both the client and server must be restarted and then choose the wanted mode.

5.5 Experiences

5.5.1 Size and complexity of the project

One of the main experiences and perhaps the most important one, is that development takes more time than expected in most occasions. When the pre-report was made, the group and supervisors was a bit ambitious on the scale of this project. Making an own design from scratch, develop software and autonomous operation, is a bit much to grasp, even for four students. A project with less objectives would make the function perform better, as the group could spend more time on improvement and tuning.

5.5.2 Project planning

During the pre project, a Gantt form was made to plan each activity and the time to spend on the different aspects of the project. As mentioned earlier, the group experienced that the amount of work was too great in order to finish on time, and the time estimates on some of the activities are not met.

5.5.3 Working as a team

To get the most out of every group member, a workplace based on mutual respect, discipline and high work ethic has been the foundation of this project. As a result, this makes members of the group look forward to meet again day after day to reach the common goal. All members have different backgrounds and experiences, but this has never been an issue, in fact, it has been an benefit for the group as members have been able to learn from each other.

5.6 Possible operations for a semi-submersible USV

As intended, the platform is designed and developed for operations at sea farms, but due to the design's versatility, it is capable of doing other tasks as well.

Since the idea is to have the platform fully autonomous, the vessel can do tedious operations like scanning the sea bed for world map improvements, or scanning the sea for contamination

around an oilfield.

As mentioned in the above section, the platform could, given the correct equipment, also be used as a fully submersible- as well as a semi-submersible vessel. This opens a lot of opportunities while the platform could perform operations beneath the surface as well as above.

Another concept is using several of these platforms and combine them for heavier loads. There could theoretically be hundred platforms combined and controlled using swarm technology. The combination could operate as a barge for transporting heavy equipment, or even as a temporary bridge for vehicles to use.

Chapter 6

Conclusions

The specification of the thesis was to develop a platform designed for handling a ROV attached to a winch. The vessel design should withstand waves better than a regular boat hull considering the operation of the ROV. Active stabilisation must be implemented for the purpose of correcting an angle of list within 30 seconds. The platform is meant to run autonomously but with a manual override functionality.

The platform is designed with a large opening in the bottom reaching up to the deck as seen in figure 4.24. There is room enough for a regular sized ROV to be hoisted in to the center of the platform and it has a buoyancy which allows up to 15 Kg of payload, which is sufficient for the intended equipment. Having the open design, it allows water to run through hull with very little interference. The test results of stability in figure 4.58 supports this statement. The platform withstands waves with shorter wavelengths, but as figure 4.59 indicates, larger wavelengths may cause the platform to oscillate. On the coast, which is the operation area of interest, the platform would have very little oscillations. Considering the design for both ROV operation and wave resistance, the platform is believed to meet the given requirements. The stability system is capable of correcting an angle of list within the specification as seen in table 4.4 and 4.5.

With the developed client application, you can choose between manual- or autopilot mode, which works as intended. The manual mode gives the possibility to manoeuvre the platform in all directions as well as a live feed from the platform using the on board camera. The autopilot mode has a user friendly interface where one can make a list of way-points for the platform to follow. The autopilot works as intended until the vessel starts to handle the incoming sensor

values. The handling of these values has to be tested and adjusted further with the platform in sea, for reaching a desirable behaviour.

A valuable experience from the project is that it is possible to develop sustainable prototypes using only low-cost components. The Arduino is more than capable of handling continuous work, such as the stability system, and the Odroid is powerful enough to operate as the server with all the software threads running at once. An application for a tablet or smart-phone is a good solution as user interface, because the hardware is cheap and the software can be downloaded directly from the internet.

Overall the group agrees on the project being a success. The experience from planning, working as a team, software development, hydrodynamics, and all the other topics are greatly appreciated, and will be a valuable asset in future working careers.

6.1 Further development

In the course of the project, we have had some ideas and thoughts of possible use and changes. Here we have listed the most interesting of them.

- Implement full autonomous system.
- Attach a winch and a ROV.
- Expand client application with more data and functionality.
- Rewrite the server code in such manner that it can handle every client function flawlessly.
- Upgrade stability system with fuzzy logic.
- Use motor controllers to run pumps instead of relays.
- 4G mobile network to eliminate range limits.
- Swarm technology, multiple platforms working together to solve more complex challenges.

Appendices

A Preproject report

B Gantt diagram

C Project A3

D Progress report 26.01.17

E Progress report 10.02.17

F Progress report 24.02.17

G Progress report 24.03.17

H Progress report 02.05.17

I Meeting report 13.01.17

J Meeting report 30.01.17

K Meeting report 10.02.17

L Meeting report 24.03.17

M Meeting report 02.05.17

- N Platform mechanical drawings**
- O Electrical drawing stability system**
- P Electrical drawing thruster control**
- Q Stability calculation Rectangular platform**
- R Stability calculation Hexagon platform**
- S Server source code**
- T Client source code**
- U Arduino source code**

Bibliography

- [1] Meet android studio. <https://developer.android.com/studio/intro/index.html>.
- [2] Thread. <https://docs.oracle.com/javase/7/docs/api/java/lang/Thread.html>.
- [3] European Space Agency. European geostationary navigation overlay service. URL http://www.esa.int/Our_Activities/Navigation/EGNOS/What_is_EGNOS.
- [4] Anda-Olsen-AS. *CP1232 Datasheet*. Anda-Olsen-AS, 1 edition, 1 2003. CP1232 Datasheet.
- [5] Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, The Edinburgh Building, Cambridge, CB2 8RU, UK, 2004.
- [6] Halvor Bothner-By. Tcm – nettverksprotokoll. URL https://snl.no/TCM_-_nettverksprotokoll.
- [7] KAI CHANG. *RF and Microwave Wireless Systems*. JOHN WILEY and SONS, INC., John Wiley and Sons, Inc., 605 Third Avenue, New York, NY 10158-0012, 2000.
- [8] Oracle Java Documentation. Networking basics. URL <https://docs.oracle.com/javase/tutorial/networking/overview/networking.html>.
- [9] Børje Forssell. Global positioning system. URL <https://snl.no/GPS>.
- [10] Thor I. Fossen. Lecture notes ttk 4190 guidance and control of vehicles (t. i. fossen). URL <http://www.fossen.biz/wiley/Ch2.pdf>.
- [11] Thor I. Fossen. *Handbook of Marine Craft Hydrodynamics and Motion Control*. JOHN WILEY and SONS, Ltd., John Wiley and Sons Ltd, The Atrium, Southern Gate, Chichester, West Sussex, PO19 8SQ, United Kingdom, 2011.

- [12] Thor I. Fossen and Tor A. Johansen. A survey of control allocation methods for ships and underwater vehicles. *Mathematical Association of America*, 6(1):1–6, 2006 14th Mediterranean Conference on Control and Automation. doi: <http://ieeexplore.ieee.org/document/4124854/>.
- [13] Rodolfo Giometti. *BeagleBone Essentials*. Packt Publishing, Birmingham, UK, 2015.
- [14] Mark A. Haidekker. *Linear FeedBack Controls - The Essentials*. Elsevier, 32 Jamestown Road, London NW1 7BY, UK, 2013.
- [15] Jakob Sandstad Thor Hansen. antenne – radioteknikk. URL https://snl.no/antenne-_radioteknikk.
- [16] Helge Holden. matematisk programmering. URL https://snl.no/matematisk_programmering.
- [17] Analog Devices Inc. *AD620 Datasheet*. Analog Devices Inc., 1 edition, 1 2003. AD620 Datasheet.
- [18] Motorola Inc. *SPI Block Guide*. Motorola Inc., 2 edition, 2 2003. S12SPIV3/D.
- [19] N. B. Nichols J. G. Zigeler. Optimum settings for automatic controllers. URL <http://chem.engr.utc.edu/Student-files/x2008-Fa/435-Blue/1942-paper.pdf>.
- [20] joptimizer.com. Primal-dual interior-point method. URL <http://www.joptimizer.com/primalDualMethod.html>.
- [21] Rudolph Emil Kalman. A new approach to linear filtering and prediction problems. *Transactions of the ASME–Journal of Basic Engineering*, 82(Series D):35–45, 1960.
- [22] 829 Kandidatnummer: 807, 819. Usv - unmanned surface vessel. *Bachelor oppgave*, 368 (10):74–77, 26.05.2016.
- [23] A.D. King. Inertial navigation, forty years of evolution. URL [link:http://www.imar-navigation.de/downloads/papers/inertial_navigation_introduction.pdf](http://www.imar-navigation.de/downloads/papers/inertial_navigation_introduction.pdf).

- [24] Kongsberg. Imo dp clasification. URL <https://www.km.kongsberg.com/ks/web/nokbg0240.nsf/AllWeb/D9479D5DB35FCA01C1256A4C004A876E?OpenDocument>.
- [25] James F. Kurose and Keith W. Ross. *Computer Networking Sixth Edition*. Pearson, Edinburgh Gate, Harlow CM20 2JE, England, 2013.
- [26] MathWorks. Estimate flat eart position from geodetic latitude, longitude and altitude, mathworks. 2011. URL <http://se.mathworks.com/help/aerotbx/ug/11a2flat.html?requestedDomain=www.mathworks.com>.
- [27] Wolfram Mathworld. Euler angles. URL <link:http://mathworld.wolfram.com/EulerAngles.html>.
- [28] LLC MetaGeek. inssider. URL <http://www.metageek.com/products/inssider/>.
- [29] Oracle. Socket programming. <http://docs.oracle.com/javase/tutorial/networking/sockets/definition.htm>
- [30] PipeLife. Pipelife. URL <http://www.pipelife.no/no/>.
- [31] C. C. Robusto. The cosine-haversine formula. (USA). *Mathematical Association of America*, 64(1):38–40, 1957. doi: <http://www.jstor.org/stable/pdf/2309088.pdf>.
- [32] Eirik Rossen. Usb – it. URL https://snl.no/USB_-_IT.
- [33] Freescale Semiconductor. *MPX2010 series Datasheet*. Freescale Semiconductor, 13 edition, 10 2008. MPX2010 Datasheet.
- [34] Asgeir J. Sørensen. A survey of dynamic positioning control systems. *Annual Reviews in Control*, 35(1):123–136, 2011. doi: <http://www.sciencedirect.com/science/article/pii/S1367578811000095>.
- [35] Texas-Instruments-Inc. *TCA9548A Datasheet*. Texas-Instruments-Inc., 2 edition, 1 2015. TCA9548A Datasheet.
- [36] tp link.com. How to improve my wireless speed or range? URL <http://www.tp-link.com/us/faq-468.html>.

- [37] Andy Wellings. *Concurrent and Real-Time Programming in Java*. JOHN WILEY and SONS, Ltd., John Wiley and Sons, Ltd., The atrium, Southern Gate, Chichester, West Sussex PO19 8SQ, England, 2004.

TITLE:

Autonomous platform

CANDIDATES (NAMES):

Christer Bakken, Per Martin Leinan, Matias Heggen, Martin Blom

DATE:

16.01.17

SUBJECT CODE:

IE303612

SUBJECT:

Bachelor thesis

DOCUMENT ACCESS:

- Open

STUDIUM:

AUTOMATION TECHNIQUE

NR PAGES/APPENDIX:

/

BIBL. NR:

- Not in use -

PRINCIPALS/SUPERVISORS

Ottar Osen, Houxiang Zhang

SUMMARY:

CONTENTS

1 INTRODUCTION	3
2 NOTION	3
3 PROJECT ORGANIZATION	3
3.1 PROJECT GROUP	3
3.2 SUPERVISORS(VIELEDER OG KONTAKTPERSON OPPDRAGSGIVER)	4
4 AGREEMENTS	4
4.1 AGREEMENTS WITH PRINCIPAL SUPERVISORS	4
4.2 WORKPLACE AND RESOURCES	4
4.3 GROUP NORMS— AGREEMENT ON CO-OPERATION— ATTITUDES	4
5 PROJECT DESCRIPTION	4
5.1 PROBLEM- GOAL- PURPOSE	4
5.2 SPECIFICATIONS AND DEMANDS OF END RESULT	4
5.3 SCHEDULED PROGRESS FOR DEVELOPMENT- METHOD(S)	4
5.4 INFORMATION COLLECTION	5
5.5 RISK ANALYSIS	5
5.6 MAIN ACTIVITIES	5
5.7 PROGRESS MANAGEMENT	5
5.8 DECISION-MAKING PROCESS	6
6 DOCUMENTATION	6
6.1 REPORTS AND TECHNICAL DOCUMENTS	6
7 PLANNED MEETINGS AND REPORTS	6
7.1 MEETINGS	6
7.2 PERIODIC REPORTS	6
8 TREATMENT OF NONCONFORMANCE	6
9 EQUIPMENT REQUIREMENTS / CONDITIONS FOR IMPLEMENTATION	7
10 REFERENCES	7
APPENDIX	

1 INTRODUCTION

In maritime industry there are numerous operations that could be improved by using smart systems. Operations such as survey, inspection, sensor and hoisting are examples that needs positioning control and stability. In addition, acquisition and operating cost are among the most important factors when new systems are being accessed.

A solution for/to this could be an autonomous platform with the capability of self - operating, DP1 and active stabilization. Equipped with different types of sensors, there are no restriction to what operations a system like this could perform. Not only would the efficiency be increased, but also the safety, since there are no operator on the platform itself. This makes it versatile, thus more cost efficient.

2 NOTION

- GUI: Graphical User Interface
- GPS: Global Positioning System
- DP1: Dynamic Positioning System. A standard from Germanischer Lloyd that describes the requirements of the system. DP1 has no redundancy.

3 PROJECT ORGANIZATION

Project members

Per Martin Leinan (Chief executive)
Christer Bakken
Matias Heggen (Report supervisor)
Martin Blom

3.1.1 Project tasks - organization

Platform design (Primary)

- Martin, with assistance from Per Martin

Stabilization (Primary)

- Martin, with assistance from Per Martin

Autopilot (Primary)

- Christer, with assistance from Matias

Dynamic positioning (Primary)

- Christer, with assistance from Matias

Autonomous (Primary)

- Per Martin, with assistance from Martin

Communication (Primary)

- Matias, with assistance from Christer

GUI (Primary)

- Matias, with assistance from Christer

3.1.2 Project leader responsibilities

- Convene meetings with bachelor supervisors every second friday
- Update progress reports

3.1.3 Secretary responsibilities

- Log activities and progress every week
- Write meeting report

3.1.4 Other group members responsibilities

- All members are responsible for continuous update of documentation and main report.

Supervisors (veileder og kontaktperson oppdragsgiver)

Houxiang Zhang and Ottar L. Osen

4 AGREEMENTS

Agreement with supervisors

From the first meeting between students and supervisors, several things were discussed considering the size and use of time in the project. An agreement was made that the students grasp a wide spectrum of functionalities, four or five, and do some research whether it is possible to achieve all or not. The supervisors made it clear that a final result with 2-3 of this functionalities would be sufficient.

Workplace and resources

NTNU Ålesund

Group Norms - Agreement on cooperation - Attitudes

The workdays will be as in any normal company where work hours and place is accordingly. A normal day of work is 0800 to 1600 and work place is NTNU Ålesund. The group has to be notified if there are deviations from this.

Respect for each other. We are all working toward a common goal and we are dependent of each other. The product we develop is a proof of what we are able to create together as a group and what knowledge from our studies we are able to use in a real life project.

5 PROJECT DESCRIPTION

Problem - goal - purpose

For the bachelor thesis, the main goal is to make the concept idea of an autonomous platform to a real life application.

The final product is to be an autonomous platform with DP capabilities and self stabilizing system for solving a vary of general purposes.

The platform can also be equipped with accessories that fits several types of maritime operations. Things to take in consideration is:

1. Make a design to fit several types of applications
2. Decide what sensors are necessary to reach final result
3. How will the design behave in different types of water conditions

Specifications and demands of end result

The platform will be a prototype, capable of doing numerous autonomous operations.

Runtime for a minimum of two hours in transport-mode, this means two of the thrusters run on max capacity for two hours straight.

Routing is done by the GUI, plotting a course for the platform to run, inspect or other. A manual override safe-mode will be implemented with an RC-controller on its own frequency spectrum.

A winch will be mounted on top of the platform for handling subsea equipment, the load capacity of this winch will be limited by the carrying capacity of the platform.

Dynamic positioning is one of the specifications needed for the platform to be versatile, to aid equipment for accuracy.

Instruments implemented would be, sonar for observing and logging the sea bottom as well as giving the platform information on depth, thermistors for logging sea temperature, GPS for route navigation, 360 degree laser sensor for detecting obstacles and an accelerometer for stabilization and direction.

Thrusters has to meet requirements for use in saltwater, the natural selection is ROV thrusters since they are already dimensioned for this specific use. There will be four thrusters, giving the opportunity to maneuver in all directions, this will help the DP functionality tremendously.

A 360 degree camera for remote platform maneuvering.

Scheduled progress for development - method(s)

The scheduled progression-plan for the bachelor-thesis:

Lean project management is used for planning and executing the project. The method used for this is Deming cycle, also known as A3.

The benefit of this method is that the progress is reevaluated continuously.

This gives the opportunity to look at the problem from different angles.

One document explains the whole problem.

A disadvantage with the A3 method can be time consumption.

Information gathering

Earlier bachelor thesis and projects regarding autonomous systems will be used as background information. Furthermore, companies as Rolls Royce and Sintef are currently working on autonomous water vehicles, and will be contacted for general information and guidance.

Two of the students in the group worked on a platform project in Introduction to Mechatronics and will bring a lot of valuable experience into the project.

Risk analysis

Probability

Very High	Single day absence	Software issues	Hardware issues	
High			Health/illness	Miscalculations Time issues
Medium				
Low			Bad communication	
Very Low				
	Very Low	Low	High	Very High

Impact

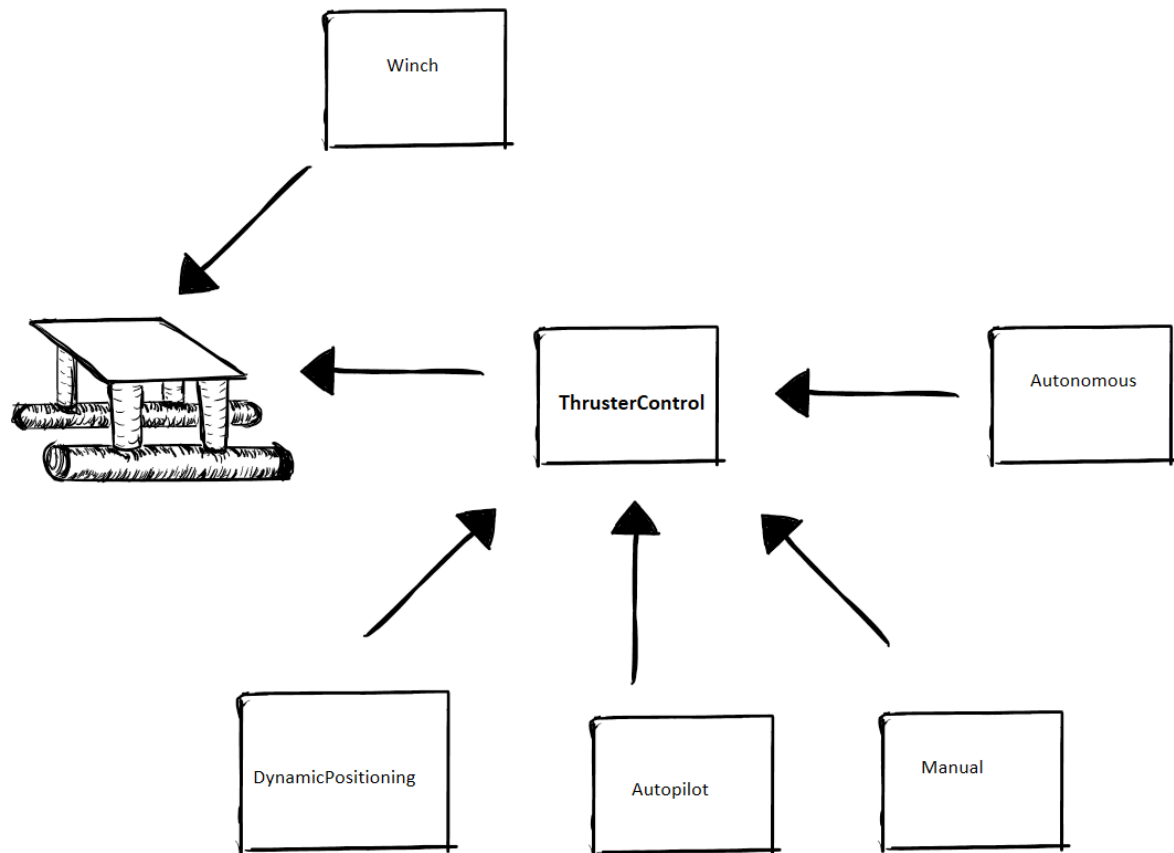
Issues like single day absence and illness are things that are difficult to anticipate. But they can result in bad communication, if the work is not properly documented continuously.

If illness occur among group members, another member must be able to continue the work. The consequence of stalling the project because only one student, will have massive impact concerning time issues. Ways to prevent this is to document the progress properly and work in pairs to keep the workflow running.

Time issues are best solved by proper planning and continuously supplementing the report.

Hardware and software issues will occur. To avoid spending unnecessary time on the issues, we will have to do proper research, cooperate on problems within the group, and use our supervisors or other persons with high competence within the specific field.

To minimize the effect of one software component failing, the different software components are independent. i.e. If the autonomous system fails, the dynamic positioning software will remain functional.



Main activities

Nr	Main activity	Responsibility	Cost	Time/scope
A1	Platform design	Martin/Per	15600,-	26.01.17 - 20.02.17
A2	Autopilot	Christer/Matias	2000,-	26.01.17 - 06.04.17
A3	Dynamic positioning	Christer/Matias	0,-	26.01.17 - 06.04.17
A4	Autonomous	Per/Martin	10000,-	21.02.17 - 07.04.17
A5	Communication	Matias/Christer	3000,-	26.01.17 - 01.03.17
A6	GUI	Matias/Christer	5000,-	26.01.17 - 01.03.17
A7	Active stabilization	Per/Martin	2000,-	26.01.17 - 20.02.17

Progress management

5.7.1 Master plan

Platform design (Primary)

- With only a concept idea of what the platform should look like, it has to be made specifications for what it is supposed to do and calculations given its requirements. The platforms requirements considering payload lays the groundwork for buoyancy needs.

Start date: 26.01.17

Estimated date of completion: 26.01.17

- When this requirements is definite, all things considered, there will be made several sketches of the platform with different designs. All the designs will be run i 3D simulator and from there we will use the prefered model.

- Draw Prototypes

Start date: 27.01.17

Estimated date of completion:30.01.17

- Calculate mass and center of gravity

Start date: 27.01.17

Estimated date of completion: 30.01.17

- 3D simulation

Start date: 30.01.17

Estimated date of completion: 1.02.17

- When the prefered model is identified, the platform will be, as much as possible, built in 3D before ordering parts. This helps detecting build issues for eliminating them before the actual build. To keep time spent on building and modifying parts as low as possible, all parts will be ordered as is on drawing.

Start date: 30.01.17

Estimated date of completion: 31.01.17

- Ordering of parts for the platform.

Start date: 01.02.17

Estimated date of completion: 14.02.17

- Building the platform

Start date: 15.02.17

Estimated date of completion: 20.02.17

Autopilot (Primary)

- For the autopilot and the DP part of the project, a report considering these issues is at hand and has to be read through carefully. The plan is to improve the result from the earlier group and not redo the mistakes of the project.

Start date: 26.01.17

Estimated date of completion: 02.02.17

- Get to know the GPS and how it works on Odroid. An issue to be solved is to get a communication between NetBeans and the GPS itself.

Start date: 03.02.17

Estimated date of completion:08.02.17

- A GUI with a map and possibility to insert coordinates will be our solution to make the platform find the wanted waypoints.
Start date: 09.02.17
Estimated date of completion: 15.02.17
- Create final software
Start date: 16.02.17
Estimated date of completion: 28.03.17
- Implement on platform
Start date: 29.03.17
Estimated date of completion: 03.04.17
- Test system
Start date: 04.04.17
Estimated date of completion: 06.04.17

Dynamic positioning (Primary)

- The DP will be based on the autopilot system. A program that identifies if there is a offset from the wanted position to the actual position of the platform will regulate this accordingly. Information collection as first step.
Start date: 26.01.17
Estimated date of completion: 02.02.17
- Create software for testing the functionality
Start date: 07.04.17
Estimated date of completion: 09.05.17
- Implement system to main software
Start date: 10.05.17
Estimated date of completion: 10.05.17
- Test DP on the platform
Start date: 11.05.17
Estimated date of completion: 16.05.17

Autonomous (Primary)

- To make the platform autonomous it has get to the waypoints and maneuver through obstacles which is not seen by the GPS. For this task the platform will be equipped with some sort of sensor for detecting obstacles.
- The first task is to find the desirable sensors and learn how it's used.
Start date: 21.02.17
Estimated date of completion: 27.02.17
- Order sensor and other needed equipment for the task.
Start date: 28.02.17
Estimated date of completion: 23.03.17
- Create software for sensors with I/O for use in main software.
Start date: 28.02.17
Estimated date of completion: 07.04.17

Communication (Primary)

- The communication between server (operator computer) and client (platform) has a required distance of 200m. The primary communication link will be through WiFi, and an optional would be to use a lower frequency radio controller. WiFi will provide enough bandwidth for camera feed and GPS plots. An appropriate secondary solution is to use a Radio Controller with a lower operating frequency than WiFi in order to eliminate interference. A Radio Controller with a lower operating frequency will naturally have a farther range.

- In order to meet our requirements we will have to do necessary research and information collection.

Start date: 26.01.17

Estimated date of completion: 02.02.17

- Decide transmission protocol, UDP or TCP, and get familiar with programming sockets in java language according to preferred protocol choice.

Start date: 31.01.17

Estimated date of completion: 07.02.17

- Establish communication over WiFi and send data packets with relevant information to what we will be transmitting on the final product.

Start date: 08.02.17

Estimated date of completion: 15.02.17

- Implement into main code

Start date: 16.02.17

Estimated date of completion: 28.03.17

GUI (Primary)

- The user interface for the platform would consist of a program on the computer, able to override the platform maneuvering as well as showing live stream from the onboard camera. The program displays a map, on the map you would see the platforms current location as well as the possibility of setting a new waypoint or route for it to go. Designing a GUI consist of several phases.

- Gather information and knowledge about how to make a GUI.

Start date: 26.01.17

Estimated date of completion: 01.02.17

- Make a simple GUI and do some testing in order to get familiar with concepts like screen scaling.

Start date: 30.01.17

Estimated date of completion: 03.02.17

- Make a visual sketch to use as a guide for the layout

Start date: 06.02.17

Estimated date of completion: 07.02.17

- Start developing a final layout and bind keyboard buttons to the visual buttons in the GUI

Start date: 08.02.17

Estimated date of completion: 22.02.17

- Merge the GUI with functional code.

Start date: 23.02.17

Estimated date of completion: 01.03.17

Stabilization (Primary)

- To stabilize the platform it would need some way to move ballast from one tank to another. This would be controlled by the onboard accelerometer and a water pump system. The Ballast tanks itself will be designed along with the main platform design.

Start date: Along with platform design.

Estimated date of completion:

- The system will use water for stabilization, a water pump with valves for controllability is used in the idea. Which water pump to use, what extra equipment is needed and how to implement it to a whole system is what to figure out as of this step.

- **Start date: 26.01.17**

- **Estimated date of completion:26.01.17**

- Order parts

- **Start date: 03.02.17**

- **Estimated date of completion:03.02.17**

- Build complete system for testing before mounting on platform, create software for testing

- **Start date: 06.02.17**

- **Estimated date of completion:14.02.17**

- Mount system on platform

- **Start date: 15.02.17**

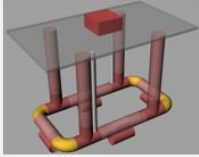
- **Estimated date of completion:20.02.17**

5.7.2 Project management tools

Gantt diagram, deming cycle using the A3 problem solving method.

Problem: Platform design

1) Background/current situation:
We have a concept idea and a prototype which has to be improved.



2) New desirable situation:
Main goal: To have a design that can be used in several types of applications.

- It has to be naturally stable
- Maneuverability in all four directions without needing to turn.
- Has to be able to pivot.

3) Analyse (Det helhetlige kunnskapssystemet)
Since our task is regarding automation, the design part of the project falls outside the automation curriculum. Therefore we need assistance for this part by experienced design engineers in order to free time for our main topic of the bachelor.

Requirements:

- The platform should be able to run for a minimum of 2 hours.
- Active stabilization must be taken in consideration.
- Top section for mounting equipment
- Minimum payload of ---
- The Haswing 20 thruster use 17A at maximum thrust
- 2 thrusters with 17A gives 34A, other electronics, 6A, total of 40A. Two hours of operation need at least 80Ah battery.
- With 18650 batteries, this results in 5.2kg of battery pack.
- Li-ion because of low voltage drop over time.

Safety:

- The platform has to be naturally stable.
- Sustainable materials.
- Thrusters must be encapsulated.

4) Aksjon: Valg av tiltak

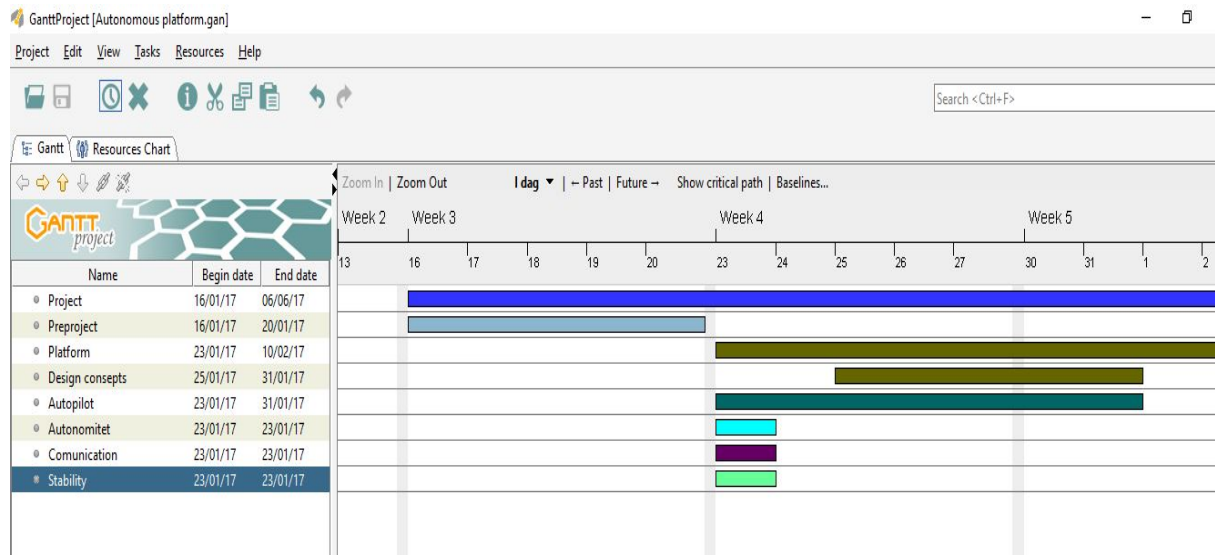
- Get assistance for drawing
- Design as part of bachelor thesis.
- Run the finished design in a simulator.
- Use Li-ion batteries for both ballast and operation time.

5. Hva Tiltak

Hva Tiltak	Hvordan gjennomføre?	Hvor?	Hvem er ansvarlig?	Når?

6. Evaluering, standardisering 7 og læring (8)

Picture: A3 form



Picture: Gantt diagram

5.7.3 Development tools

In order to complete the project a wide array of tools are required, such as 3D design tools, simulation tools, material to create a functional prototype of the design, wave simulation tank, sensors, thrusters, gps, computers, microcontrollers, proper software for software development, mobile network route.

5.7.4 Internal control - evaluation

The follow-up process of the progress is based on a gantt table, deming cycle(A3 model),daily and weekly meetings.

Decision-making process

The decisions made in this preliminary report has been done in unity.
The parts where we had different opinions the decisions were made by a poll.

6 DOCUMENTATION

Reports and technical documents

- Weekly update on the main report, all work done on project will be documented in this report.
- All technical documents and data sheet on equipment used is uploaded to the group folder on Office 365.
This way, all documentation is easily accessed for later use and for completion of the main report.

7 PLANNED MEETINGS AND REPORTS

Meetings

7.1.1 Meeting with supervisors

Meetings with the supervisors are scheduled to every second Friday where progress is presented.

7.1.2 Project meetings

Every day will be started off with a quick brief of what needs to be done, unexpected challenges and a review of the schedule.

Periodic reports

7.2.1 Progress reports(incl. milestones)

Progress reports will be updated continuously, and presented during the meetings with the supervisors.

8 TREATMENT OF NONCONFORMANCE

If anyone is struggling with their task for more than a day without progress, brainstorm with his assistant co-worker to try and find a solution. If another day passes without progress, talk with the rest of the group or one of our supervisors. Everyone is responsible for their own task, and to seek aid if needed.

9 EQUIPMENT REQUIREMENTS / CONDITIONS FOR IMPLEMENTATION

- Welding equipment for PE tubes (located at the school's workshop)

10 REFERENCES

APPENDIX

Appendix 1	Gantt diagram
Appendix 2	A3
Appendix 3	Budget

Bachelor thesis Autonomous platform

20.jan.2017

NTNU Aalesund

Prosjektleder

Per Martin Leinan

Prosjekt start/slutt datoer

16.jan.2017 - 22.jun.2017

Avsluttet

0%

Oppgave

45

Deltakere

4

Oppgave

2

Navn	Startdato	Sluttdato
Project	16.01.17	21.06.17
Preproject	16.01.17	20.01.17
Platform	26.01.17	20.02.17
Platform requirements	26.01.17	26.01.17
Design concepts	27.01.17	31.01.17
Drawing prototypes	27.01.17	30.01.17
Calculate mass and center of gravity	27.01.17	27.01.17
3D simulation	27.01.17	31.01.17
Build in 3D	30.01.17	31.01.17
Order parts	01.02.17	14.02.17
Build platform	15.02.17	20.02.17
Stability	26.01.17	20.02.17
Design ballast tank system	26.01.17	27.01.17
Find desirable equipment	01.02.17	03.02.17
Order parts	03.02.17	03.02.17
Build system for testing	06.02.17	14.02.17
Mount system on platform	15.02.17	20.02.17
Autonomitet	21.02.17	07.04.17
Information collection	21.02.17	27.02.17
Order sensors	28.02.17	23.03.17
Create software with I/O for use in main software	28.02.17	07.04.17
Autopilot	26.01.17	06.04.17
Information collection	26.01.17	02.02.17
Communication from GPS to Odroid	03.02.17	08.02.17
Create GUI for testing	09.02.17	15.02.17

Oppgave

3

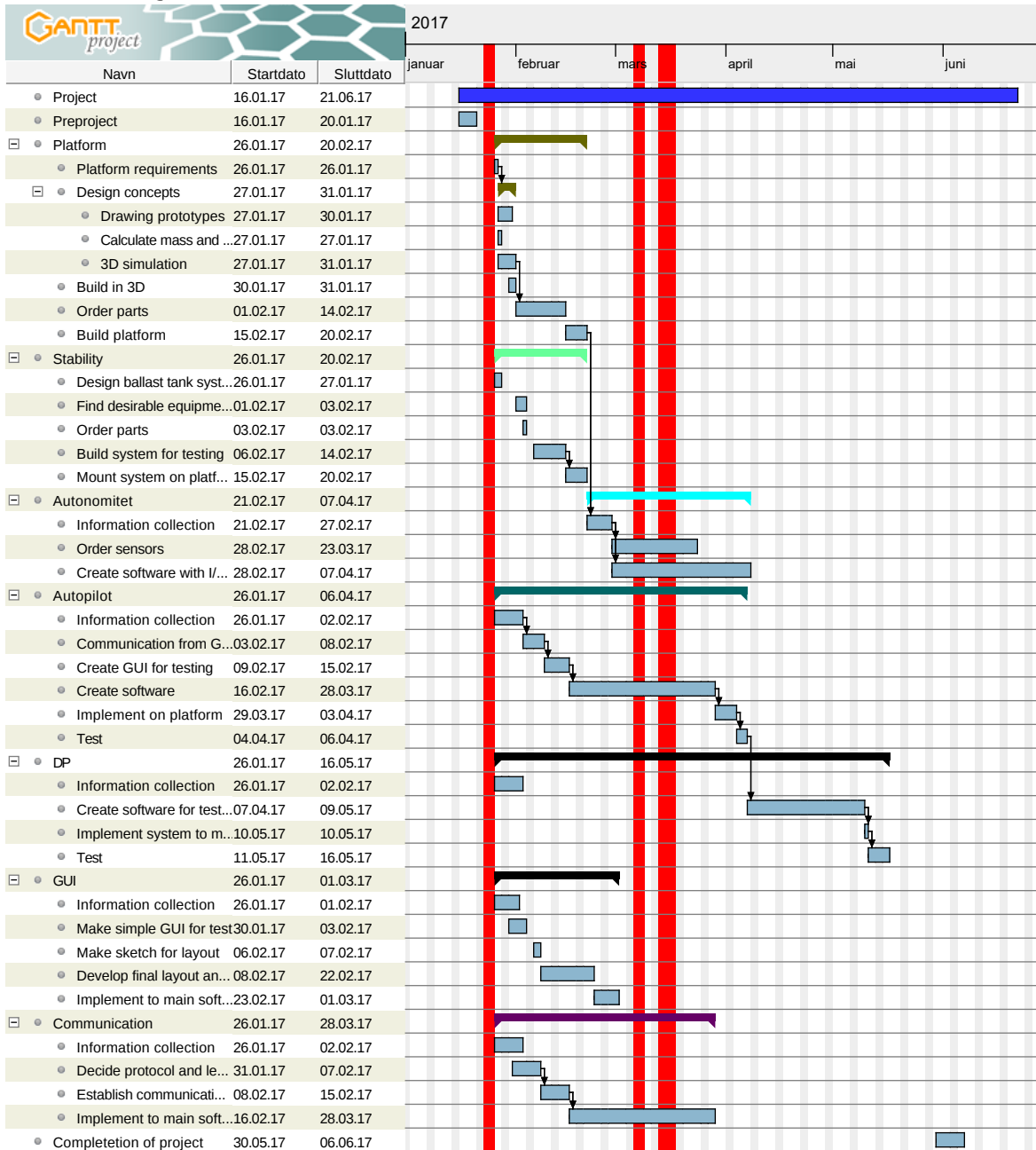
Navn	Startdato	Sluttdato
Create software	16.02.17	28.03.17
Implement on platform	29.03.17	03.04.17
Test	04.04.17	06.04.17
DP	26.01.17	16.05.17
Information collection	26.01.17	02.02.17
Create software for testing	07.04.17	09.05.17
Implement system to main software	10.05.17	10.05.17
Test	11.05.17	16.05.17
GUI	26.01.17	01.03.17
Information collection	26.01.17	01.02.17
Make simple GUI for test	30.01.17	03.02.17
Make sketch for layout	06.02.17	07.02.17
Develop final layout and functionality	08.02.17	22.02.17
Implement to main software	23.02.17	01.03.17
Communication	26.01.17	28.03.17
Information collection	26.01.17	02.02.17
Decide protocol and learn protocol	31.01.17	07.02.17
Establish communication over wifi	08.02.17	15.02.17
Implement to main software	16.02.17	28.03.17
Completion of project	30.05.17	06.06.17

Deltakere

4

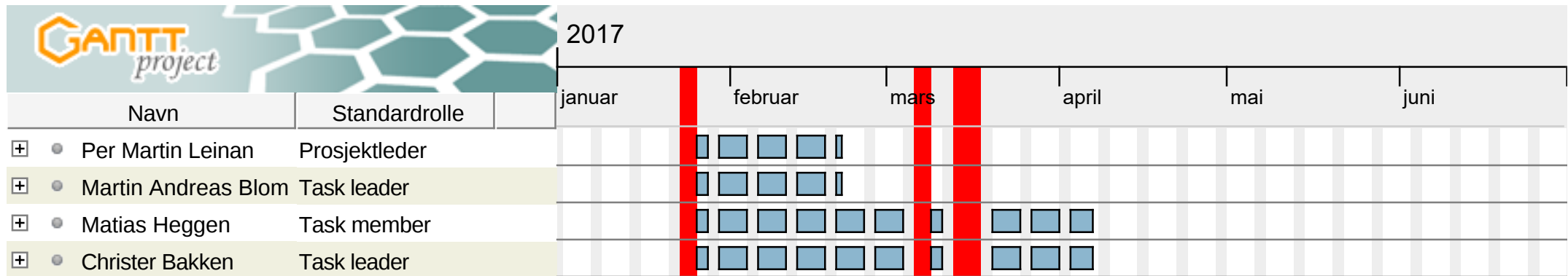
Navn	Standardrolle
Per Martin Leinan	Prosjektleder
Martin Andreas Blom	Task leader
Matias Heggen	Task member
Christer Bakken	Task leader

Gantt-skjema



Personelloversikt

6



Theme: Stabilization

1) Bakgrunn/Nå-situasjon:

- Without active stabilization, the platform could tilt sideways due to drag or wind.

2) Ny ønsket situasjon:

- Stabilized platform**
We want a stable platform for enhancing accuracy of tools and sensors.

3) Analyse (Det helhetlige kunnskapssystemet)

- Which environment is the platform supposed to operate in?
 - In sheltered waters
 - In lakes
- Why do we need active stabilization?
 - Sensor
 - Payload
- What challenges might occur?
 - Transport
 - Ocean currents
 - Wind
 - Payload
- Safety
 - The system can not be able to tip the platform.

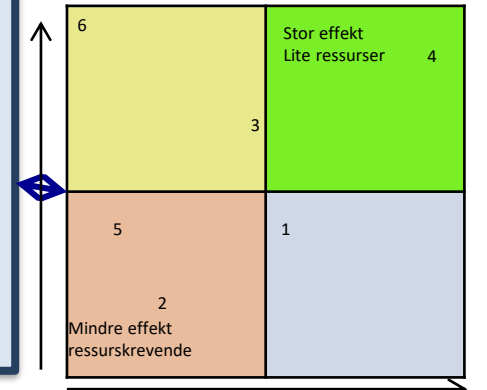
A3 Eier: Bachelorgruppe Autonom Plattform

Dato: 11.01.17

4) Aksjon: Valg av tiltak

- Modular
- Foils
- Thrusters
- Water tanks and pump systems
- Linear actuators
- Gyro

Stor Effekt



Lite ressurser

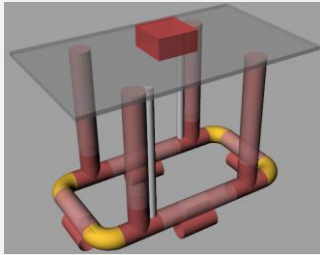
5. Hva Tiltak	Hvordan gjennomføre?	Hvor?	Hvem er ansvarlig?	Når?
4	Main project	NTNU	Group platform	Now – 06.06

6. Evaluering, standardisering 7 og læring (8)

Problem: Platform design

1) Background/current situation:

We have a concept idea and a prototype which has to be improved.



2) New desirable situation:

Main goal: To have a design that can be used in several types of applications.

- It has to be naturally stable
- Maneuverability in every direction on the surface without having to turn
- Has to be able to pivot.

3) Analyse (Det helhetlige kunnskapssystemet)

Requirements:

- The platform should be able to run for a minimum of 2 hours.
- Active stabilization must be taken in consideration.
- Top section for mounting equipment
- Minimum payload of ---
- The Haswing 20 thruster use 17A at maximum thrust
- 2 thrusters with 17A gives 34A, other electronics, 6A, total of 40A. Two hours of operation need at least 80Ah battery.
- With 18650 batteries, this results in 5.2kg of battery pack.
- Li-ion because of low voltage drop over time.

Safety:

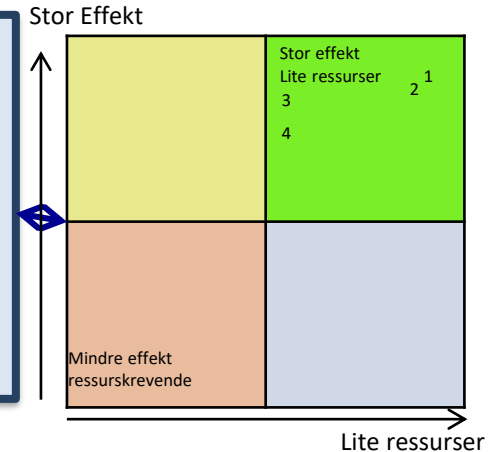
- The platform has to be naturally stable.
- Sustainable materials.
- Thrusters must be encapsulated.

A3 Eier: Bachelorgruppe Autonom Plattform

Dato: 11.01.17

4) Aksjon: Valg av tiltak

1. Make several concept designs
2. Run the finished designs in a simulator.
3. Use Li-ion batteries for both ballast and operation time.
4. Compare suitable materials, such as: Aluminium, Plastic, Plastic and Aluminium, Steel and Aluminium, Carbon fiber etc.



5. Hva Tiltak	Hvordan gjennomføre?	Hvor?	Hvem er ansvarlig?	Når?
1	Main project	NTNU	Group platform	Now – 06.06
2	Main project	NTNU	Group platform	Now – 06.06
3	Main project	NTNU	Group platform	Now – 06.06
4	Main project	NTNU	Group platform	Now – 06.06

6. Evaluering, standardisering 7 og læring (8)

Solutions for autopilot using GPS

1) Bakgrunn/Nå-situasjon:

We need solutions for autopilot for the watercraft.
We have a partially autonomous watercraft developed by another group earlier.

2) Ny ønsket situasjon:

A stable solution for implementing autonomous operation of the watercraft.

The watercraft has to be able to maneuver through sea towards a given target by itself.

3) Analyse (Det helhetlige kunnskapssystemet)

Requirements:

- The accuracy of the GPS
 - +/-1 meter
- The GPS have to have mean value functionality
- Wind and sea current compensation
 - PID controller
- Has to follow a track to waypoint.
- Accelerometer for direction

Safety:

- GPS loose it's signal, what happens?
- Safety protocol for GPS signal loss.
 - Manual override
 - Emergency stop

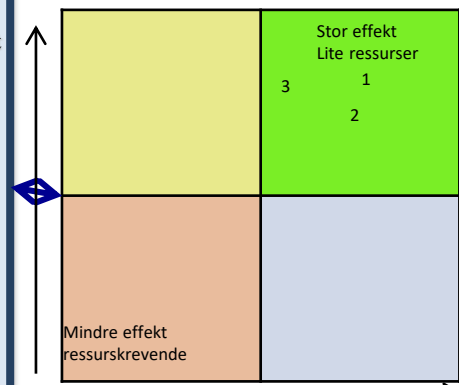
A3 Eier: Bachelorgruppe Autonom Plattform

Dato: 11.01.17

4) Aksjon: Valg av tiltak

1. Compare GPS: Arduino against GPS: USB
2. Find suitable Accelerometer
3. Design a PID controller

Stor Effekt



Lite ressurser

5. Hva Tiltak	Hvordan gjennomføre?	Hvor?	Hvem er ansvarlig?	Når?
1	Main project	NTNU	Group	Now – 06.06
2	Main project	NTNU	Group	Now – 06.06
3	Main project	NTNU	Group	Now – 06.06



6. Evaluering, standardisering 7 og læring (8)

Solutions for autonomous operations

1) Bakgrunn/Nå-situasjon:

The GPS can not see everything around the watercraft itself.

2) Ny ønsket situasjon:

A stable solution for implementing autonomous operation

The vessel has to be able to manouvre through sea towards a given target by itself.
Foreign obstacles has to be taken in consideration.

3) Analyse (Det helhetlige kunnskapssystemet)

- What to detect and avoid
 - Small boats without AIS
 - Kayaks
 - In general foreign obstacles
 - Shallow waters
- How to react meeting obstacles
 - The autonomous system should plan a new route around the obstacle in the way, always keeping a clear distance to other boats.
- Safety
- Mimimum distance to obstacles
- Visibility

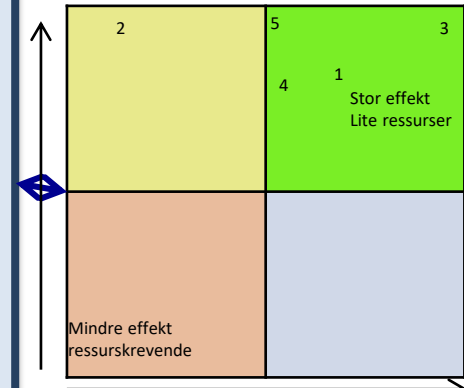
A3 Eier: Bachelorgruppe Autonom Plattform

Dato: 11.01.17

4) Aksjon: Valg av tiltak

- Design software as a concurrent program
- Automatic Identification System (AIS)
- Attach a lantern for visibility
- Depth sensor
- Design sensor system to detect surface obstacles

Stor Effekt



Lite ressurser

5. Hva Tiltak	Hvordan gjennomføre?	Hvor?	Hvem er ansvarlig?	Når?
1	Main Project	NTNU	Group	Now -06.06
2	Sub Project	NTNU	Group	Now -06.06
3	Main Project	NTNU	Group	Now -06.06
4	Main Project	NTNU	Group	Now -06.06
5	Main Project	NTNU	Group	Now -06.06



6. Evaluering, standardisering 7 og læring (8)

Wireless communication and GUI from platform to land

1) Bakgrunn/Nå-situasjon:

We have a bluetooth system for operating the vehicle.

2) Ny ønsket situasjon:

We want a wireless communication with a minimum range of 100m
The GUI to control system.

3) Analyse (Det helhetlige kunnskapssystemet)

The communication distance can not be a problem because of the vast distances the platform is operating in.

- Bandwidth?
 - Video stream
 - Realtime feed

Safety

- Connection issues
 - Distance, weather, operation frequency
 - Redundancy

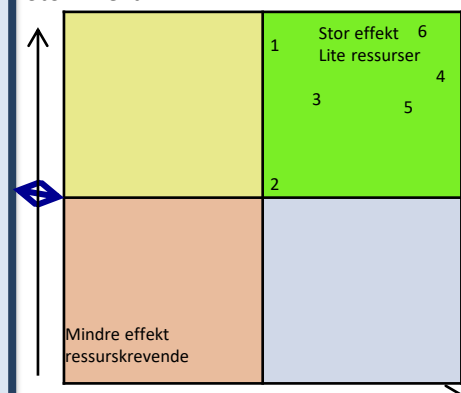
A3 Eier: Bachelorgruppe Autonom Plattform

Dato: 11.01.17

4) Aksjon: Valg av tiltak

- Design GUI in JAVA with necessary functions to control the platform
- Find GUI software - Reverse engineering
- Compare communication systems, WIFI, Bluetooth, VHF, cellular network etc.
- Compare network protocols
- Redundant manual control system
- Camera feed

Stor Effekt



Lite ressurser

5. Hva Tiltak	Hvordan gjennomføre?	Hvor?	Hvem er ansvarlig?	Når?
1	Main Project	NTNU	Group GUI	Now – 06.06
3	Main Project	NTNU	Group GUI	Now – 06.06
4	Main Project	NTNU	Group GUI	Now – 06.06
5	Main Project	NTNU	Group GUI	Now – 06.06
6	Main Project	NTNU	Group GUI	Now – 06.06

6. Evaluering, standardisering 7 og læring (8)

Payload

1) Bakgrunn/Nå-situasjon:
We have no specific task for the platform

1) Bakgrunn/Nå-situasjon:
We have no specific task for the platform

2) Ny ønsket situasjon:
We want an instrument to show versatility for the platform.

The system has to be complete and ready to implement on the vehicle.

2) Ny ønsket situasjon:
We want an instrument to show versatility for the platform.

The system has to be complete and ready to implement on the vehicle.

3) Analyse (Det helhetlige kunnskapssystemet)

The payload can be a variety of components, such as:

- Sensors
- Winch

Sensors will be external sensors for gathering data of the environment.

The winch has to be designed in such manner that the platform will retain its stability.

Safety

Point of gravity issues

Overload systems

- The payload can be a variety of components, such as:
- Sensors
 - Winch
- Sensors will be external sensors for gathering data of the environment.
- The winch has to be designed in such manner that the platform will retain its stability.
- Safety
- Point of gravity issues
- Overload systems

The payload can be a variety of components, such as:

- Sensors
- Winch

Sensors will be external sensors for gathering data of the environment.

The winch has to be designed in such manner that the platform will retain its stability.

Safety

Point of gravity issues

Overload systems

The payload can be a variety of components, such as:

- Sensors
- Winch

Sensors will be external sensors for gathering data of the environment.

The winch has to be designed in such manner that the platform will retain its stability.

Safety

Point of gravity issues

Overload systems

The payload can be a variety of components, such as:

- Sensors
- Winch

Sensors will be external sensors for gathering data of the environment.

The winch has to be designed in such manner that the platform will retain its stability.

Safety

Point of gravity issues

Overload systems

The payload can be a variety of components, such as:

- Sensors
- Winch

Sensors will be external sensors for gathering data of the environment.

The winch has to be designed in such manner that the platform will retain its stability.

Safety

Point of gravity issues

Overload systems

The payload can be a variety of components, such as:

- Sensors
- Winch

Sensors will be external sensors for gathering data of the environment.

The winch has to be designed in such manner that the platform will retain its stability.

Safety

Point of gravity issues

Overload systems

A3 Eier: Bachelorgruppe Autonom Plattform
Dato: 11.01.17

4) Aksjon: Valg av tiltak

1. Buy a winch
2. Design and build a winch
3. Attach a sonar
4. Sensor package for plug and play systems
5. 360 degree camera

Stor Effekt

Lite ressurser

5. Hva Tiltak	Hvordan gjennomføre?	Hvor?	Hvem er ansvarlig?	Når?
2	sub project	NTNU	Group Plaform	Now – 06.06
3	sub project	NTNU	Group Platform	Now – 06.06
5	sub project	NTNU	Group GUI	Now – 06.06

6. Evaluering, standardisering 7 og læring (8)

A3 Eier: Bachelorgruppe Autonom Plattform
Dato: 11.01.17

4) Aksjon: Valg av tiltak

1. Buy a winch
2. Design and build a winch
3. Attach a sonar
4. Sensor package for plug and play systems
5. 360 degree camera

Stor Effekt

Lite ressurser

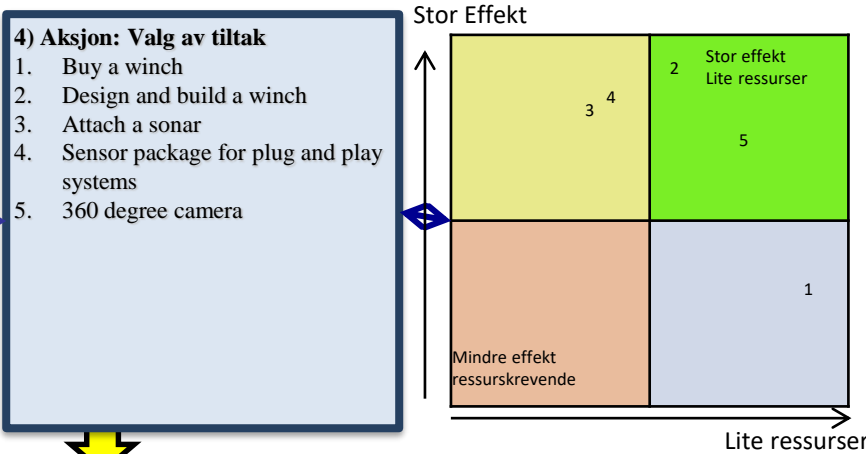
5. Hva Tiltak	Hvordan gjennomføre?	Hvor?	Hvem er ansvarlig?	Når?
2	sub project	NTNU	Group Plaform	Now – 06.06
3	sub project	NTNU	Group Platform	Now – 06.06
5	sub project	NTNU	Group GUI	Now – 06.06

6. Evaluering, standardisering 7 og læring (8)

4) Aksjon: Valg av tiltak

1. Buy a winch
2. Design and build a winch
3. Attach a sonar
4. Sensor package for plug and play systems
5. 360 degree camera

- #### 4) Aksjon: Valg av tiltak
1. Buy a winch
 2. Design and build a winch
 3. Attach a sonar
 4. Sensor package for plug and play systems
 5. 360 degree camera



5. Hva Tiltak	Hvordan gjennomføre?	Hvor?	Hvem er ansvarlig?	Når?
2	sub project	NTNU	Group Platform	Now – 06.06
3	sub project	NTNU	Group Platform	Now – 06.06
5	sub project	NTNU	Group GUI	Now – 06.06

6. Evaluering, standardisering 7 og læring (8)

BUDGET BACHELOR THESIS

Cost related to different activities		2017	SUM
Activity 1 - Platform main structure		15 600	15 600
Activity 2 - Communication		3 000	3 000
Activity 3 - Autonomous		10 000	10 000
Activity 4 - Autopilot		2 000	2 000
Activity 5 - Active stabilization		2 000	2 000
Activity 6 - GUI		5 000	5 000
Activity 7 - Dynamic positioning		0	0
SUM		30 600	37 600

Cost plan	Pcs	Price/pcs	SUM
Pipe: PE100 SDR41 160mm, 6m	2	500	1 000
Rørkrage: PE 100 SDR41 160m	12	50	600
Monteringsplate/Monteringsmateriell:	1	8 000	8 000
18650 Battery pack	1	6 000	6 000
Rotasjons sensor	1	10 000	10 000
Kommunikasjonsutstyr	1	3 000	3 000
Stabilisering	1	2 000	2 000
GPS	1	2 000	2 000
360 kamera	1	5 000	5 000
SUM		8 550	37 600

Provider		2017	SUM
Brødrene Dahl AS		1 600	1 600
Stette Vannskjæring AS		8 000	8 000
Anda Olsen AS		6 000	6 000
Other		22 000	22 000
SUM		37 600	37 600

Financial plan		2017	SUM	
NTNU		37 600	37 600	%-vis
				100 %
SUM		37 600	37 600	#REF!

ID301702 Hovedprosjekt	Prosjekt Autonom platform	Antall møter denne periode 1).	Firma - Oppdragsgiver NTNU Ålesund /	Side 1 av 2
Rapport fra prosess Framdriftsrapport	Periode/uke(r) uke 3 og 4	Antall timer denne per. (fra logg)	Prosjektgruppe (navn)	Dato 26.01.17

Hovedhensikt / fokus for arbeidet i denne perioden

Gjennomføre et godt forprosjekt. Starte tankeprosessen rundt de forskjellige temaene i hovedprosjektet og lage en god plan for gjennomføringen.

Planlagte aktiviteter i denne perioden

Gjennomføre forprosjektet. Oppstart på forprosjektet blir sammen med første modul i industri 4.0. Derfor ligger mye av grunnlaget i forprosjektet i arbeid utført ifm første modul. Idé myldring på de første temaene, kommunikasjon, autopilot og platform.

Virkelig gjennomførte aktiviteter i denne perioden

Alle planlagte oppgaver ble utført i perioden.

Beskrivelse av/begrunnelse for eventuelle avvik mellom planlagte og virkelige aktiviteter

NIL

Beskrivelse av /begrunnelse for endringer som nå ønskes i selve prosjektets innhold eller i den videre framgangsmåten - eller framdriftsplanen

NIL

Hovederfaring fra denne perioden

Beregne tid er vanskelig. Spesielt på software utvikling og langt frem i tid.

Hovedhensikt/fokus neste periode

Få ferdig designet av Platform til det punkt at deler kan bestilles og 3D modell er tegnet. Få avklart hvilken kommunikasjonsløsning som skal benyttes og starte utvikling og test av denne. Informasjonsinnhenting på tidligere utviklede autopilotssystem. Lage en enkel utgave av et kartsystem.

Planlagte aktiviteter neste periode

Ferdigstille Platform desig frem til endelig design og 3D tegninger. Materialer bestilt. Informasjonsinnhenting for kommunikasjonsløsning og autopilot. Avklare krav til kommunikasjonsløsning, lande endelig løsning og sette opp et enkelt testsystem. Lage en enkel GUI for testing av trustere og kommunikasjon. Lage et utkast til kartsystem for bruk i autopilotsystemet.

Annet

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

ID301702 Hovedprosjekt	Prosjekt Autonom platform	Antall møter denne periode 1).	Firma - Oppdragsgiver NTNU Ålesund /	Side 2 av 2
Rapport fra prosess Framdriftsrapport	Periode/uke(r) uke 3 og 4	Antall timer denne per. (fra logg)	Prosjektgruppe (navn)	Dato 26.01.17

Ønske om /behov for veiledning, tema i undervisningen – drøfting ellers	
Godkjenning/signatur gruppeleder	Signatur øvrige gruppedeltakere

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

ID301702 Hovedprosjekt	Prosjekt Autonom platform	Antall møter denne periode 1).	Firma - Oppdragsgiver NTNU Ålesund /	Side 1 av 2
Rapport fra prosess Framdriftsrapport	Periode/uke(r) uke 3 og 4	Antall timer denne per. (fra logg)	Prosjektgruppe (navn)	Dato 26.01.17

Hovedhensikt / fokus for arbeidet i denne perioden

Platform: Fullføre design og simulering. Lage kommunikasjonssystem og enkel GUI for testing.

Planlagte aktiviteter i denne perioden

Ferdigstille Platform design frem til endelig design og 3D tegninger. Materialer bestilt.
Informasjonsinnhenting for kommunikasjonsløsning og autopilot.
Avklare krav til kommunikasjonsløsning, lande endelig løsning og sette opp et enkelt testsystem.
Lage en enkel GUI for testing av trustere og kommunikasjon.
Lage et utkast til kartsystem for bruk i autopilotsystemet.

Virkelig gjennomførte aktiviteter i denne perioden

Platform design frem til endelig design og 3D tegninger
Informasjonsinnhenting for kommunikasjonsløsning og autopilot.
Avklare krav til kommunikasjonsløsning, lande endelig løsning og sette opp et enkelt testsystem.
Lage en enkel GUI for testing av kommunikasjon.
Lage et utkast til kartsystem for bruk i autopilotsystemet.

Beskrivelse av/begrunnelse for eventuelle avvik mellom planlagte og virkelige aktiviteter

Materialer ikke bestilt. Grunnet vanskeligheter med simulering av stabilitet.

Beskrivelse av /begrunnelse for endringer som nå ønskes i selve prosjektets innhold eller i den videre framgangsmåten - eller framdriftsplanen

NIL

Hovederfaring fra denne perioden

Vanskelig å få den riktige hjelpen og riktig informasjon.

Hovedhensikt/fokus neste periode

Bestille deler til platform, designe stabilitetssystemet.
Toveiskommunikasjon mellom platform og bruker. Lage algoritme for å bestemme kjøreretning.

Planlagte aktiviteter neste periode

Bestille deler. Konsept for stabiliseringssystem. Bestille deler til stabiliseringssystem. Software for stabiliseringssystem. Matematikken for stabiliseringssystem.

Annet

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

ID301702 Hovedprosjekt	Prosjekt Autonom platform	Antall møter denne periode 1).	Firma - Oppdragsgiver NTNU Ålesund /	Side 2 av 2
Rapport fra prosess Framdriftsrapport	Periode/uke(r) uke 3 og 4	Antall timer denne per. (fra logg)	Prosjektgruppe (navn)	Dato 26.01.17

Ønske om /behov for veiledning, tema i undervisningen – drøfting ellers	
Godkjenning/signatur gruppeleder	Signatur øvrige gruppedeltakere

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

ID301702 Hovedprosjekt	Prosjekt Autonom plattform	Antall møter denne periode 1).	Firma - Oppdragsgiver NTNU Ålesund /	Side 1 av 2
Rapport fra prosess Framdriftsrapport	Periode/uke(r)	Antall timer denne per. (fra logg)	Prosjektgruppe (navn)	Dato 23.02.17

Hovedhensikt / fokus for arbeidet i denne perioden

Bestille deler til plattform, designe stabilitetssystemet.

Toveiskommunikasjon mellom plattform og bruker. Lage algoritme for å bestemme kjøreretning.

Planlagte aktiviteter i denne perioden

Bestille deler. Konsept for stabiliseringssystem. Bestille deler til stabiliseringssystem. Software for stabiliseringssystem. Matematikken for stabiliseringssystem.

Få avklart hvilken type kommunikasjonsløsning som skal benyttes. Få sendt data mellom klient og server.

Virkelig gjennomførte aktiviteter i denne perioden

Deler bestilt. Balansesystem avklart, blir pumping av vann. Både luft og vann testet i bassenget. Software for balanse system påbegynt.

Kommunikasjonsløsning mellom de forskjellige enhetene avklart. Kommunikasjon mellom klient og server opprettet.

Beskrivelse av/begrunnelse for eventuelle avvik mellom planlagte og virkelige aktiviteter

NIL

Beskrivelse av /begrunnelse for endringer som nå ønskes i selve prosjektets innhold eller i den videre framgangsmåten - eller framdriftsplanen

NIL

Hovederfaring fra denne perioden

Må ha bedre verktøy for styring av programvareutvikling. Er i dialog med Arne Styve for å utstyr og litt utdanning på dette.

Hovedhensikt/fokus neste periode

Sette sammen plattformen og balansesystemet. Skrive koden for balansesystemet.

Opprette wifi radiolink for kommunikasjon mellom klient og plattform. Få implementert json som dataformat. Sende kontrolldata fra autopilot og manuell styring.

Planlagte aktiviteter neste periode

Bygge plattform. Montere balansesystem. Skrive kode til balansesystem. Teste plattform og balansesystem.

Sette opp radiolink og teste denne. Få implementert json som dataformat. Sende data fra autopilot og manuell modus.

Annet

Ønske om /behov for veiledning, tema i undervisningen – drøfting ellers

Godkjenning/signatur gruppeleder

Signatur øvrige gruppedeltakere

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

ID301702 Hovedprosjekt	Prosjekt Autonom platform	Antall møter denne periode 1).	Firma - Oppdragsgiver NTNU Ålesund /	Side 2 av 2
Rapport fra prosess Framdriftsrapport	Periode/uke(r)	Antall timer denne per. (fra logg)	Prosjektgruppe (navn)	Dato 23.02.17

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

ID301702 Hovedprosjekt	Prosjekt Autonom plattform	Antall møter denne periode 1).	Firma - Oppdragsgiver NTNU Ålesund /	Side 1 av 1
Rapport fra prosess Framdriftsrapport	Periode/uke(r)	Antall timer denne per. (fra logg)	Prosjektgruppe (navn)	Dato 23.03.17

Hovedhensikt / fokus for arbeidet i denne perioden

Sette sammen plattformen og balansesystemet. Skrive koden for balansesystemet.
Opprette wifi radiolink for kommunikasjon mellom klient og plattform. Få implementert json som dataformat. Sende kontrolldata fra autopilot og manuell styring.
Innføre et kontrollsystem for software utvikling.

Planlagte aktiviteter i denne perioden

Bygge plattform. Montere balansesystem. Skrive kode til balansesystem. Teste plattform og balansesystem.
Sette opp radiolink og teste denne. Få implementert json som dataformat. Sende data fra autopilot og manuell modus.
Utdanning i Jira programmvare

Virkelig gjennomførte aktiviteter i denne perioden

Plattform montert. Deler av balansesystem montert og programvare for innhenting av pitch/roll samt avlesning av trykksensorer er ferdig. Kommunikasjon mellom balansesystem og server opprettet.
Utstyr til radiolink mottatt og testet. Json implementert. Sender data fra klient til server i manuell og autopilot modus. Manuell modus er nå fullført men ikke testet.
Server godt på vei, håndtering av lese/skrive problematikk implementert.

Beskrivelse av/begrunnelse for eventuelle avvik mellom planlagte og virkelige aktiviteter

Tilpassing av delene tok lengre tid en beregnet.

Beskrivelse av /begrunnelse for endringer som nå ønskes i selve prosjektets innhold eller i den videre framgangsmåten - eller framdriftsplanen

NIL

Hovederfaring fra denne perioden

Avhengighet av eksterne er et usikkerhetsmoment mtp tid.

Hovedhensikt/fokus neste periode

Fullføre balansesystem og få testet dette. Fullføre plattform.
Autopilot

Planlagte aktiviteter neste periode

Ferdigstille plattform. Fullføre software til balansesystem. Lage kontrollalgoritmene. Teste balansesystemet.
Håndtere waypoints. Algoritme for utregning av retning og avstand. Test av manuelt styresystem.
Konfigurere Odroid med server programvare. Teste klient på tablet.

Annet

Ønske om /behov for veiledning, tema i undervisningen – drøfting ellers

Godkjenning/signatur gruppeleder

Signatur øvrige gruppedeltakere

ID301702 Hovedprosjekt	Prosjekt Autonom plattform	Antall møter denne periode 1).	Firma - Oppdragsgiver NTNU Ålesund /	Side 1 av 1
Rapport fra prosess Framdriftsrapport	Periode/uke(r)	Antall timer denne per. (fra logg)	Prosjektgruppe (navn)	Dato 23.03.17

Hovedhensikt / fokus for arbeidet i denne perioden Ferdigstille plattform og balansesystem. Test av manuelt styresystem Teste klient på tablet oppmot server og manuell styring.	
Planlagte aktiviteter i denne perioden Ferdigstille plattform. Fullføre software til balansesystem. Lage kontrollalgoritmene. Teste balansesystemet. Handtere waypoints. Algoritme for utregning av retning og avstand. Test av manuelt styresystem. Konfigurere Odroid med server programvare. Teste klient på tablet.	
Virkelig gjennomførte aktiviteter i denne perioden Plattform ferdig og testet. Balansesystem montert og testet. Tatt avgjørelse på forbedring i programvare og sensor. Waypoint utregning klar, Systemet klart til test. Testtrigg ferdig, programvare utvikles. Odroid klar til serverprogramvare og klient er testet mot server og plattform.	
Beskrivelse av/begrunnelse for eventuelle avvik mellom planlagte og virkelige aktiviteter Grunnet endring i instrumentering av balansesystemet. (Fire sensorer i stedet for 1) Er ikke ny programvare ferdig.	
Beskrivelse av /begrunnelse for endringer som nå ønskes i selve prosjektets innhold eller i den videre framgangsmåten - eller framdriftsplanen NIL	
Hovederfaring fra denne perioden Endringer ift opprinnelig plan tar lang tid og implementere.	
Hovedhensikt/fokus neste periode Fullføre oppgrader balansesystem. Teste waypoint handtering. GUI klient.	
Planlagte aktiviteter neste periode Fullføre installasjon og programvare til nye sensorer. Teste dette. Fullført testtrigg for kontroll av waypoint handling. Knytte Arduinoer opp mot server. Videreutvikle GUI på klienten.	
Annet	
Ønske om /behov for veiledning, tema i undervisningen – drøfting ellers	
Godkjenning/signatur gruppeleder	Signatur øvrige gruppedeltakere

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

Meeting report 13.01.17

Opening

A supervisor meeting regarding the **Sea farm platform thesis** was held at NTNU Ålesund the 13.01.17. All members participated to the meeting. **Matias Heggen, Christer Bakken, Per Martin Leinan, Martin Blom, Ottar Osen and Houxiang Zhang.**

Approval of the progress report

The meeting starts with a review of the first progress report. The supervisors approved of the form, and agreed with the planned progress of the thesis.

Questions from the members

Since the thesis contains the development of a new product, the members wonder what the supervisors approve as a final product.

The platform should have active stabilization, since this is an easy part to implement.

The platform could have several functions, we should consider at least 4-5 of them but it is OK if the final product only has 2-3.

The group members would like someone in ship design to help with the design and the workshop to help build the platform, considering that this thesis drifts past what is expected from automation students.

First of all, the school doesn't have the capacity of putting this amount of work at any of the teachers, secondly, automation students should have the basic knowledge considering all parts of their thesis. There are however teachers who can help us along the way.

Expect to see calculations regarding buoyancy and stability theories, done by the bachelor thesis members.

Discussion

Houxiang asks why we are designing a new platform when the school already has a boat for such projects, why would the platform be better and what are the benefits. The reason should be well explained in the final report.

Since the platform are to have active stabilization, Ottar asks if it should be able of raise and lowering itself as well? We agree that this should be possible.

Recommendations

Ask for help by Morten or Henrique for designing the platform. Start by reading the book Vessel Stability.

Meeting report 30.01.17

Opening

A supervisor meeting regarding the **Sea farm platform thesis** was held at NTNU Ålesund the 30.01.17. All members participated to the meeting. **Matias Heggen, Christer Bakken, Per Martin Leinan, Martin Blom, Ottar Osen and Houxiang Zhang.**

Approval of the progress report

The meeting starts with a review of our progress report. The supervisors approved of the report, and agreed with the planned progress of the thesis.

Questions from the members

What is the budget of the project?

A budget should be made considering all parts of the platform, if the project is well planned and documented, the budget will be approved.

Discussion

The group agreed on using an Android application as GUI for the platform, the supervisors did not seem convinced, if there is an android application, would we need both PC and tablet or could this be merged to function on only the tablet or only the PC? A layout of the system should be made for the next meeting.

We presented the low version of the hexagon model, both Ottar and Houxiang said the model was too low, there would be insufficient buoyancy in it and told us to make it at least taller.

Even though the model wasn't used, all models should be mentioned and displayed in the final report. Houxiang wanted to see documentation from simulation for each of the models as well.

Recommendations

For help doing the simulations, Houxiang recommended asking Lars Ivar Hatledal, a student of his.

Meeting report 10.02.17

Opening

A supervisor meeting regarding the **Sea farm platform thesis** was held at NTNU Ålesund the 10.02.17. Participants: **Christer Bakken, Per Martin Leinan, Martin Blom, Ottar Osen and Houxiang Zhang**. Members who didn't participate in this meeting: **Matias Heggen**.

Approval of the progress report

The meeting starts with a review of our progress report. The supervisors approved of the report, and agreed with the planned progress of the thesis.

Discussion

The first rectangular model was presented to the supervisors, they approved of the basic design but requested further work on the structure since the model was too complex. More models were to be presented to Houxiang before starting the build.

The calculations for buoyancy and stability was done with help from Henrique, Ottar and Houxiang still wanted to see some simulations for the models in addition to the calculations, at least for the main two or three models.

A topic of how to run the thrusters came up, and Ottar believes the thrusters should be controlled individually for maximizing the maneuverability of the platform.

Recommendations

For communication between separate components, Ottar suggests using the MQTT protocol for several reasons, quality of service, at most once, at least once and only once. Send a new in three seconds, redundancy of sending several messages. Low footprint, doesn't use many bytes. Adds up to a big code but there are libraries for it. Houxiang believed using I^2C was the best solution for us.

Meeting report 24.03.17

Opening

A supervisor meeting regarding the **Sea farm platform thesis** was held at NTNU Ålesund the 24.03.17. Participants: **Christer Bakken, Matias Heggen, Per Martin Leinan, Martin Blom and Ottar Osen**. Members who didn't participate to this meeting: **Houxiang Zhang**.

Approval of the progress report

The meeting starts with a review of our progress report. Ottar approved of the report, and agreed with the planned progress of the thesis.

Discussion

A video showing the platform behavior in waves from the second test were displayed, at this stage the platform was only equipped with thrusters and batteries. Ottar suggested calculating the platform depth with and without the rest of the electrical components. Verify the center of rotation for seeing the calculations to be correct.

Meeting report 02.05.17

Opening

A supervisor meeting regarding the **Sea farm platform thesis** was held at **NTNU Ålesund** the **02.05.17**.
Partisipants: **Matias Heggen, Per Martin Leinan, Martin Blom, Ottar Osen and Houxiang Zhang**.
Members who didn't participate to this meeting: **Christer Bakken**.

Approval of the progress report

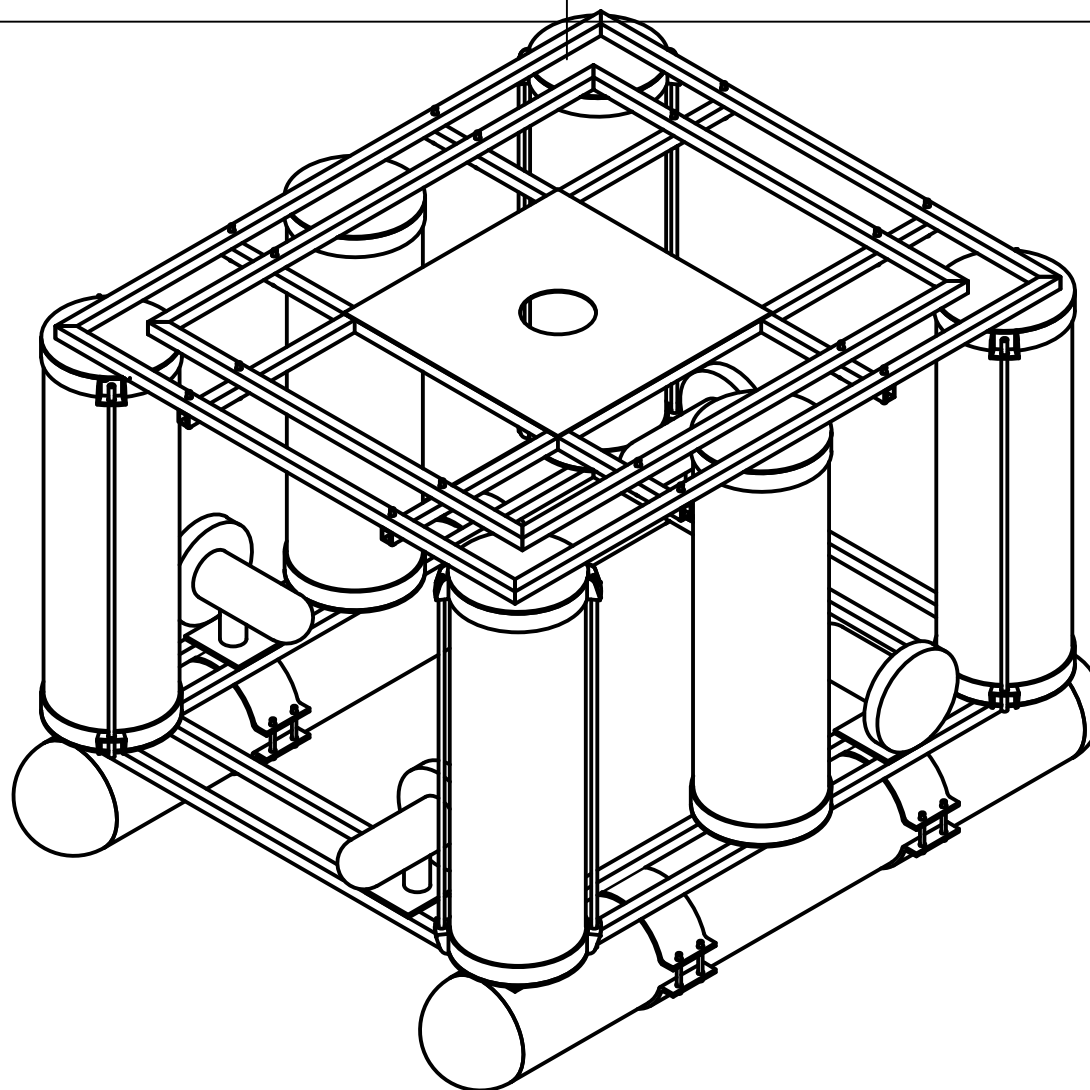
The meeting starts with a review of our progress report. The supervisors approved of the report, and agreed with the planned progress of the thesis.


Discussion

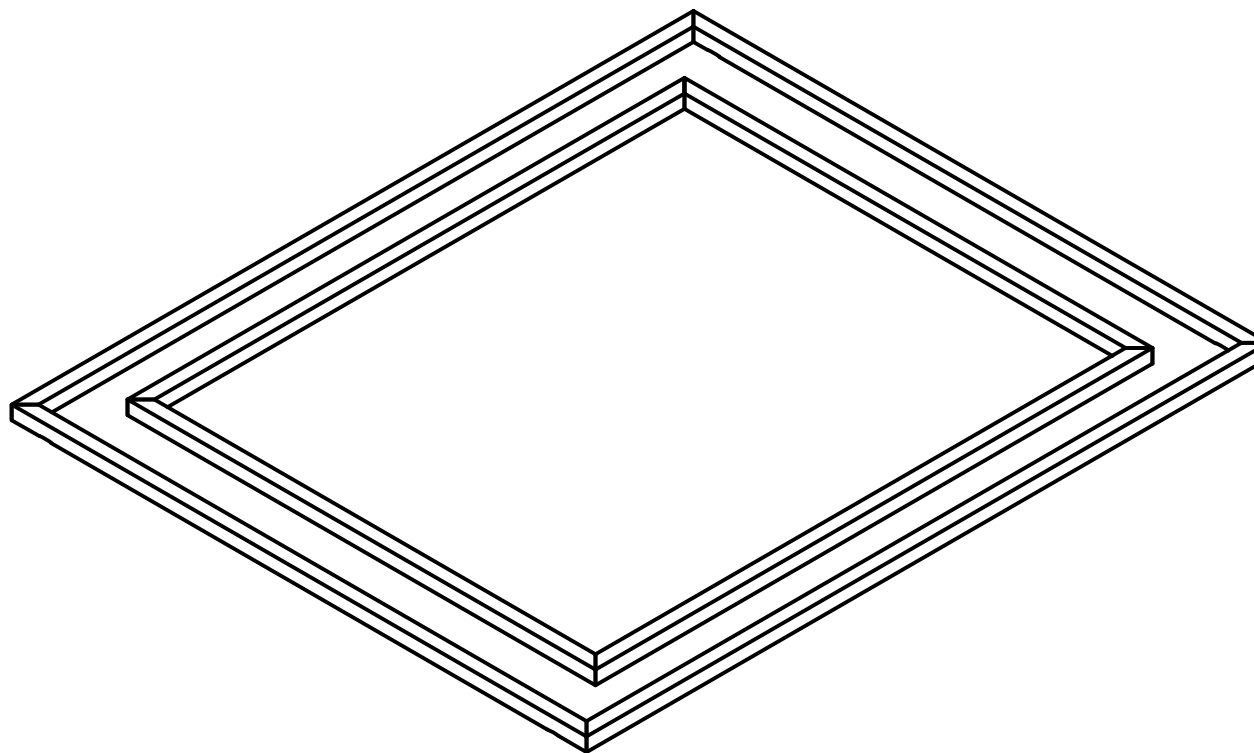
Houxiang requested a draft of the thesis at the end of the week.

Recommendations


Houxiang suggests using a feed forward system for the stability system

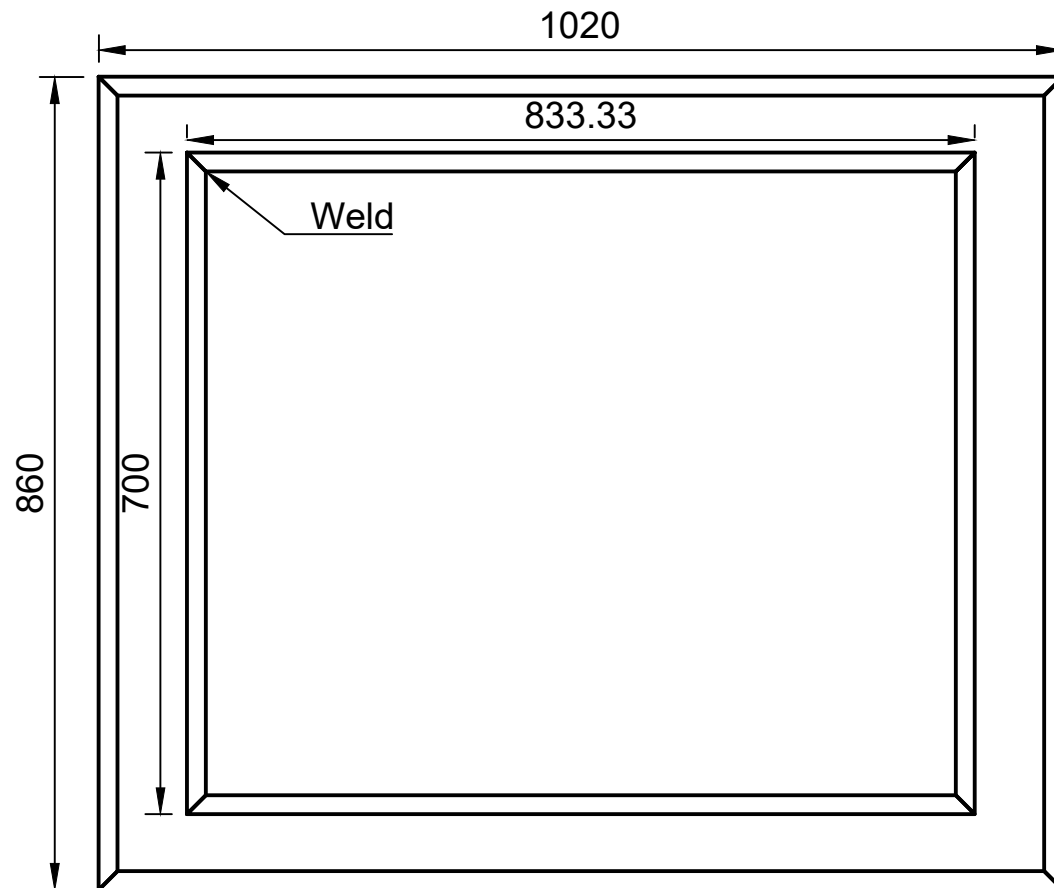


Dept.	Technical reference	Created by Martin Andreas Blom 12.02.2017	Approved by		
		Document type	Document status		
		Title Sea farm platform	DWG No.		
			Rev.	Date of issue	Sheet 1/18



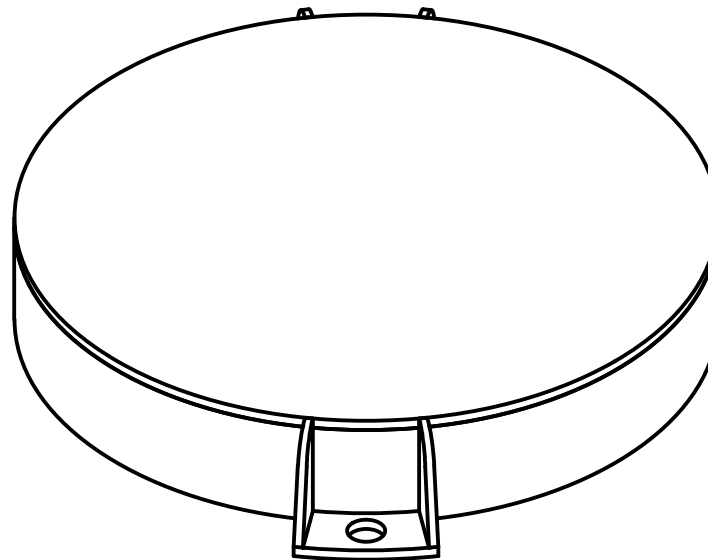
Base framework
Consists of 20x20mm square
aluminium pipes, 2mm walls.

Dept.	Technical reference	Created by Martin Andreas Blom 12.02.2017	Approved by		
 NTNU Kunnskap for en bedre verden		Document type	Document status		
		Title Sea farm platform Base framework	DWG No.		
			Rev.	Date of issue	Sheet 2/18



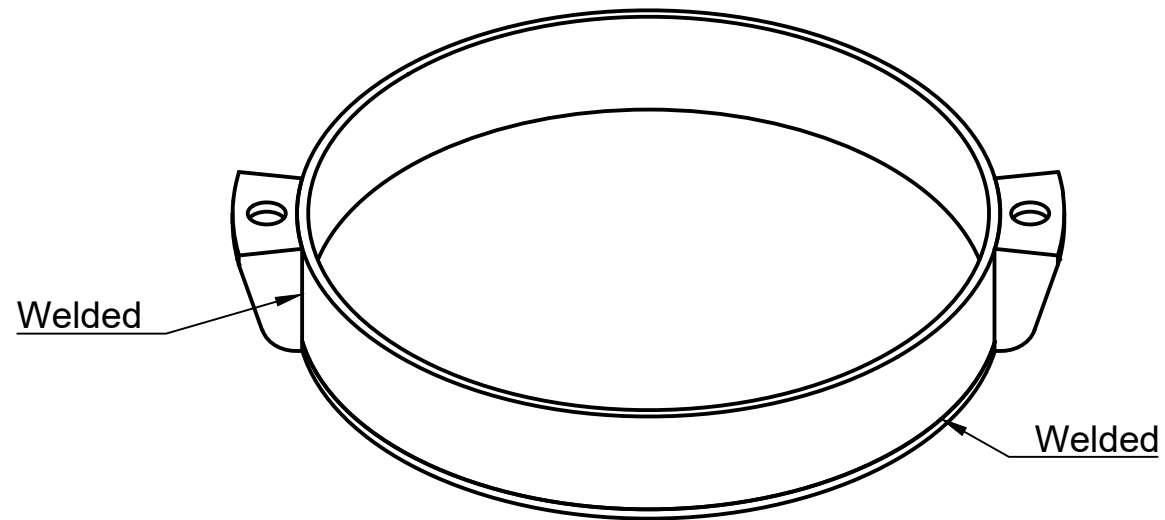
Equivalent to top frame,
without holes.

Dept.	Technical reference	Created by Martin Andreas Blom 12.02.2017	Approved by		
 NTNU Kunnskap for en bedre verden		Document type	Document status		
		Title Sea farm platform Base framework	DWG No.		
			Rev.	Date of issue	Sheet 3/18



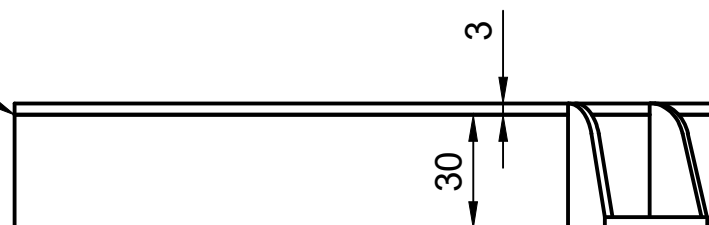
Cylinder caps, total 8 pcs with the side ears and 2 without.


Dept.	Technical reference	Created by Martin Andreas Blom 12.02.2017	Approved by		
 NTNU Kunnskap for en bedre verden		Document type	Document status		
		Title Sea farm platform Cylinder caps	DWG No.		
			Rev.	Date of issue	Sheet 4/18

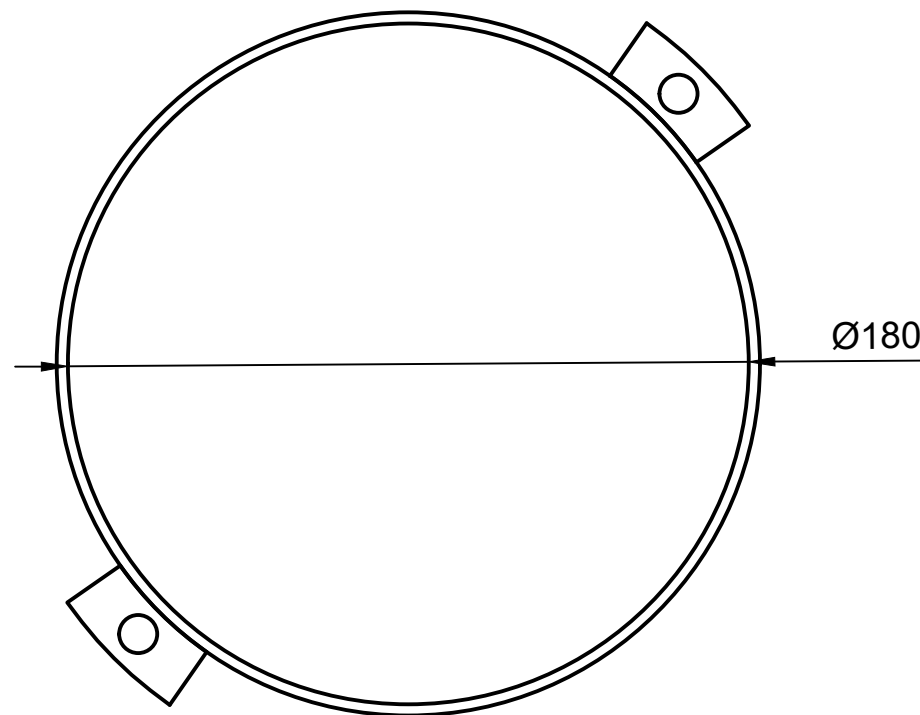



Dept.	Technical reference	Created by Martin Andreas Blom 12.02.2017	Approved by		
		Document type	Document status		
		Title Sea farm platform Cylinder caps	DWG No.		
			Rev.	Date of issue	Sheet 5/18

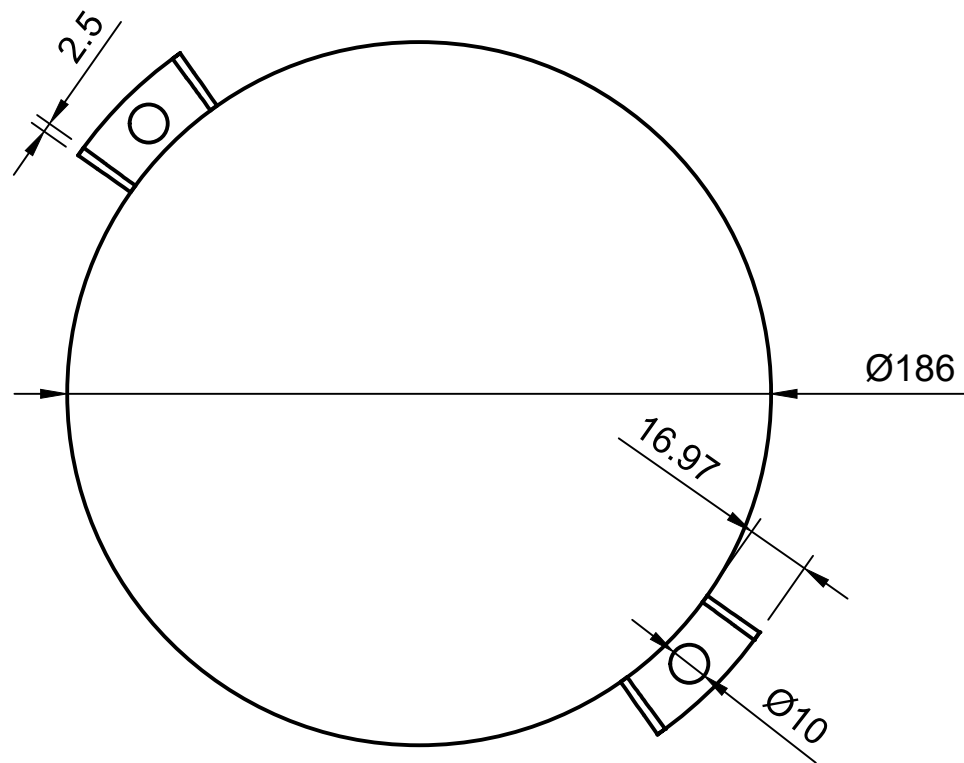
Welded




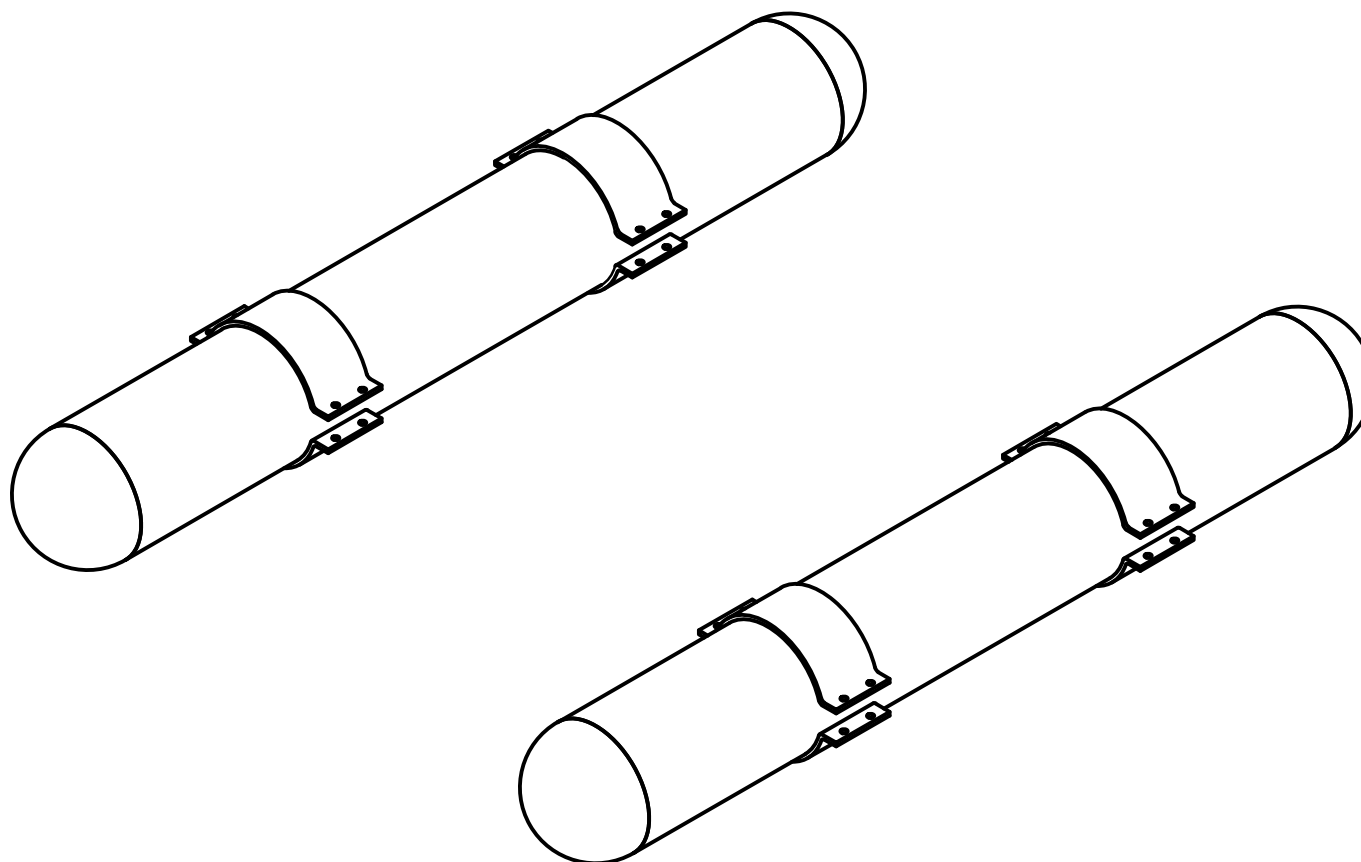
Dept.	Technical reference	Created by Martin Andreas Blom 12.02.2017	Approved by		
 NTNU Kunnskap for en bedre verden		Document type	Document status		
		Title Sea farm platform Cylinder caps	DWG No.		
			Rev.	Date of issue	Sheet 6/18




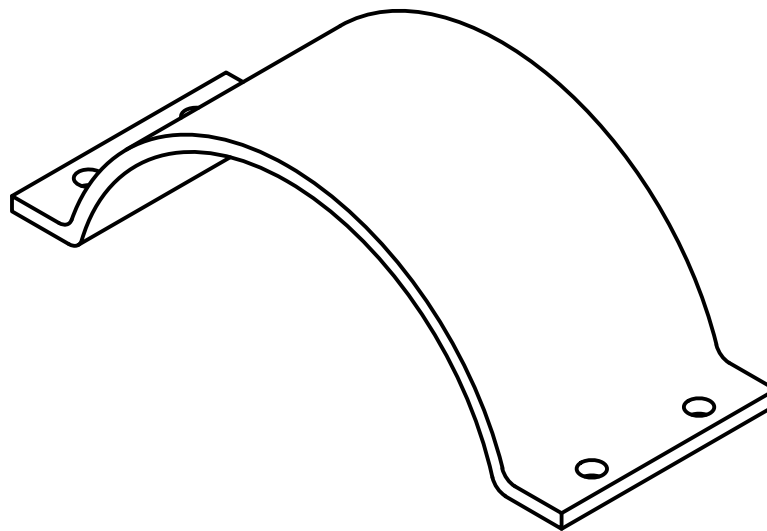
Dept.	Technical reference	Created by Martin Andreas Blom 12.02.2017	Approved by		
		Document type	Document status		
		Title Sea farm platform Cylinder caps	DWG No.		
			Rev.	Date of issue	Sheet 7/18




Dept.	Technical reference	Created by Martin Andreas Blom 12.02.2017	Approved by		
 NTNU Kunnskap for en bedre verden		Document type	Document status		
		Title Sea farm platform Cylinder caps	DWG No.		
			Rev.	Date of issue	Sheet 8/18

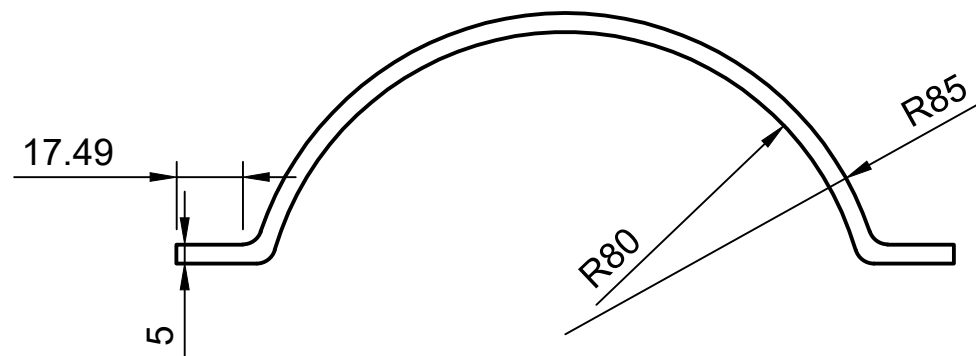



Dept.	Technical reference	Created by Martin Andreas Blom 12.02.2017	Approved by		
		Document type	Document status		
		Title Sea farm platform Battery pontoons	DWG No.		
			Rev.	Date of issue	Sheet 9/18

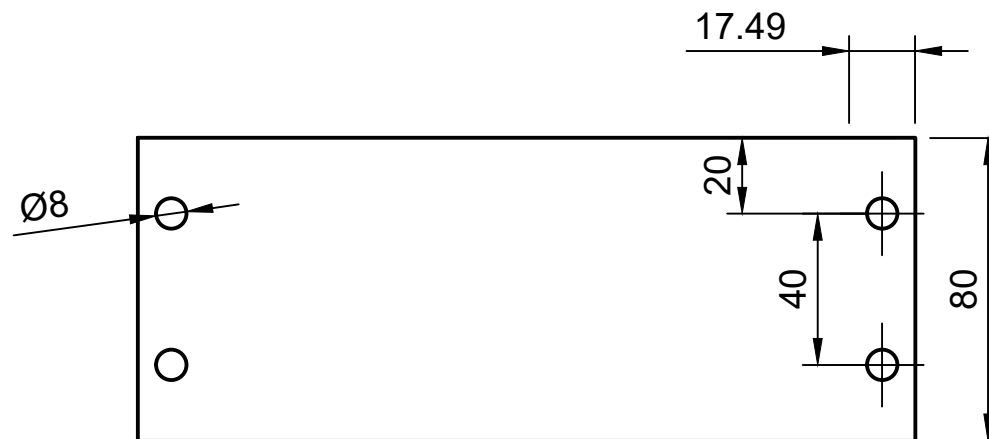



Mounting brackets for
bottom pontoons.
8 pieces in total

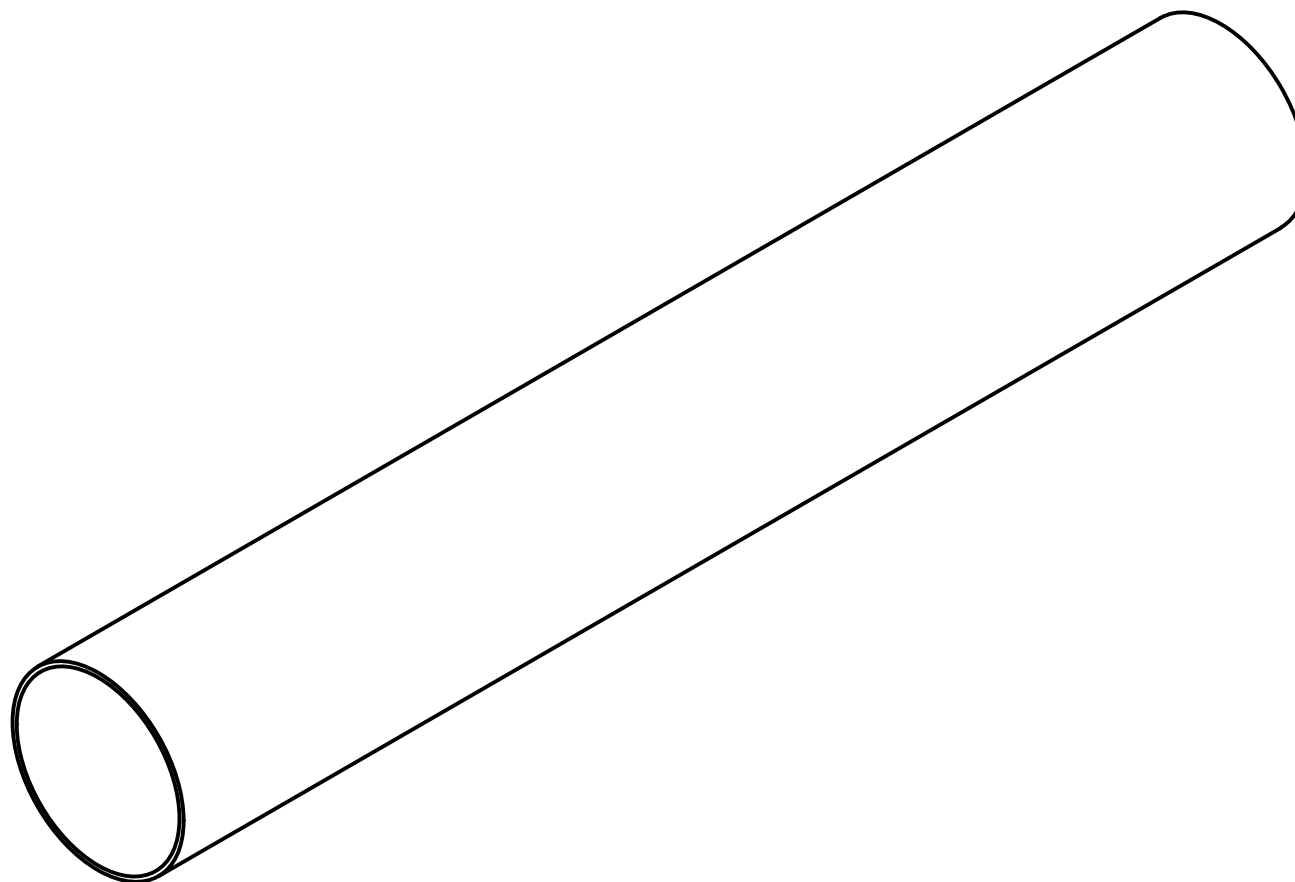
Dept.	Technical reference	Created by Martin Andreas Blom 12.02.2017	Approved by		
		Document type	Document status		
		Title Sea farm platform Mounting brackets	DWG No.		
			Rev.	Date of issue	Sheet 10/18




Dept.	Technical reference	Created by Martin Andreas Blom 12.02.2017	Approved by		
 NTNU Kunnskap for en bedre verden		Document type	Document status		
		Title Sea farm platform Mounting brackets	DWG No.		
			Rev.	Date of issue	Sheet 11/18



Dept.	Technical reference	Created by Martin Andreas Blom 12.02.2017	Approved by		
 NTNU Kunnskap for en bedre verden		Document type	Document status		
		Title Sea farm platform Mounting brackets	DWG No.		
			Rev.	Date of issue	Sheet 12/18




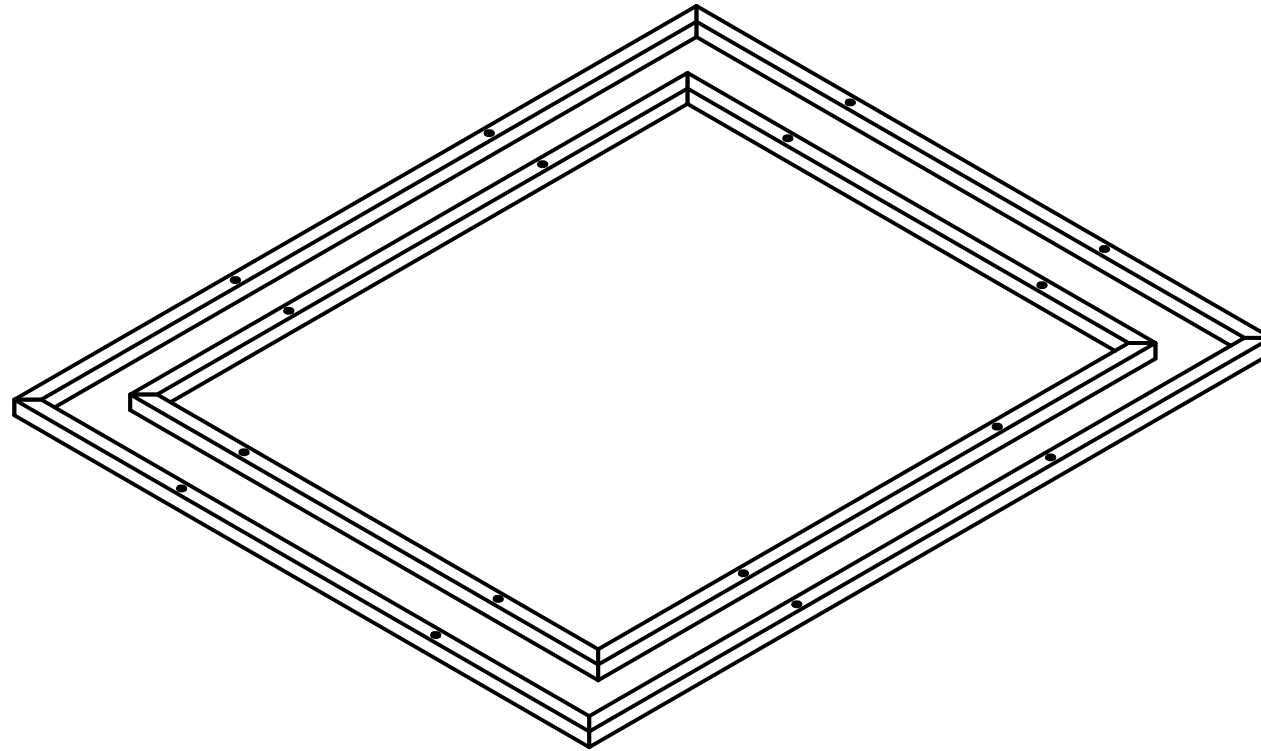
PE 100/300, SDR 41
Diameter: 160
Length: 1000mm
2 pieces in total

Dept.	Technical reference	Created by Martin Andreas Blom 12.02.2017	Approved by		
		Document type	Document status		
		Title Sea farm platform Battery pontoons	DWG No.		
			Rev.	Date of issue	Sheet 13/18



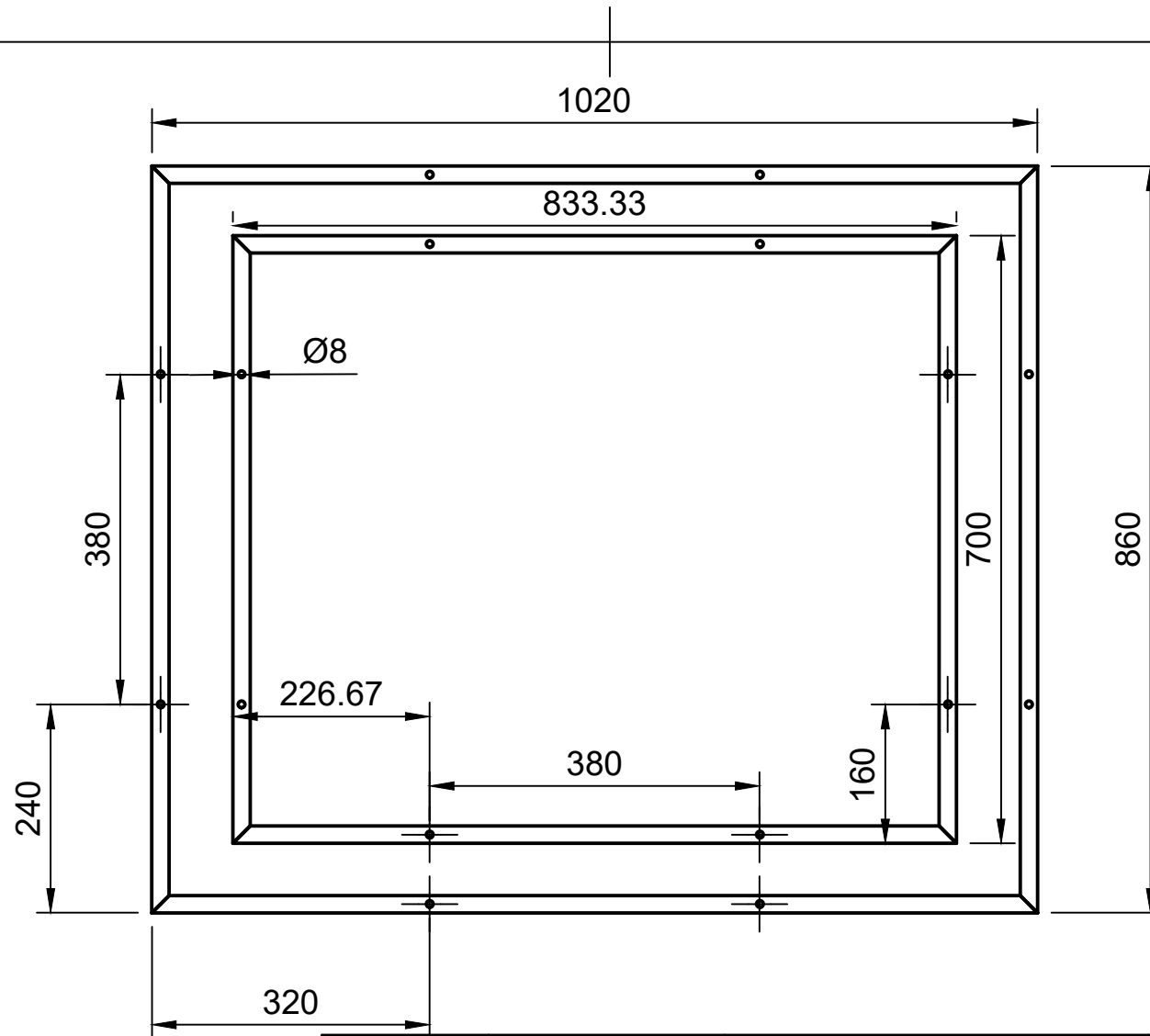
PE 100/300, SDR 41
Diameter: 180mm
Length: 600mm
6 pieces in total

Dept.	Technical reference	Created by Martin Andreas Blom 12.02.2017	Approved by		
 NTNU Kunnskap for en bedre verden		Document type	Document status		
		Title Sea farm platform Vertical pontoons	DWG No.		
			Rev.	Date of issue	Sheet 14/18

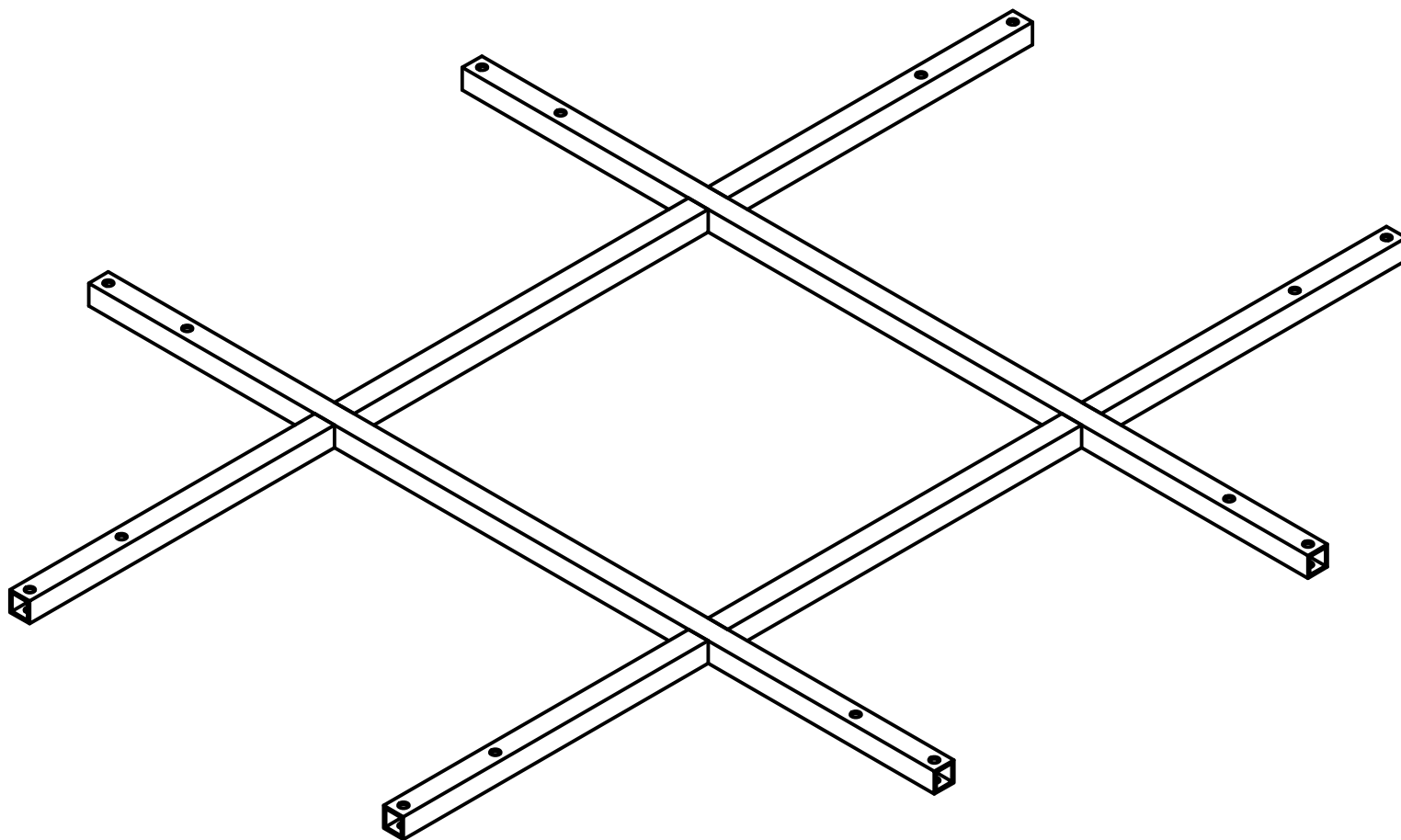


Base framework
Consists of 20x20mm
square aluminium pipes,
2mm walls.

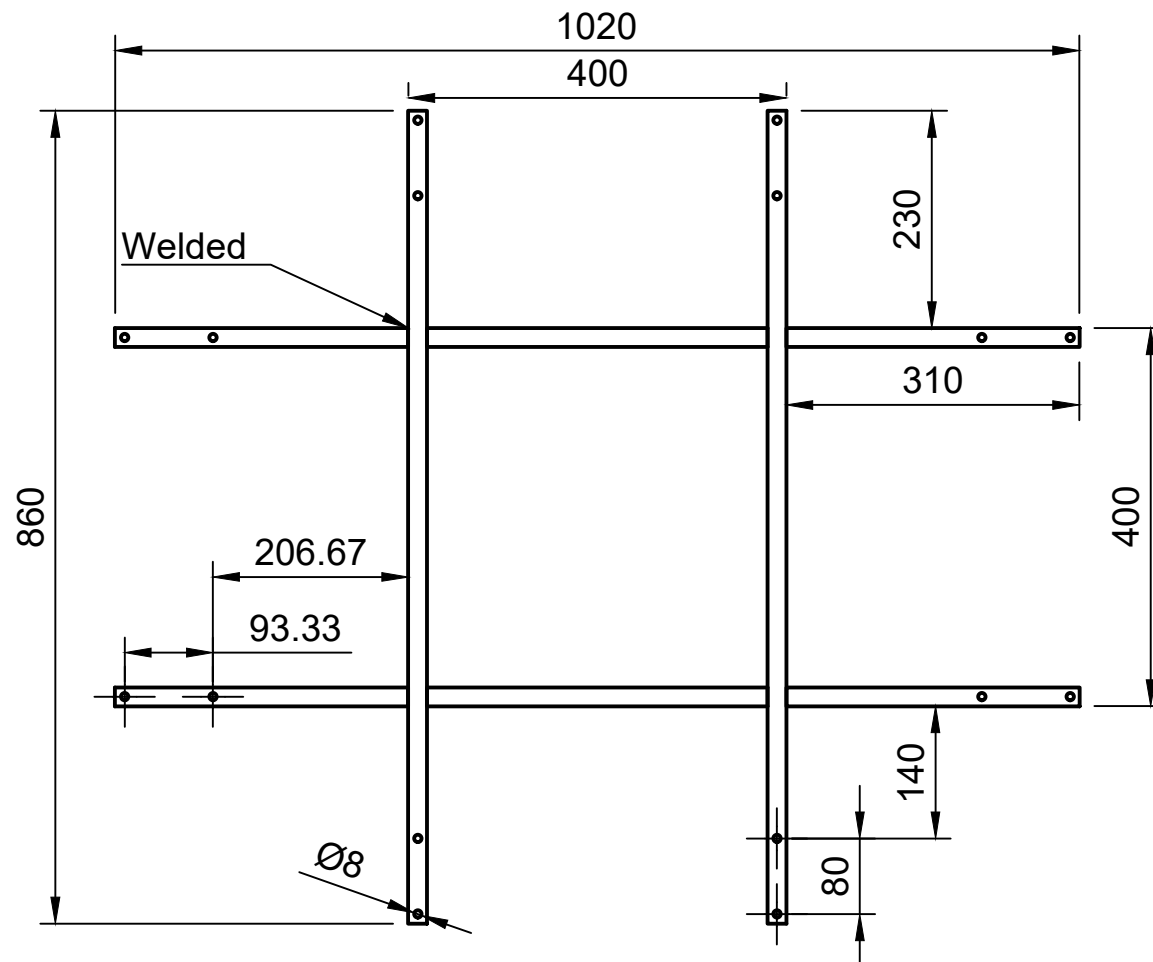
Dept.	Technical reference	Created by Martin Andreas Blom 12.02.2017	Approved by		
		Document type	Document status		
		Title Sea farm platform Top framework	DWG No.		
			Rev.	Date of issue	Sheet 15/18



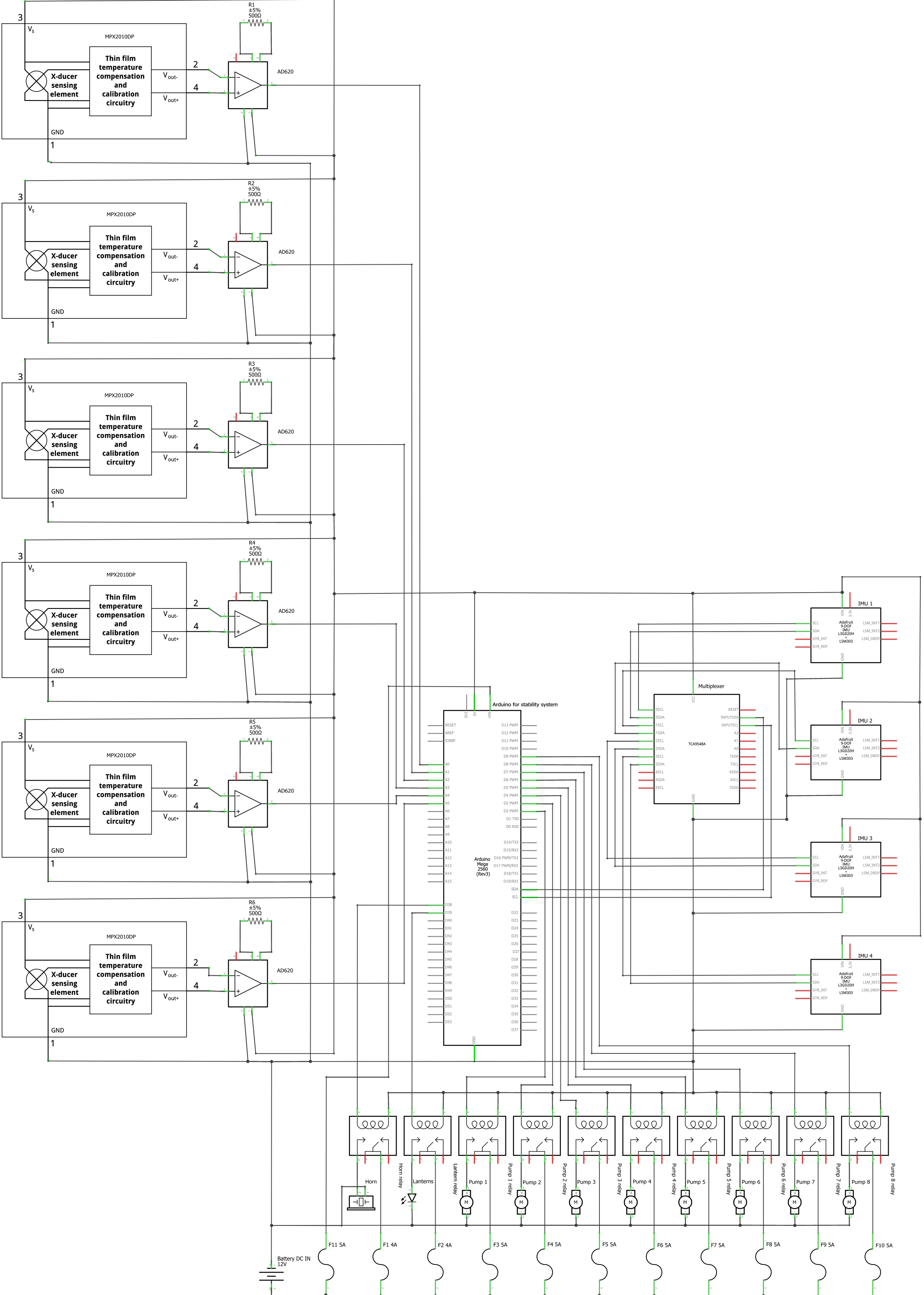
Dept.	Technical reference	Created by Martin Andreas Blom 12.02.2017	Approved by	
 NTNU Kunnskap for en bedre verden		Document type	Document status	
		Title Sea farm platform Top framework	DWG No.	
			Rev.	Date of issue
				Sheet 16/18

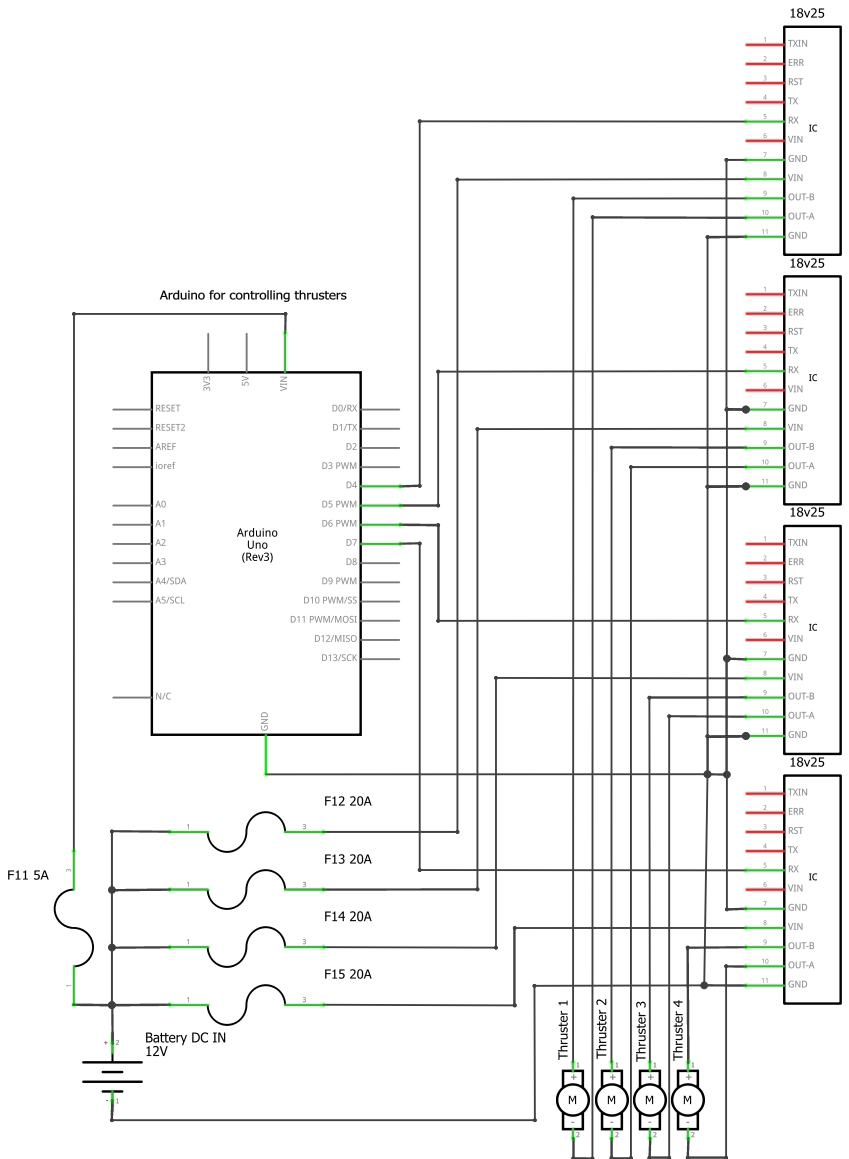


Dept.	Technical reference	Created by Martin Andreas Blom 12.02.2017	Approved by		
 NTNU Kunnskap for en bedre verden		Document type	Document status		
		Title Sea farm platform Top center frame	DWG No.		
			Rev.	Date of issue	Sheet 17/18



Dept.	Technical reference	Created by Martin Andreas Blom 12.02.2017	Approved by	
 NTNU Kunnskap for en bedre verden		Document type	Document status	
		Title Sea farm platform Top center frame	DWG No.	
		Rev.	Date of issue	Sheet 18/18





Platform Basic Stability
Assumption: floating on columns (zero deck in top 0,0,0)
Stage 1: floating only battery pontoon
Stage 2: floating columns
* Calculation only for Stage 2!
GZ - Small angles (±17°)

Watertight Volume *				
	Volume	Fresh	Sea	
Stage 1	Bat. A	0,02010619	0,020106193	0,020608848
	Bat. B	0,02010619	0,020106193	0,020608848
	Thrusters	0,002	0,002	0,00205
Total Disp		0,042212386	0,043267696	
Kg Left		57,12761403	56,07230438	
		OK - Valid - Stage 2	OK - Valid - Stage 2	
Stage 2	Col 1	0,01781283		
	Col 2	0,01781283		
	Col 3	0,01781283		
	Col 4	0,01781283		
	Col 5	0,01781283		
	Col 6	0,01781283		
Total		0,10687698	106,8769821	109,5489066
		%	53,45 %	51,18 %
Draft		0,604162229	0,588293061	

Structure	x	y	z	mass
Col 1	0,45	-0,38	-0,35	5,14
Col 2	0,45	0,38	-0,35	5,14
Col 3	0	-0,38	-0,35	5,14
Col 4	0	0,38	-0,35	5,14
Col 5	-0,45	-0,38	-0,35	5,14
Col 6	-0,45	0,38	-0,35	5,14
Deck	0	0	0	8,8
Base	0	0	-0,7	24,2
Bat. A	0	-0,38	-0,78	8,4
Bat. B	0	0,38	-0,78	8,4
Payload	0	0	0	0
Div electrical	0	0	0	7
Water pumps	0	0	-0,55	2,4
Ventilblokk	0	0	0	0
Water col 1	0,45	-0,38	-0,4	0
Water col 2	0,45	0,38	-0,4	0
Water col 3	0	-0,38	-0,4	0
Water col 4	0	0,38	-0,4	0
Water col 5	-0,45	-0,38	-0,4	0
Water col 6	-0,45	0,38	-0,4	0
Thrusters	0	0	-0,55	5
Cabinet	0,45	0	0	2,15
Cabinet	-0,45	0	0	2,15
Total	0	0	-0,45206362	99,34

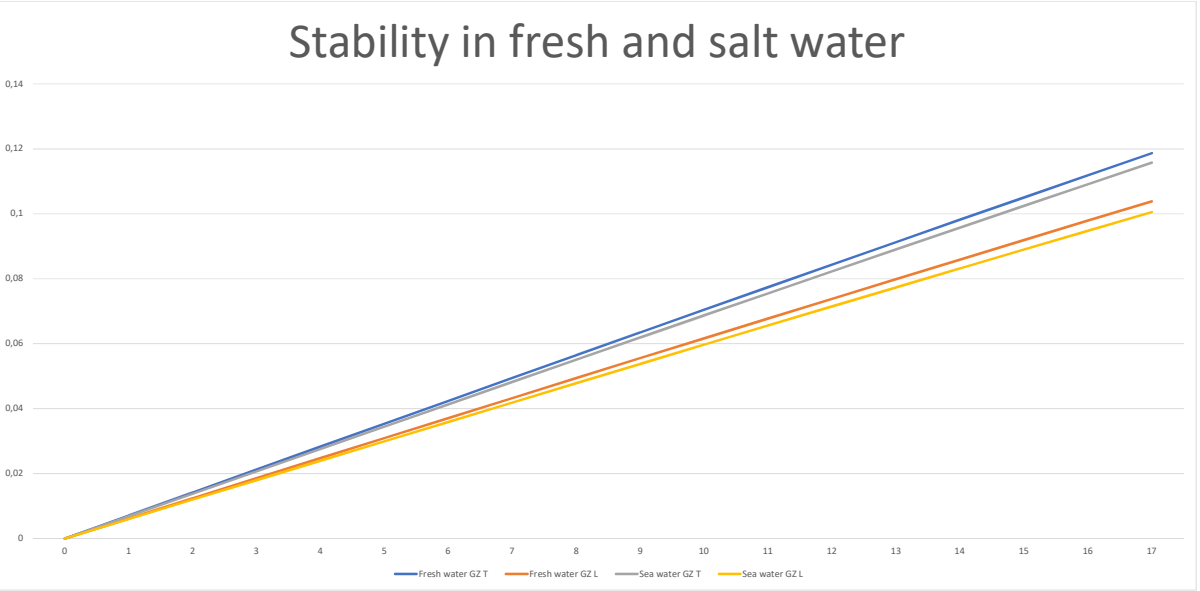
Num. Columns	Diameter	Lenght C	Lenght P	Breadth
6	0,18	0,7	1	0,85
Density	Sea	Fresh	Weight pipe	
	1,025	1		5,4
				4,26
Battery weight pr pontoon		8,4	Total Volume	147,089368

Inertia Columns		
Original I T	0,00005153	
Transf T	0,003674532	
I T total	0,022356374	
Original I Lon	0,00005153	
Transf Long	0,004277624	
I Long total	0,017419676	
Submerged Volume		
Fresh	0,09734	0,094917073
Stability partly submerged		
	Fresh	Sea
KB	0,604162229	0,588293061
KG	0,42793638	0,42793638
BM Trans.	0,229673047	0,23553586
BM Long.	0,178957014	0,183525209
GMT	0,405898896	0,39589254
GML	0,355182863	0,34388189
Angle of list		
	GG1	Angle of list
Y-retning	0,000000	0
X-retning	0,000000	0

SDR41	SDR26	SDR17	
2,33	3,5	5,4	
1,88	3	4,26	

Fresh water				Sea water			
GZ T	GZ L	GZ T	GZ L	GZ T	GZ L	GZ T	GZ L
0	0	0	0	0	0	0	0
1	0,007083913	0,006198796	0,006909278	0,006001567			
2	0,014165667	0,012395703	0,01381645	0,012001305			
3	0,021243107	0,018588835	0,020719415	0,017997388			
4	0,028314076	0,024776304	0,027616068	0,023987988			
5	0,03537642	0,030956226	0,034504308	0,029971282			
6	0,042427988	0,037126719	0,041382039	0,035945446			
7	0,049466632	0,043285902	0,048247164	0,04190866			
8	0,056490208	0,0494319	0,055097593	0,047859109			
9	0,063496577	0,055562841	0,061931238	0,05379498			
10	0,070483604	0,061676857	0,068746018	0,059714464			
11	0,077449161	0,067772085	0,075539858	0,065615758			
12	0,084391126	0,07384667	0,082310687	0,071497065			
13	0,091307385	0,079898759	0,089056444	0,077356594			
14	0,09819583	0,085926511	0,095775074	0,083192559			
15	0,105054365	0,091928089	0,102464529	0,089003182			
16	0,111880898	0,097901665	0,109122773	0,094786695			
17	0,118673352	0,103845419	0,115747777	0,100541334			

Stability in fresh and salt water



Platform Basic Stability
Assumption: floating on columns (zero deck in top 0,0,0)
Stage 1: floating only battery pontoon
Stage 2: floating columns
* Calculation only for Stage 2!
GZ - Small angles (±17°)

Watertight Volume *				
Stage 1	Bat. A	Volume	Fresh	Sea
		0,02010619	0,020106193	0,020608848
	Bat. B	0,02010619	0,020106193	0,020608848
	Total Disp		0,040212386	0,041217696
	Kg Left		52,79561403	51,79030438
OK - Valid - Stage 2			OK - Valid - Stage 2	
Stage 2	Col 1	0,01526814		
	Col 2	0,01526814		
	Col 3	0,01526814		
	Col 4	0,01526814		
	Col 5	0,01526814		
	Col 6	0,01526814		
	Total	0,09160884	91,60884178	93,89906282
		%	57,63 %	55,16 %
	Draft	0,505789422	0,490931765	

Structure	x	y	z	mass
Col 1	0,404	0	-0,3	1,398
Col 2	0,2	0,35	-0,3	1,398
Col 3	-0,2	0,35	-0,3	1,398
Col 4	-0,404	0	-0,3	1,398
Col 5	-0,2	-0,35	-0,3	1,398
Col 6	0,2	-0,35	-0,3	1,398
Deck	0	0	0	10,8
Base	0	0	-0,6	17,9
Bat. A	0	-0,35	-0,68	21,88
Bat. B	0	0,35	-0,68	21,88
Mass 1	0	0	0	10
Div	0	0	0	0
Festeklemme	0	0	-0,68	2,16
Ventilblokk	0	0	0	0
Water col 1	0,41	-0,335	-0,4	0
Water col 2	0,41	0,335	-0,4	0
Water col 3	0	-0,335	-0,4	0
Water col 4	0	0,335	-0,4	0
Water col 5	-0,41	-0,335	-0,4	0
Water col 6	-0,41	0,335	-0,4	0
Total	0	0	-0,478259935	93,008

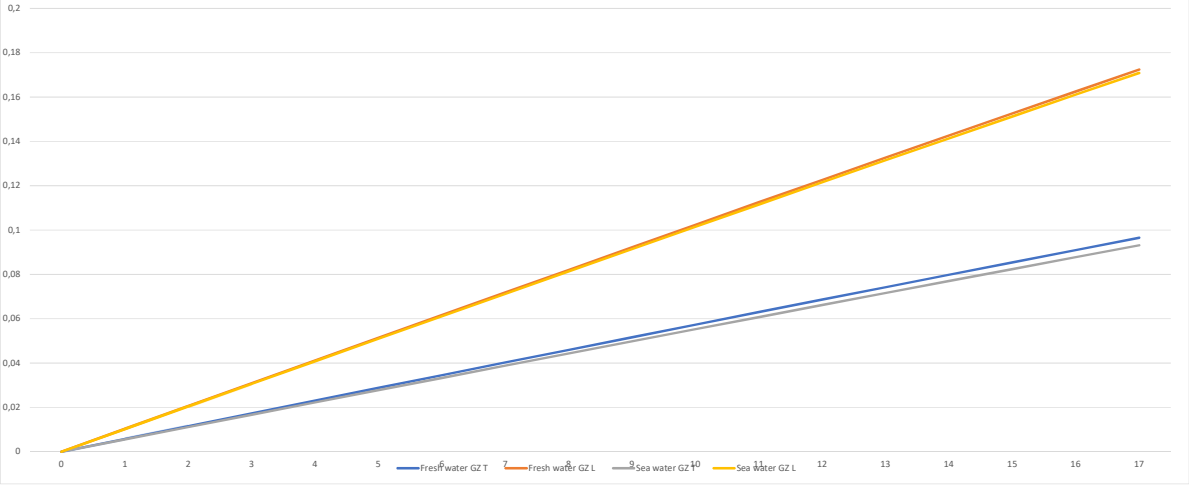
Num. Columns	Diameter	Lenght C	Lenght P	Breadth
6	0,18	0,6	1	0,85
Density	Sea	Fresh	Weight pipe	
	1,025	1		2,33
Battery weight pr pontoon	20		Total Volume	1,88
			131,8212277	

Inertia Columns	
Original I T	0,00005153
Transf T	0,002855778
I T total	0,011732293
Original I Lon	0,00005153
Transf Long	0,008906415
I Long total	0,035873417
Submerged Volume	
Fresh	Sea
0,093008	0,090739512

Stability partly submerged		
	Fresh	Sea
	0,505789422	0,490931765
KB	0,301740065	0,301740065
KG	0,126142842	0,129296413
BM Trans.	0,385702486	0,395345048
BM Long.		
GMT	0,330192199	0,318488113
GML	0,589751842	0,584536748
Angle of list		
	GG1	Angle of list
	0,000000	0
Y-retning		
X-retning	0,000000	0

Fresh water		Sea water	
GZ T	GZ L	GZ T	GZ L
0	0	0	0
1	0,005762648	0,010292589	0,005558384
2	0,011523542	0,020582042	0,011115075
3	0,017280924	0,030865227	0,01666838
4	0,023033043	0,041139009	0,022216608
5	0,0287778146	0,05140026	0,027758068
6	0,034514483	0,061645854	0,033291073
7	0,040240306	0,07187267	0,038813937
8	0,045953872	0,082077593	0,044324978
9	0,05165344	0,092257514	0,049822518
10	0,057337274	0,102409333	0,05530488
11	0,063003642	0,112529957	0,060770397
12	0,068650818	0,122616303	0,066217402
13	0,074277083	0,132665299	0,071644237
14	0,079880723	0,142673884	0,077049248
15	0,08546003	0,152639009	0,082430789
16	0,091013305	0,162557638	0,087787221
17	0,096538856	0,172426751	0,093116913

Stability in fresh and salt water



ServerSocketWorkerReceive

```
/*
 * 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 serialgpsserver;

import java.io.*;
import java.net.Socket;
import java.util.concurrent.Semaphore;
import java.util.logging.Level;
import java.util.logging.Logger;
import org.json.*;

/**
 *
 * @author Matias
 */
public class ServerSocketWorkerReceive implements Runnable {
    //Felt

    private Socket socket = null;
    private StorageBoxSocketReceive storageBoxSocketReceive;
    private Semaphore semaphoreStorageBoxSocketSend;
    private int portNumber;
    private boolean available;
    private JSONObject jsonWaypointRecieved;
    /**
     * Constructor
     * @param storageBox
     * @param semaphore
     * @param portNumber
     * @param socket
     */
    public ServerSocketWorkerReceive(StorageBoxSocketReceive
storageBoxSocketReceive, Semaphore semaphoreStorageBoxSocketSend, int
portNumber, Socket socket) {
        //super("ServerSocketWorker"); // Hva skjer her?

        this.socket = socket;
        this.storageBoxSocketReceive = storageBoxSocketReceive;
        this.semaphoreStorageBoxSocketSend = semaphoreStorageBoxSocketSend;
        this.portNumber = portNumber;
        this.available = false;

    }

    /**
     * Run metode, overrider Thread klassens run()
     */
    @Override
    public void run() {
```

```

        ServerSocketWorkerReceive
        boolean finished = false;
        try {
            while (!finished) {
                InputStreamReader inRead = new
InputStreamReader(this.socket.getInputStream()); //leser input stream på socket
                BufferedReader bufRead = new BufferedReader(inRead); //buffer på
in stream

                String jsonCoordMsg = bufRead.readLine(); // lagrer input på
socket i "msg"
                if(jsonCoordMsg != null && jsonCoordMsg.startsWith("{")){
                    this.jsonWaypointRecieved = new JSONObject(jsonCoordMsg);
                    System.out.println("Recived from client: " +jsonCoordMsg);
                }
                if(jsonWaypointRecieved != null &&jsonWaypointRecieved.length()>
0){

                    System.out.println("Rec acquirer");
                    semaphoreStorageBoxSocketSend.acquire();
                    System.out.println("Rec Putter");
                    storageBoxSocketReceive.putWaypointCoord(jsonWaypointRecieved);
                    semaphoreStorageBoxSocketSend.release();
                    System.out.println("Rec release");
                }

//                if (jsonCoordMsg != null) {
//                    PrintStream printStream = new
PrintStream(socket.getOutputStream()); //Skriver til output mot klient at msg er
mottat
//                    printStream.println("Message recieved @ server");
//
//
//
//                    if (jsonCoordMsg.equals("bye")) {
//                        System.out.println("Shutting down server...");
//                        finished = true;
//                    }
//                }

            } // Avslutter Server(test)
        } catch (IOException ex) {

Logger.getLogger(ServerSocketWorkerReceive.class.getName()).log(Level.SEVERE,
null, ex);
        } catch (JSONException ex) {

Logger.getLogger(ServerSocketWorkerReceive.class.getName()).log(Level.SEVERE,
null, ex);
        } catch (InterruptedException ex) {

Logger.getLogger(ServerSocketWorkerReceive.class.getName()).log(Level.SEVERE,

```

ServerSocketWorkerReceive

```
    null, ex);  
    }  
    }  
}
```

ServerSocketWorkerSend

```
/*
 * 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 serialgpsserver;

import java.net.*;
import java.io.*;
import java.util.concurrent.Semaphore;
import java.util.logging.Level;
import java.util.logging.Logger;
import org.json.*;

/**
 *
 * @author Matias
 */
public class ServerSocketWorkerSend implements Runnable {

    //Felt
    private Socket socket = null;
    private StorageBoxSocketSend storageBoxSocketSend;
    private Semaphore semaphoreStorageBoxSocketSend;
    private int portNumber;
    private boolean available;
    private long time;
    private long testTime;

    public ServerSocketWorkerSend(StorageBoxSocketSend storageBoxSocketSend,
    Semaphore semaphoreStorageBoxSocketSend, int portNumber, Socket socket) {
        //super("ServerSocketWorker");
        this.socket = socket;
        this.storageBoxSocketSend = storageBoxSocketSend;
        this.semaphoreStorageBoxSocketSend = semaphoreStorageBoxSocketSend;
        this.portNumber = portNumber;
        this.available = false;
    }

    @Override
    public void run() {
        boolean finished = false; //Boolean for while loop i run() så den kan
        avsluttes uten "break;"

        try {
            PrintWriter outputStreamWrite = new
PrintWriter(this.socket.getOutputStream(), true);
            // double p = 62.467067;
            while (!finished) {
                JSONObject jsonData = new JSONObject();
                semaphoreStorageBoxSocketSend.tryAcquire();
            }
        }
    }
}
```

```

ServerSocketWorkerSend
//System.out.println("stuck on
acquire!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!");

        available = storageBoxSocketSend.getAvailable();
//System.out.println("send Acq");
time = System.currentTimeMillis();
if(time > testTime){
    if (available) {
        jsonData = storageBoxSocketSend.getGPSdata();
        outputStreamWrite.println(jsonData);
        System.out.println("JSONData_Sent_To_Client: " + jsonData);
    }
    testTime = time +1000;
}
semaphoreStorageBoxSocketSend.release();
}

        socket.close();
    } catch (SocketException se) {
        System.out.println("Client closed connection to server. Ending
client communication.." + se);
    } catch (IOException e) {
        e.printStackTrace();
    }
    // catch (InterruptedException ex) {
    //
    Logger.getLogger(ServerSocketWorkerSend.class.getName()).log(Level.SEVERE, null,
ex);
    //
    //
    // catch (JSONException ex) {
    //
    Logger.getLogger(ServerSocketWorkerSend.class.getName()).log(Level.SEVERE, null,
ex);
    //
    }
}

```



```

StorageBoxArduinoComm

/*
 * 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 serialgpsserver;

import org.json.JSONObject;

/**
 *
 * @author Matias
 */
public class StorageBoxArduinoComm {

    private boolean available = false; //Flag
    private JSONObject jsonArduinoData;

    /**
     * Boolean metode for å sjekke om koordinater er tilgjengelig
     *
     * @return true if coordinates are available
     */
    public boolean getAvailable() {
        return available;          //Returnerer flag
    }

    /**
     * Putter GPS koordinater i variabel som String
     *
     * @param coords Koorinater lagret som en string
     */
    public void putArduinoData(JSONObject jsonReceived) {
        if (available == false) {
            this.jsonArduinoData = jsonReceived; //Lagrer koordinater som
JSONObject
        }
        available = true;          //Setter flag true
    }

    public JSONObject getArduinoData() {
        if (available == true) {
            available = false;
        }
        return this.jsonArduinoData;
    }

}

```

StorageBoxGPS

```
/*
 * 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 serialgpsserver;

import java.util.logging.Level;
import java.util.logging.Logger;
import org.json.JSONException;
import org.json.JSONObject;

/**
 *
 * @author Matias
 */
public class StorageBoxGPS {
//Felt

    private boolean available = false; //Flag
    private double lat;
    private double lon;
    private double speed;

    /**
     * Boolean metode for å sjekke om koordinater er tilgjengelig
     *
     * @return true if coordinates are available
     */
    public boolean getAvailable() {
        return available; //Returnerer flag
    }

    /**
     * Putter GPS koordinater i variabel som JSONObject Putter feltene; lat,
     * lon, speed, henter disse ut som JSONObject
     *
     * @param jsonObj "leash" fra Conductor klassen
     */
    public void getGPSCoord(JSONObject jsonObj) {
        if (available == true) {
            try {
                //Lagrer koordinater og speed som double
                jsonObj.put("lat", lat);

                jsonObj.put("lon", lon);
                jsonObj.put("speed", speed);
            } catch (JSONException ex) {

                Logger.getLogger(StorageBoxGPS.class.getName()).log(Level.SEVERE, null, ex);
            }
        }
    }
}
```

```

StorageBoxGPS
    }
    available = false;           //Setter flag true
}

/**
 * Putter datafelter fra GPSRead klassen
 *
 * @param latitude, double
 * @param longitude, double
 * @param speed, double
 */
public void putGPSCoord(Double latitude, Double longitude, Double speed) {
    if (available == true) { //Sjekker om Available = true
        available = false;   // setter available = false før koden
fortsetter.
    }
    this.lat = latitude;
    this.lon = longitude;
    this.speed = speed;

    available = true;
}
}

```

StorageBoxNorthEastGPS

```
/*
 * 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 serialgpsserver;

import java.util.logging.Level;
import java.util.logging.Logger;
import org.json.JSONException;
import org.json.JSONObject;

/**
 *
 * @author Matias
 */
public class StorageBoxNorthEastGPS {
//Felt

    private boolean available = false; //Flag
    private double lat;
    private double lon;
    private double speed;

    private JSONObject northEastJson;

    /**
     * Boolean metode for å sjekke om koordinater er tilgjengelig
     *
     * @return true if coordinates are available
     */
    public boolean getAvailable() {
        return available; //Returnerer flag
    }

    /**
     * Putter GPS koordinater i variabel som JSONObject Putter feltene; lat,
     * lon, speed, henter disse ut som JSONObject
     *
     * @param jsonObj "leash" fra Conductor klassen
     */
    public void getGPSCoord(JSONObject jsonObj) {
        if (available == true) {
            try {
                //Lagrer koordinater og speed som double
                jsonObj.put("latNE", lat);
                jsonObj.put("lonNE", lon);
                jsonObj.put("speedNE", speed);
            } catch (JSONException ex) {

                Logger.getLogger(StorageBoxGPS.class.getName()).log(Level.SEVERE, null, ex);
            }
        }
    }
}
```

StorageBoxNorthEastGPS

```
    }
    available = false;           //Setter flag true
}

/**
 * Putter datafelter fra GPSRead klassen
 *
 * @param latitude, double
 * @param longitude, double
 * @param speed, double
 */
public void putGPSCoord(Double latitude, Double longitude, Double speed) {
    if (available == true) { //Sjekker om Available = true
        available = false;   // setter available = false før koden
fortsetter.
    }
    this.lat = latitude;
    this.lon = longitude;
    this.speed = speed;

    available = true;
}

public void putGPSCoord(JSONObject xyNorthEast) {
    if(available == false){
        this.northEastJson = xyNorthEast;
    }
    available = true;
}

public JSONObject getNorthEastJson(){
    if(available == true){
        available = false;
    }
    return this.northEastJson;
}
}
```

StorageBoxPlatformMode

```
/*
 * 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 serialgpsserver;

import java.util.logging.Level;
import java.util.logging.Logger;
import org.json.JSONException;
import org.json.JSONObject;

/**
 *
 * @author Matias
 */
public class StorageBoxPlatformMode {
//Felt

    private boolean available = false; //Flag
//    private double lat;
//    private double lon;
//    private double speed;
    private int mode = 0;
    private JSONObject jsonWPData = new JSONObject();
    private boolean hasArrived;

    /**
     * Boolean metode for å sjekke om ny koordinat er tilgjengelig
     *
     * @return true if coordinates are available
     */
    public synchronized boolean getAvailable() {
        return available; //Returnerer flag
    }
    /**
     * Putter waypoint data fra GUI
     *
     * @param jsonData
     */
    public void putGPSWp(JSONObject jsonData) {
        if (available == false) {
            this.jsonWPData = jsonData; //Lagrer koordinater som JSONObject
        }
        available = true; //Setter flag true
    }

    /**
     * Henter waypoint koordinater satt i GUI
     * @return
     */
    public int getMode() {
```

```

        StorageBoxPlatformMode

        try {
            if(jsonWPData.length() > 0 && jsonWPData.has("Mode")) mode =
jsonWPData.getInt("Mode");
            else System.out.println("No mode recived");
        } catch (JSONException ex) {

Logger.getLogger(StorageBoxPlatformMode.class.getName()).log(Level.SEVERE, null,
ex);
        }
        return mode;
    }

    /**
     * *****Test Metoder *****
     */
    /**
     * Putter waypoint data fra GUI
     *
     * @param jsonData
     */
    public void putGPSCoord(JSONObject jsonData) {
        if (available == false) {
            this.jsonWPData = jsonData; //Lagrer koordinater som JSONObject
        }
        available = true;                //Setter flag true
    }

    /**
     * Henter waypoint koordinater satt i GUI
     * @return
     */
    public JSONObject getGPSdata(){
        if (available == true) {
            available = false;
        }
        //System.out.println("Data in PM storage box: " +jsonWPData);
        return this.jsonWPData;
    }

    public void putData(String key, String data){
        if (available == false) {
            try {
                this.jsonWPData.put(key, data);
            } catch (JSONException ex) {

Logger.getLogger(StorageBoxPlatformMode.class.getName()).log(Level.SEVERE, null,
ex);
            }
        }
        available = true;
    }
}

```

StorageBoxPlatformMode

```
public void putDataInt(String key, int data){
    if (available == false) {
        try {
            this.jsonWPData.put(key, data);
        } catch (JSONException ex) {
            Logger.getLogger(StorageBoxPlatformMode.class.getName()).log(Level.SEVERE, null,
ex);
        }
    }
    available = true;
}
}
```



```

                                StorageBoxRunningMode
/*
 * 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 serialgpsserver;

import java.util.logging.Level;
import java.util.logging.Logger;
import org.json.JSONException;
import org.json.JSONObject;

/**
 *
 * @author Bakken
 */
public class StorageBoxRunningMode {

    private boolean available = false; //Flag

    private JSONObject jsonManualData; // Object to store Manual data
    private boolean hasArrived;
    private JSONObject jsonWPData;

    /**
     * Boolean method to check for new incoming data
     *
     * @return true if data are available
     */
    public boolean getAvailable() {
        return available;          //Returner flag
    }

    /**
     * Method to put data i JSONObject
     */
    public void putManualData(JSONObject jsonReceived) {
        if (available == false) {
            this.jsonManualData = jsonReceived; //Lagrer data som JSONObject
        }
        available = true;          //Setter flag true
    }

    public JSONObject getManualData() {
        if (available == true) {
            available = false;
        }
        return this.jsonManualData;
    }
    /**
     * Return true if the platform has arrived the waypoint received from
     client.

```

```

StorageBoxRunningMode

* @return hasArrived
*/
public boolean hasArrivedToWaypoint(){
return hasArrived;
}

public void setHasArrivedToWaypoint(boolean value){
    this.hasArrived = value;
}

/**
 * Putter waypoint data fra GUI
 *
 * @param jsonData
 */
public void putGPSWp(JSONObject jsonData) {
    if (available == false) {
        this.jsonWPData = jsonData; //Lagrer koordinater som JSONObject
    }
    available = true;                //Setter flag true
}

/**
 * Henter waypoint koordinater satt i GUI
 * @return
 */
public JSONObject getGPSdata(){
    if (available == true) {
        available = false;
    }
    //System.out.println("Data in PM storage box: " +jsonWPData);
    return this.jsonWPData;
}

public void putData(String key, double data){
    if (available == false) {
        try {
            this.jsonWPData.put(key, data);
        } catch (JSONException ex) {

        }
    }
    available = true;
}
}
}

```

StorageBoxSocketReceive

```
/*
 * 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 serialgpsserver;

import org.json.JSONObject;

/**
 *
 * @author Matias
 */
class StorageBoxSocketReceive {

    private boolean available = false; //Flag
    private String coordinates;        // Variabel for å lagre koordinater
    private JSONObject jsonWaypointData;

    /**
     * Boolean metode for å sjekke om koordinater er tilgjengelig
     *
     * @return true if coordinates are available
     */
    public boolean getAvailable() {
        return available;        //Returnerer flag
    }

    /**
     * Putter GPS koordinater i variabel som String
     *
     * @param coords Koorinater lagret som en string
     */
    public void putWaypointCoord(JSONObject jsonReceived) {
        if (available == false) {
            this.jsonWaypointData = jsonReceived; //Lagrer koordinater som
JSONObject
            System.out.println("Putter i stbox");
        }
        available = true;        //Setter flag true
    }

    public JSONObject getWaypointData() {
        if (available == true) {
            available = false;
        }
        return this.jsonWaypointData;
    }

    public void getWaypointData(JSONObject jsonWaypoint) {
        if (available == true) {
            available = false; //Setter flag true
        }
    }
}
```

```
StorageBoxSocketReceive
    jsonWaypoint.equals(this.jsonWaypointData);
}
// this.jsonWaypointData = jsonWaypoint;
}
}
```

StorageBoxSocketSend

```
/*
 * 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 serialgpsserver;

import org.json.JSONObject;

/**
 *
 * @author Matias
 */
public class StorageBoxSocketSend {

    private boolean available = false; //Flag
    private JSONObject jsonData;      // Variabel for å lagre koordinater

    /**
     * Boolean metode for å sjekke om koordinater er tilgjengelig
     *
     * @return true if coordinates are available
     */
    public boolean getAvailable() {
        return available;      //Returnerer flag
    }

    /**
     * Putter GPS koordinater i variabel som String
     *
     * @param coords Koorinater lagret som en string
     */
    public void putGPSCoord(JSONObject jsonData) {
        if (available == false) {
            this.jsonData = jsonData; //Lagrer koordinater som JSONObject
        }
        available = true;      //Setter flag true
    }

    public JSONObject getGPSdata() {
        if (available == true) {
            available = false;
        }
        return this.jsonData;
    }

    /**
     * Henter ut Koordinater som string
     * @return
     */
    public String getGPSCoord() {
        if (available == true) { //Sjekker om Available = true

```

```
StorageBoxSocketSend
//          available = false;  // setter available = false før koden
fortsetter.
//      }
//      return coordinates;      //Returnerer Koordinat info som en string
//  }
}
```

ThrustAllocator

```
/*
 * 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 serialgpsserver;

import com.joptimizer.functions.ConvexMultivariateRealFunction;
import com.joptimizer.functions.LinearMultivariateRealFunction;
import com.joptimizer.functions.PDQuadraticMultivariateRealFunction;
import com.joptimizer.optimizers.JOptimizer;
import com.joptimizer.optimizers.OptimizationRequest;
import com.joptimizer.optimizers.OptimizationResponse;
import org.apache.commons.math3.linear.ArrayRealVector;
import org.apache.commons.math3.linear.BlockRealMatrix;
import org.apache.commons.math3.linear.RealVector;

/**
 *
 * @author vegard
 */
public class ThrustAllocator {

    private double[][] Phi;
    private double[][] A1;
    private BlockRealMatrix A2;
    private BlockRealMatrix C2;
    private ArrayRealVector p;
    private PDQuadraticMultivariateRealFunction objectiveFunction;
    ConvexMultivariateRealFunction[] inequalities;
    private JOptimizer opt;

    // NOTE: Lx1 er negativ, Lx2 er positiv, Ly1 er negativ, Ly2 er positiv
    public ThrustAllocator(double Lx1, double Lx2, double Ly1, double Ly2) {
        setUp(Lx1, Lx2, Ly1, Ly2);
    }

    /**
     * Konstruktør for Plattform
     */
    public ThrustAllocator() {
        setUp(-0.45, 0.45, -0.19, 0.19);
    }

    /**
     * Sett opp matriser og vektorer
     */
    private void setUp(double Lx1, double Lx2, double Ly1, double Ly2) {
        Phi = new double[][]{{1., 0., 0., 0., 0., 0., 0.},
                               {0., 1., 0., 0., 0., 0., 0.},
                               {0., 0., 1., 0., 0., 0., 0.},
                               {0., 0., 0., 1., 0., 0., 0.},
                               {0., 0., 0., 0., 1., 0., 0.},
                               {0., 0., 0., 0., 0., 1., 0.},
                               {0., 0., 0., 0., 0., 0., 1.}};
    }
}
```

```

        ThrustAllocator
        {0., 0., 1., 0., 0., 0., 0.},
        {0., 0., 0., 1., 0., 0., 0.},
        {0., 0., 0., 0., 10., 0., 0.},
        {0., 0., 0., 0., 0., 10., 0.},
        {0., 0., 0., 0., 0., 0., 10.}};

A1 = new double[][]{
    {1., 1., 0., 0., -1., 0., 0.},
    {0., 0., -1., -1., 0., -1., 0.},
    {-Ly1, -Ly2, -Lx1, -Lx2, 0., 0., -1.}};

A2 = new BlockRealMatrix(new double[][]{
    {-1., 0., 0., 0., 0., 0., 0.},
    {0., -1., 0., 0., 0., 0., 0.},
    {0., 0., -1., 0., 0., 0., 0.},
    {0., 0., 0., -1., 0., 0., 0.},
    {1., 0., 0., 0., 0., 0., 0.},
    {0., 1., 0., 0., 0., 0., 0.},
    {0., 0., 1., 0., 0., 0., 0.},
    {0., 0., 0., 1., 0., 0., 0.}});

C2 = new BlockRealMatrix(new double[][]{
    {0., 0., 0., -1., 0., 0., 0., 0., 0., 0., 0.},
    {0., 0., 0., 0., -1., 0., 0., 0., 0., 0., 0.},
    {0., 0., 0., 0., 0., -1., 0., 0., 0., 0., 0.},
    {0., 0., 0., 0., 0., 0., -1., 0., 0., 0., 0.},
    {0., 0., 0., 0., 0., 0., 0., 1., 0., 0., 0.},
    {0., 0., 0., 0., 0., 0., 0., 0., 1., 0., 0.},
    {0., 0., 0., 0., 0., 0., 0., 0., 0., 1., 0.},
    {0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 1.}});

//Skyvkraft i NEWTON. Dataene er omgjort fra Kg til N
//Original:
//p = new ArrayRealVector(new double[]{0., 0., 0., -29., -29., -29.,
-29., 34., 34., 34., 34.});
p = new ArrayRealVector(new double[]{0., 0., 0., -110., -110., -110.,
-110., 110., 110., 110., 110.});

objectiveFunction = new PDQuadraticMultivariateRealFunction(Phi, null,
0);
inequalities = new ConvexMultivariateRealFunction[8];
opt = new JOptimizer();
}

/*
Beregn output basert på en ønsket kraftvektor
*/
public double[] calculateOutput(double[] tau) throws Exception {
//    long time1 = System.currentTimeMillis();
//    Sett opp p-vektoren med ønskede verdier for tau (kraftvektor)
p.setEntry(0, tau[0]);
p.setEntry(1, tau[1]);

```



```

                                ThrustAllocator
p.setEntry(2, tau[2]);

// Oppsett for ulikheten  $A2 \cdot z \leq C2 \cdot p$ 
RealVector v2 = C2.operate(p);

for (int i = 0; i < 8; i++) {
    // Hver ulikhet settes opp som et LinearMultivariateRealFunction
    inequalities[i] = new LinearMultivariateRealFunction(A2.getRow(i),
-v2.toArray()[i]);
}
// Optimaliseringsproblemet
OptimizationRequest or = new OptimizationRequest();
// Sett objektivfunksjonen
or.setF0(objectiveFunction);

// Sett ulikheten
or.setFi(inequalities);

// Sett likheten
or.setA(A1);
or.setB(p.getSubVector(0, 3).toArray());

// Sett toleransen på resultatet. Lavere tall = større nøyaktighet
or.setTolerance(1.E-1);

// Optimalisering
opt.setOptimizationRequest(or);
int returnCode = opt.optimize();

if (returnCode == OptimizationResponse.FAILED) {
    System.out.println("Optimization FAIL");
}
// long time2 = System.currentTimeMillis()-time1;
//System.out.println("Tidsbruk = " + time2);
return opt.getOptimizationResponse().getSolution();
}
}

```

ThrustWriter

```
package serialgpsserver;

import java.util.logging.Level;
import java.util.logging.Logger;
import org.json.JSONException;
import org.json.JSONObject;

/**
 *
 * @author root
 */
public class ThrustWriter {

    private int pulseWidth1;
    private int pulseWidth2;
    private int pulseWidth3;
    private int pulseWidth4;

    private int i = 0;

    private SerialHandler thrustArduino;
    private JSONObject thrustValuesToArduino;

    /**
     * Klasse for å konvertere newton til pwm og skrive verdier
     *
     * @param serialConnection
     * @param ID
     */
    public ThrustWriter(SerialHandler thrustArduino) {
        pulseWidth1 = 0;
        pulseWidth2 = 0;
        pulseWidth3 = 0;
        pulseWidth4 = 0;

        this.thrustArduino = thrustArduino;
        thrustValuesToArduino = new JSONObject();
    }

    /**
     * skriver verdier via SerialConnection
     */
    public void writeThrust() {
        try {
            pulseWidth2 = (pulseWidth2*4);
            pulseWidth4 = (pulseWidth4*4);
            if(pulseWidth2>3200){
                pulseWidth2 = 3200;
            }
            if(pulseWidth4>3200){
                pulseWidth4 = 3200;
            }
        }
    }
}
```

```

        ThrustWriter
        thrustValuesToArduino.put("smcSerial1", (-pulseWidth4));
        thrustValuesToArduino.put("smcSerial2", (-pulseWidth2));
        thrustValuesToArduino.put("smcSerial3", (-pulseWidth3));
        thrustValuesToArduino.put("smcSerial4", (-pulseWidth1));
        System.out.println(thrustValuesToArduino);
        thrustArduino.send(thrustValuesToArduino);

    } catch (JSONException ex) {
        Logger.getLogger(ThrustWriter.class.getName()).log(Level.SEVERE,
null, ex);
    }
}

/**
 * setter pwm variabelene
 *
 * @param newton
 */
public void setThrustForAll(double[] newton) {
    pulseWidth1 = newtonToPulseWidth(newton[0]);
    pulseWidth2 = newtonToPulseWidth(newton[1]);
    pulseWidth3 = newtonToPulseWidth(newton[2]);
    pulseWidth4 = newtonToPulseWidth(newton[3]);
}

/**
 * konverterer newton til pwm
 *
 * @param xNewton
 * @return
 */
public int newtonToPulseWidth(double xNewton) {

    int pulseWidth = 0;

    if (xNewton > 0.0) {
        double pulseWidthDouble = 32*xNewton;
        pulseWidth = (int) pulseWidthDouble;
    } else if (xNewton <= 0.0) {

        double pulseWidthDouble = 32*xNewton;
        pulseWidth = (int) pulseWidthDouble;
    } else {
        pulseWidth = 0;
    }

    if (pulseWidth < -3200) {
        pulseWidth = -3200;
    }
    if (pulseWidth > 3200) {
        pulseWidth = 3200;
    }
}

```

```

                                ThrustWriter
        }
        return pulseWidth;
    }

    /**
     * lukker seriell forbindelsen
     */
    void closeSerialConn() {
        thrustArduino.close();
        System.out.println("ThrustWriter: Connection closed");
    }
}

```

VideostreamUDP

```
/*
 * 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 serialgpsserver;

import java.util.concurrent.Semaphore;
import java.util.logging.Level;
import java.util.logging.Logger;
import no.ntnu.videostream.ImageStorageBox;
import no.ntnu.videostream.UDPCameraStream;
import org.json.JSONObject;
import org.opencv.core.Core;
import org.opencv.core.Mat;
import org.opencv.highgui.VideoCapture;

/**
 *
 * @author Matias
 */
public class VideostreamUDP extends Thread {

    static {
        System.loadLibrary(Core.NATIVE_LIBRARY_NAME);
    }

    private boolean isManualMode;
    private Semaphore semaphoreVideoStream;
    private ImageStorageBox imageStorageBox;

    public VideostreamUDP(Semaphore semaphore, ImageStorageBox imgStorageBox) {
        this.semaphoreVideoStream = semaphore;
        this.imageStorageBox = imgStorageBox;
        // String ipAddress = "127.0.0.1";
        String ipAddress = "192.168.0.101";
        int portNumber = 5555;
        UDPCameraStream videoStream = new UDPCameraStream(semaphore, imgStorageBox,
        ipAddress, portNumber);
        videoStream.setName("UDP Camera Stream Thread");
        videoStream.start();
    }

    @Override
    public void run(){
        Mat webcamMatImage = new Mat();
        VideoCapture capture = new VideoCapture(0);
        int i = 0;
        isManualMode = true;

        while (isManualMode) {
```

```

                                VideostreamUDP
i++;
try {
    capture.read(webcamMatImage);
    semaphoreVideoStream.acquire();
    if (!webcamMatImage.empty()) {
        imageStorageBox.putImage(webcamMatImage);
    }
    semaphoreVideoStream.release();
} catch (InterruptedException ex) {

Logger.getLogger(VideostreamUDP.class.getName()).log(Level.SEVERE, null, ex);
    }
    // System.out.println("Count = " + i );

}

}

```

ArduinoHandler

```
/*
 * 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 serialgpsserver;

import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.io.OutputStream;
import gnu.io.CommPortIdentifier;
import gnu.io.SerialPort;
import gnu.io.SerialPortEvent;
import gnu.io.SerialPortEventListener;
import java.io.IOException;
import java.util.ArrayList;
import java.util.Enumeration;
import java.util.HashMap;
import java.util.List;
import java.util.concurrent.Semaphore;
import java.util.logging.Level;
import java.util.logging.Logger;
import org.json.JSONException;
import org.json.JSONObject;

/**
 *
 * @author Bakken, Per Martin, Matias
 */
public class ArduinoHandler {

    private Enumeration portList;
    private CommPortIdentifier portId;
    private HashMap<String, CommPortIdentifier> comList;

    private ArrayList<String> listOfPorts = new ArrayList<String>();

    private SerialHandler StabilityArduino;
    private SerialHandler remoteOperation;

    SerialPort serialPort;
    StorageBoxRunningMode storageBoxManualMode;
    StorageBoxArduinoComm storageBoxArduinoComm;
    private byte[] b = new byte[2];
    private String[] d;

    String[] sa = new String[2];

    private JSONObject jsonRemoteControl;

    private Semaphore semaphore;
    Boolean available;
```

ArduinoHandler

```
private static final String PORT_NAMES[] = {
    "/dev/ttyACM0",
    "COM4",};

private BufferedReader input;
private OutputStream output;
//private DataOutputStream output;
private static final int TIME_OUT = 2000;
private static final int DATA_RATE = 9600;
private SerialHandler LidarArduino;

private String stabilityArd = "/dev/stabilityArduino";
private boolean stabilityArdConnected = false;
private String remoteArd = "/dev/thrustArduino";
private boolean remoteArdConnected = false;
private ArrayList<String> comName;

//{
//
//    }
public ArduinoHandler() {
    comName = new ArrayList<>();
    comName.add(stabilityArd);
    comName.add(remoteArd);
    for(String name : comName){

        System.setProperty("gnu.io.rxtx.SerialPorts", name);

        comList = new HashMap<>();
        portList = CommPortIdentifier.getPortIdentifiers();
        while (portList.hasMoreElements()) {
            portId = (CommPortIdentifier) portList.nextElement();
            listOfPorts.add(portId.getName());
            comList.put(portId.getName(), portId);

            System.out.println(portId.getName());

        }
        connectToArduino();
    }
}

public synchronized void close() {
    if (serialPort != null) {
        serialPort.removeEventListener();
        serialPort.close();
    }
}

private void connectToArduino() {
```


ArduinoHandler

```
for (String temp : listOfPorts) {
    if (temp.equalsIgnoreCase(stabilityArd)) {
        CommPortIdentifier portId = comList.get(stabilityArd);
        StabilityArduino = new SerialHandler(stabilityArd, 9600,
portId);
        System.out.println("Connected to stability controll Arduino");
        stabilityArdConnected = true;
    }
    if(temp.equalsIgnoreCase(remoteArd)){
        CommPortIdentifier portId = comList.get(remoteArd);
        remoteOperation = new SerialHandler(remoteArd, 9600, portId);
        System.out.println("Connected to thruster controll Arduino");
        remoteArdConnected = true;
    }
    if(temp.equalsIgnoreCase("COM")){
        CommPortIdentifier portId = comList.get("COM");
        StabilityArduino = new SerialHandler("COM", 9600, portId);
    }
}

}

public SerialHandler getArduino(String name){
    SerialHandler temp = null;
    if(name.equalsIgnoreCase("StabilityArduino")){
        temp = StabilityArduino;
    }
    else if(name.equalsIgnoreCase("remoteOperation")){
        temp = remoteOperation;
    }
    else if(name.equalsIgnoreCase("LidarArduino")){
        temp = LidarArduino;
    }
    return temp;
}

public boolean isStabilityArdConnected(){
    return stabilityArdConnected;
}

public boolean isRemoteArdConnected(){
    return remoteArdConnected;
}

}
```

AutoPilot

```
package serialgpsserver;
```

```
/*  
 * 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.  
 */
```

```
//import java.io.BufferedReader;  
import java.util.concurrent.Semaphore;  
import java.util.logging.Level;  
import java.util.logging.Logger;  
import org.json.JSONException;  
import org.json.JSONObject;
```

```
/**  
 * Klasse for AutoPilot, "Følger waypoints"  
 *  
 * @author Matias  
 */
```

```
public class AutoPilot extends Thread {
```

```
    public static final double RADIUS = 6378137.0;  
    private static double distanceToWaypoint;  
    private final StorageBoxRunningMode StorageBoxRunningMode;  
    private final Semaphore semaphoreStorageBoxRunningMode;
```

```
    private JSONObject xyPosDataJson; //Felt for GPS rådata  
    //private JSONObject rawWaypointData; //Felt for Waypoint rådata  
    private final JSONObject headingAndVectLength; // Felt for kurs og avstand  
    private JSONObject jsonCourse;  
    //private JSONObject arduinoData;
```

```
    //private BufferedReader input;
```

```
    private double headingAngleWaypoint; //Felt for vinkel til waypoint i grader  
    private double platformHeading; // Vinkel på plattform ift sant nord  
    private double headingFromArduino;
```

```
    private double outputX;  
    private double outputY;  
    private double outputN;
```

```
    private double refLatBody = 62.000000;  
    private double refLonBody = 6.000000;
```

```
    private final ThrustAllocator thrustAllocator;  
    private final ArduinoHandler ardHandler;  
    private RotationMatrix Rz;  
    private ThrustWriter tWriter;  
    private double[] forceOutputNewton;
```

```
    private final PIDController xNorthPID;
```

```

                                AutoPilot
private final PIDController yEastPID;
private final PIDController headingPID;
private boolean isRunning;

private long time;
private long testTime;

/**
 * Autopilot Konstruktør
 *
 * @param storageBoxRunningMode
 * @param semaphoreStorageBoxRunningMode
 * @param ardHandler Kontakten med Arduinoer
 */
public AutoPilot(StorageBoxRunningMode storageBoxRunningMode, Semaphore
semaphoreStorageBoxRunningMode, ArduinoHandler ardHandler) {

    this.StorageBoxRunningMode = storageBoxRunningMode;
    this.semaphoreStorageBoxRunningMode = semaphoreStorageBoxRunningMode;
    this.ardHandler = ardHandler;
    this.headingAndVectLength = new JSONObject();
    this.jsonCourse = new JSONObject();
    forceOutputNewton = new double[4];
    thrustAllocator = new ThrustAllocator();
    if (ardHandler.isRemoteArdConnected()) {
        tWriter = new
ThrustWriter(ardHandler.getArduino("remoteOperation"));
    }
    xNorthPID = new PIDController();
    yEastPID = new PIDController();
    headingPID = new PIDController();
    //pid tunings
    xNorthPID.setTunings(30000, 1, 10);
    yEastPID.setTunings(30000, 1, 10);

}

/**
 * Method for å hente heading
 *
 * @return jsonobjekt med vinkel og lengde til ønsket waypoint
 */
public JSONObject calculateHeadingAndDistance() {
    try {
        double rawLat = xyPosDataJson.getDouble("xyLatBody"); //Rådata fra
GPS
        double rawLon = xyPosDataJson.getDouble("xyLonBody"); //Rådata fra
GPS
        double wpLat = xyPosDataJson.getDouble("xyLatWaypoint");
//Waypointdata fra app
        double wpLon = xyPosDataJson.getDouble("xyLonWaypoint");
//Waypointdata fra app

```

```

        AutoPilot
        double GPSSpeed = xyPosDataJson.getDouble("speed");
//        if(GPSSpeed < 0.5){
//
//            platformHeading = arduinoData.getDouble("Heading");
//        }
        double latitude1 = Math.toRadians(rawLat);
        double latitude2 = Math.toRadians(wpLat);
        double longDiff = Math.toRadians(wpLon - rawLon);
        double y = Math.sin(longDiff) * Math.cos(latitude2);
        double x = Math.cos(latitude1) * Math.sin(latitude2) -
Math.sin(latitude1) * Math.cos(latitude2) * Math.cos(longDiff);
        double vectorLength = Math.abs((wpLat - (rawLat)) + (wpLon -
(rawLon)));
        //Putter til jsonObjekt for senere h ntering
        headingAngleWaypoint = (Math.toDegrees(Math.atan2(y, x)) + 360) %
360;

        headingAndVectLength.put("heading", headingAngleWaypoint);
        headingAndVectLength.put("length", vectorLength);

    } catch (JSONException ex) {
        Logger.getLogger(AutoPilot.class.getName()).log(Level.SEVERE, null,
ex);
    }
    return headingAndVectLength;
}

@Override
public void run() {

    //Boolean for   bryte while
    isRunning = true;

    try {
        while (isRunning) {

            if (ardHandler.isStabilityArdConnected()) {
                getHedingFromArduino();
                // System.out.println("Autopilot: " +headingFromArduino);
            }

            time = System.currentTimeMillis();
            if (time > testTime) {
                testTime = time + 500;
                //Tar semaforen for storageBoxAutoPilot
                semaphoreStorageBoxRunningMode.acquire();
                //Sjekker at storageBoxAutoPilot har f tt koordinater fra
gps/gui

                boolean isAvailable = StorageBoxRunningMode.getAvailable();

                if (isAvailable) {
                    //Lagrer jsonobjektene i egne json objekter
                    xyPosDataJson = new JSONObject();

```

```

        AutoPilot
        xyPosDataJson = StorageBoxRunningMode.getGPSdata();
        platformHeading = bearing();
    }
    //haversine(double latPlatform, double lonPlatform, double
latWaypoint, double lonWaypoint)
        if (xyPosDataJson != null && xyPosDataJson.length() > 0 &&
(haversine(xyPosDataJson.getDouble("lat"), xyPosDataJson.getDouble("lon"),
xyPosDataJson.getDouble("waypointLat"), xyPosDataJson.getDouble("waypointLon")))
< 2) {
            StorageBoxRunningMode.setHasArrivedToWaypoint(true);
        } else {
            StorageBoxRunningMode.setHasArrivedToWaypoint(false);
        }
    //Slipper semaforen så storageBoxAutoPilot kan entres av
andre tråder.
        semaphoreStorageBoxRunningMode.release();
    //Sjekker om kriteriene for ny kurs er oppfylt, json
objektene må inneholde lat og lon
        if (xyPosDataJson != null && xyPosDataJson.length() > 0) {
            jsonCourse = calculateHeadingAndDistance();
            System.out.println("Heading: " +
jsonCourse.getString("heading") + " Degrees");
            System.out.println(bearing());
            //int led = testHeading(); // Test program for å kjøre
test med led og Arduino
                //Henter output fra hver PID regulator
                double X =
xNorthPID.computeOutput((xyPosDataJson.getDouble("xyLatBody")*100),
(xyPosDataJson.getDouble("xyLatWaypoint")*100), false);
                double Y =
yEastPID.computeOutput((xyPosDataJson.getDouble("xyLonBody")*100),
(xyPosDataJson.getDouble("xyLonWaypoint")*100), false);
                double N = headingPID.computeOutput(headingFromArduino,
headingAngleWaypoint, true);
                System.out.println("X output: " + X + " Y Output: " + Y
+" N output: " + N);
                //setter kreftene X og Y, og momentet N.
                setPIDOutputVector(X, Y, N);//synchronized

                Rz = new RotationMatrix(headingFromArduino); //HEADING
MÅ VÆRE FRA IMU
                double[] XYNtransformed = Rz.multiplyRzwithV(outputX,
outputY, outputN);
                forceOutputNewton =
thrustAllocator.calculateOutput(XYNtransformed);

                if (ardHandler.isRemoteArdConnected()) {
                    tWriter.setThrustForAll(forceOutputNewton);
                    tWriter.writeThrust();
                } else {
                    System.out.println("Arduino not connected");
                }
            }

```

```

                                AutoPilot
                                }
                                }
                                }
        } catch (InterruptedException ex) {
            Logger.getLogger(AutoPilot.class.getName()).log(Level.SEVERE, null,
ex);
        } catch (Exception ex) {
            Logger.getLogger(AutoPilot.class.getName()).log(Level.SEVERE, null,
ex);
            System.out.println("Error XYNtransformed output force");
        }
    }

    private double getHedingFromArduino() {

        SerialHandler stabArdu = ardHandler.getArduino("StabilityArduino");
        if (ardHandler.isStabilityArdConnected()) {
            double temp = stabArdu.getArduinoData();
            // headingFromArduino = Double.parseDouble(temp);

            headingFromArduino = temp;
            System.out.println("Get heading: " + headingFromArduino);
        }
        return headingFromArduino;
    }

    /**
     * Setter PID kreftene og momentet.
     *
     * @param X
     * @param Y
     * @param N
     */
    public synchronized void setPIDOutputVector(double X, double Y, double N) {
        outputX = X;
        outputY = Y;
        outputN = N;
    }

    public int testHeading() {
        double testHeading = bearing();
        System.out.print("GPS heading | ");
        System.out.println(testHeading);
        double diff = headingAngleWaypoint - testHeading;
        int led = 0;
        if (diff < 10 && diff > -10) {
            led = 1;
        } else if (diff > 10) {
            led = 2;
        } else if (diff < -10) {
            led = 3;
        }
    }

```

```

                                AutoPilot
        }
        return led;
    }

    /**
     * Method for å hente heading
     *
     * @return jsonobjekt med vinkel og lengde til ønsket waypoint
     */
    public double calculateHeading() {
        double GPSHeading = 0;
        try {

            double currentLatBody = xyPosDataJson.getDouble("lat"); //Rådata fra
GPS
            double currentLonBody = xyPosDataJson.getDouble("lon"); //Rådata fra
GPS

            System.out.println(currentLatBody);
            System.out.println(refLatBody);
            System.out.println(currentLonBody);
            System.out.println(refLonBody);

            double headingRadians = Math.atan2(currentLatBody - refLatBody,
currentLonBody - refLonBody);
            //double headingRadians = Math.atan2(refLatBody-currentLatBody,
refLonBody-currentLonBody);
            GPSHeading = Math.toDegrees(headingRadians);
            GPSHeading = GPSHeading - 90;
            if (GPSHeading < 0) {
                GPSHeading = 360 + GPSHeading;
            }
            double vectorLength = Math.abs((refLatBody - (currentLatBody)) +
(refLonBody - (currentLonBody)));
            //Putter til jsonObjekt for senere håndtering

            refLatBody = xyPosDataJson.getDouble("lat"); //Rådata fra GPS
            refLonBody = xyPosDataJson.getDouble("lon"); //Rådata fra GPS

        } catch (JSONException ex) {
            Logger.getLogger(AutoPilot.class.getName()).log(Level.SEVERE, null,
ex);
        }
        return GPSHeading;
    }

    protected double bearing() {
        try {
            double currentLatBody = xyPosDataJson.getDouble("lat"); //Rådata fra
GPS
            double currentLonBody = xyPosDataJson.getDouble("lon"); //Rådata fra
GPS

```

```

        AutoPilot
        double latitude1 = Math.toRadians(refLatBody);
        double latitude2 = Math.toRadians(currentLatBody);
        double longDiff = Math.toRadians(currentLonBody - refLonBody);
        double y = Math.sin(longDiff) * Math.cos(latitude2);
        double x = Math.cos(latitude1) * Math.sin(latitude2) -
Math.sin(latitude1) * Math.cos(latitude2) * Math.cos(longDiff);
        System.out.println("Y: " + y); System.out.println("X: " + x);
        refLatBody = xyPosDataJson.getDouble("lat"); //Rådata fra GPS
        refLonBody = xyPosDataJson.getDouble("lon"); //Rådata fra GPS

        return (Math.toDegrees(Math.atan2(y, x)) + 360) % 360;
    } catch (JSONException ex) {
        Logger.getLogger(AutoPilot.class.getName()).log(Level.SEVERE, null,
ex);
        return 1;
    }
}

/**
 * Calculates the current position from the Platform to the destination
 * waypoint
 *
 * @param lat1 Platform position: latitude
 * @param lon1 Platform position: longitude
 * @param lat2 Waypoint from client: latitude
 * @param lon2 Waypoint from client: longitude
 * @return
 */
public static double haversine(double lat1, double lon1, double lat2, double
lon2) {
    double dLat = Math.toRadians(lat2 - lat1);
    double dLon = Math.toRadians(lon2 - lon1);
    lat1 = Math.toRadians(lat1);
    lat2 = Math.toRadians(lat2);

    double a = Math.pow(Math.sin(dLat / 2), 2) + Math.pow(Math.sin(dLon /
2), 2) * Math.cos(lat1) * Math.cos(lat2);
    double c = 2 * Math.asin(Math.sqrt(a));
    distanceToWaypoint = RADIUS * c;
    return distanceToWaypoint;
}

public void close() {
    isRunning = false;
}
}

```


Conductor

```
/*
 * 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 serialgpsserver;

import java.util.Collection;
import java.util.concurrent.Semaphore;
import java.util.concurrent.TimeUnit;
import java.util.logging.Level;
import java.util.logging.Logger;
import org.json.JSONException;
import org.json.JSONObject;
import java.util.ArrayList;
import java.util.Iterator;
import java.util.List;
import no.ntnu.videostream.ImageStorageBox;
import org.json.JSONArray;

/**
 *
 * @author Matias
 */
public class Conductor extends Thread {

    private StorageBoxGPS storageBoxGPS;
    private StorageBoxNorthEastGPS storageBoxNorthEastGPS;
    private StorageBoxSocketSend storageBoxSocketSend;
    private StorageBoxSocketReceive storageBoxSocketReceive;
    private StorageBoxPlatformMode storageBoxPlatformMode;
    private StorageBoxRunningMode storageBoxRunningMode;

    private Semaphore semaphoreStorageBoxGPS;
    private Semaphore semaphoreStorageBoxNorthEastGPS;
    private Semaphore semaphoreStorageBoxSocketSend;
    private Semaphore semaphoreStorageBoxRunningMode;
    private Semaphore semaphoreStorageBoxSocketReceive;
    private Semaphore semaphoreStorageBoxPlatformMode;

    private boolean coordAvailable;
    private boolean northEastAvailable;
    private boolean hasDataFromClient;

    private JSONObject jsonToClient;
    private JSONObject gpsJsonData;
    private JSONObject jsonDataFromClient;
    private List<Double> latCoordList;
    private List<Double> lonCoordList;
    private int lastCoordPos;

    private SerialHandler serialHandler;
```

Conductor

```
private double xyLatWaypoint;
private double xyLonWaypoint;
private double latRef;
private double lonRef;
private double spd;

private NEDTransform gpsProc;
private boolean hasWaypoint;
private int waypointNumber;
private boolean hasArrived;
private JSONArray latList;
private JSONArray lonList;

//Hold tunga beint i munnan
Conductor(StorageBoxGPS storageBoxGPS, StorageBoxNorthEastGPS
storageBoxNorthEastGPS, StorageBoxSocketSend storageBoxSocketSend,
StorageBoxSocketReceive storageBoxSocketReceive,
StorageBoxPlatformMode storageBoxPlatformMode, StorageBoxRunningMode
storageBoxRunningMode, Semaphore semaphoreStorageBoxGPS, Semaphore
semaphoreStorageBoxNorthEastGPS, Semaphore semaphoreStorageBoxSocketSend,
Semaphore semaphoreStorageBoxSocketReceive, Semaphore
semaphoreStorageBoxPlatformMode, Semaphore semaphoreStorageBoxRunningMode) {
    this.storageBoxGPS = storageBoxGPS;
    this.storageBoxNorthEastGPS = storageBoxNorthEastGPS;
    this.storageBoxSocketSend = storageBoxSocketSend;
    this.storageBoxSocketReceive = storageBoxSocketReceive;
    this.storageBoxPlatformMode = storageBoxPlatformMode;
    this.storageBoxRunningMode = storageBoxRunningMode;
    this.semaphoreStorageBoxSocketSend = semaphoreStorageBoxSocketSend;
    this.semaphoreStorageBoxGPS = semaphoreStorageBoxGPS;
    this.semaphoreStorageBoxNorthEastGPS = semaphoreStorageBoxNorthEastGPS;
    this.semaphoreStorageBoxSocketReceive =
semaphoreStorageBoxSocketReceive;
    this.semaphoreStorageBoxPlatformMode = semaphoreStorageBoxPlatformMode;
    this.semaphoreStorageBoxRunningMode = semaphoreStorageBoxRunningMode;

    coordAvailable = false;
    northEastAvailable = false;
    jsonToClient = new JSONObject();
    gpsJsonData = new JSONObject();
    jsonDataFromClient = new JSONObject();

    gpsProc = new NEDTransform();
    latCoordList = new ArrayList<>();
    lonCoordList = new ArrayList<>();
    hasWaypoint = false;
    waypointNumber = 0;
    hasArrived = false;

/*
//UDP STREAM
int numberOfPermits = 1;
```

```

Conductor

boolean fairness = true;

Semaphore semaphore = new Semaphore(numberOfPermits, fairness);
ImageStorageBox imgStorageBox = new ImageStorageBox();

VideostreamUDP imageCapture = new VideostreamUDP(semaphore,
imgStorageBox);
imageCapture.setName("Camera Frame Capture Thread");
imageCapture.start();
*/
}

/**
 * Override run() medtode i Thread klassen
 */
@Override
public void run() {

    boolean finished = false;

    try {
        while (!finished) {

            // Get GPS coordinate from GPS
            semaphoreStorageBoxGPS.tryAcquire(); // Get flag from GPS
storage box
            semaphoreStorageBoxNorthEastGPS.tryAcquire();

            northEastAvailable = storageBoxNorthEastGPS.getAvailable();
            coordAvailable = storageBoxGPS.getAvailable();

            if (coordAvailable && northEastAvailable) {
                //System.out.println("Has new GPS data");
                storageBoxGPS.getGPSCoord(jsonToClient);
                System.out.println("New gps:" + jsonToClient);
                //System.out.println("At get statement: " + jsonToClient);
                gpsJsonData = storageBoxNorthEastGPS.getNorthEastJson();
            }

            // Release flag to GPS storage boxes
            semaphoreStorageBoxNorthEastGPS.release();
            semaphoreStorageBoxGPS.release();

            //Innhenter semaphorer for videre håndtering av data
            semaphoreStorageBoxSocketReceive.tryAcquire();
            //System.out.println("Con rec aqu");

            hasDataFromClient = storageBoxSocketReceive.getAvailable();
            if (hasDataFromClient) {
                // System.out.println("Get data from rec in con");
                jsonDataFromClient =

```

```

Conductor
storageBoxSocketReceive.getWaypointData();
    System.out.println("Has data from client" +
jsonDataFromClient);
    }
    semaphoreStorageBoxSocketReceive.release();
    // System.out.println("Con rec rel");

////////Må lage funksjon som bare putter dersom mode er endret!!!!
    if (jsonDataFromClient.length() > 0 &&
jsonDataFromClient.has("Mode")) { //////////Må lage funksjon som bare putter dersom
mode er endret!!!!
        semaphoreStorageBoxPlatformMode.tryAcquire();
        if (!storageBoxPlatformMode.getAvailable()) {
            int temp = jsonDataFromClient.getInt("Mode");
            storageBoxPlatformMode.putDataInt("Mode", temp);
            // gpsJsonData.put("Mode", );
        }
        semaphoreStorageBoxPlatformMode.release();
    }

    if (jsonDataFromClient.length() > 0 &&
jsonDataFromClient.has("Mode") && jsonDataFromClient.getInt("Mode") == 2 &&
jsonDataFromClient.has("NewWaypointList")) {
        // System.out.println("has waypointlist");
        if (jsonDataFromClient.getBoolean("NewWaypointList")) {
            getWaypointLists();
            //System.out.println("test");
        } // sends information to autopilot storage box
        else if (!jsonDataFromClient.getBoolean("NewWaypointList"))
{
            posInfo();
        }
        if (hasWaypoint &&
storageBoxRunningMode.hasArrivedToWaypoint()) {
            setReferenceCoord();
            posInfo();
        }
    } else if (jsonDataFromClient.length() > 0 &&
jsonDataFromClient.has("Mode") && jsonDataFromClient.getInt("Mode") == 1) {
        semaphoreStorageBoxRunningMode.tryAcquire();
        if (!storageBoxRunningMode.getAvailable()) {
            storageBoxRunningMode.putGPSWp(jsonDataFromClient);
        }
        semaphoreStorageBoxRunningMode.release();
    }

    //Sjekker om jsonToClient har elementer for så å aquire semafor
    til storage box,
    //putte json objektet der etter release semaforen
    // System.out.println("CONDUCTORDATA TO CLIENT");
    if (jsonToClient.length() > 0) {

```

```

        Conductor
        semaphoreStorageBoxSocketSend.tryAcquire();
        if (!storageBoxSocketSend.getAvailable()) {
            storageBoxSocketSend.putGPSCoord(jsonToClient);
            // System.out.println("New data sendt to client");
            // System.out.println("I send to client: " +
jsonToClient);
        }
        semaphoreStorageBoxSocketSend.release();
    }
    //Slipper semaforer
    semaphoreStorageBoxRunningMode.release();
    semaphoreStorageBoxSocketReceive.release();
}
}
// catch (InterruptedException ex) {
//     Logger.getLogger(Conductor.class.getName()).log(Level.SEVERE,
null, ex);
// }
catch (JSONException ex) {
    Logger.getLogger(Conductor.class.getName()).log(Level.SEVERE, null,
ex);
}
}

/**
 * Setter referansekoordinater ved å multiplisere mottatte waypoint
 * koordinater med PI/180
 */
public void setReferenceCoord() {
    if (waypointNumber < lastCoordPos) {
        try {
            latRef = latList.getDouble(waypointNumber);
            lonRef = lonList.getDouble(waypointNumber);
            xyLatWaypoint = (latRef * Math.PI / 180.0);
            xyLonWaypoint = (lonRef * Math.PI / 180.0);
            System.out.println("Moving to waypoint " + (waypointNumber + 1)
+ " Latitude: " + latRef + " Longitude: " + lonRef);
            waypointNumber++;

        } catch (JSONException ex) {
            Logger.getLogger(Conductor.class.getName()).log(Level.SEVERE,
null, ex);
        }
    } else {
        System.out.println("Arrived");

        hasWaypoint = false;
    }
}

public void getWaypointLists() {
    hasWaypoint = true;
}

```

Conductor

```

waypointNumber = 0;
try {
    latList = jsonDataFromClient.getJSONArray("LatitudeList");
    lonList = jsonDataFromClient.getJSONArray("LongitudeList");
    lastCoordPos = latList.length();
    System.out.println("Lengden på array: " + lastCoordPos);
    if (jsonDataFromClient.getBoolean("NewWaypointList")) {
        setReferenceCoord();
        posInfo();
    }
    jsonDataFromClient.put("NewWaypointList", false);
} catch (JSONException ex) {
    Logger.getLogger(Conductor.class.getName()).log(Level.SEVERE, null,
ex);
}

private void posInfo() {
    // sends information to autopilot storage box
    try {
        //System.out.println("aqu posinfo");
        semaphoreStorageBoxRunningMode.acquire();
        if (jsonDataFromClient.length() > 0 &&
!storageBoxRunningMode.getAvailable()) {
            gpsJsonData.put("xyLatWaypoint", xyLatWaypoint);
            gpsJsonData.put("xyLonWaypoint", xyLonWaypoint);
            gpsJsonData.put("waypointLat", latRef);
            gpsJsonData.put("waypointLon", lonRef);

            if (gpsJsonData.length() > 3) {
                gpsJsonData = gpsProc.getFlatEarthCoordinates(gpsJsonData);
                //System.out.println("NorthEast coords and speed: " +
gpsJsonData);
                storageBoxRunningMode.putGPSWp(gpsJsonData);
                System.out.println("Sends info to autopilot");
            }
        }
        semaphoreStorageBoxRunningMode.release();
    } catch (JSONException ex) {
        Logger.getLogger(Conductor.class.getName()).log(Level.SEVERE, null,
ex);
    } catch (InterruptedException ex) {
        Logger.getLogger(Conductor.class.getName()).log(Level.SEVERE, null,
ex);
    }
}

/**
 * Setter feltene som representerer nåværende posisjon til oppdatert versjon
 * fra gpsStoragebox
 */
// public void setCurrentCoords(){

```

Conductor

```
//      try {
//          xyLatBody = jsonToClient.getDouble("lat");
//          xyLonBody = jsonToClient.getDouble("lon");
//      } catch (JSONException ex) {
//          Logger.getLogger(Conductor.class.getName()).log(Level.SEVERE,
null, ex);
//      }
//  }
}
```

GPSTData

```
package serialgpsserver;

import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.Calendar;
import java.util.Date;

public class GPSTData {
    private final int maxAvgBase = 10;
    private static SimpleDateFormat dateFormat;

    static {
        dateFormat = new SimpleDateFormat("HHmmssddMMyy");
    }

    private Date time;

    private void setTime(Date time) {
        this.time = time;
    }

    /**
     * true if the gps has a connection
     */
    private boolean connection = false;

    private String horizontal;

    private String vertical;

    /**
     * Speed in km/h
     */
    private Double speed;

    private Double latitude;
    private Double longitude;
    private String gpgga;
    private String gprmc;
    private String gpgsa;
    private Date gpsDate;

    public Date getGpsDate() {
        return gpsDate;
    }

    public void setGpsDate(Date gpsDate) {
        this.gpsDate = gpsDate;
    }

    public void setGpsDate(String hourPart, String dayPart) {
        try {
```



```

                                GPSData
        String tmpHours = hourPart.split("\\.")[0];
        String timeStamp = tmpHours + dayPart;
        setGpsDate(dateFormat.parse(timeStamp));
    } catch (Exception e) {
        e.printStackTrace();
    }
}

/**
 * Constructor
 */
public GPSData() {
    this.time = new Date();
}

public boolean isConnection() {
    return connection;
}

private void setConnection(boolean connection) {
    this.connection = connection;
}

/**
 * E=East or W=West
 *
 * @return
 */
public String getHorizontal() {
    return horizontal;
}

private void setHorizontal(String horizontal) {
    this.horizontal = horizontal;
}

/**
 * N=North or S=South
 *
 * @return
 */
public String getVertical() {
    return vertical;
}

private void setVertical(String vertical) {
    this.vertical = vertical;
}

public Double getSpeed() {
    return speed;
}

```

GPSTData

```
private void setSpeed(Double speed) {
    this.speed = speed;
}

public Double getLongitude() {
    return longitude;
}

private void setLongitude(Double longitude) {
    this.longitude = longitude;
}

public Double getLatitude() {
    return latitude;
}

private void setLatitude(Double latitude) {
    this.latitude = latitude;
}

public String getGpgga() {
    return gpgga;
}

public void setGpgga(String gpgga) {
    this.gpgga = gpgga;
}

public String getGprmc() {
    return gprmc;
}

public void setGprmc(String gprmc) {
    this.gprmc = gprmc;
    String[] data = gprmc.split(",");
    //if data[2] is A then we have a connection. If it is V then we dont.
    setConnection(data[2].equals("A"));
    //setTime(convertStringToDate(data[1], data[9]));
    //XXX: no using gps time anymore cause it was not correct when there was
no signal
    setTime(new Date());
    if (isConnection()) {
        setGpsDate(data[1], data[9]);
        setLongitude(convertGPSdata(Double.valueOf(data[5])));
        setHorizontal(data[6]);
        setLatitude(convertGPSdata(Double.valueOf(data[3])));
        setVertical(data[4]);
        setSpeed(knotsToKmh(data[7]));
    }
}
```

GPSData

```

public String getGpgsa() {
    return gpgsa;
}

public void setGpgsa(String gpgsa) {
    this.gpgsa = gpgsa;
}

public Date getTime() {
    return time;
}

/**
 * Convert data from DDMM.MMMM system to DD.DDDD system
 *
 * @param dat
 * @return
 */
private static Double convertGPSdata(double dat) {
    double deg = Math.floor(dat / 100);
    double min = dat * 100 % 10000 / 6000;
    double result = (double) Math.round((deg + min) * 1000000) / 1000000;
    return result;
}

/**
 *
 * @param time format hhmmss e.g. 225446 - Time of fix 22:54:46 UTC
 * @param date format ddMMyy e.g. 191194 - UTC Date of fix, 19 November 1994
 * @return
 */
private static Date convertStringToDate(String time, String date) {
    Calendar cal = Calendar.getInstance();
    try {
        return dateFormat.parse(time.substring(0, time.indexOf(".")) +
date);
    } catch (ParseException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }
    return null;
}

private static double knotsToKmh(String knots) {
    if (knots != null && !knots.isEmpty()) {
        return Double.parseDouble(knots) * 1.852;
    } else {
        return 0d;
    }
}

@Override

```

GPSTData

```
public String toString() {
    return "Time: " + getTime() + "; Connected: " + isConnected() + ";
Longitude: " + getLongitude() + "; Latitude: " + getLatitude() + "; Speed: " +
getSpeed() + ";";
}
/**
 * Dummy metode for Gjennomsnittlig longitude, Bruker samme verdien pga
refreshrate til GPS
 * @return avgLongitude
 */
public double getAvgLongitude() throws InterruptedException {
    double sum = 0;
    int total = 0;
    for (total = 0; total < maxAvgBase; total++) {
        sum = sum + getLongitude();
        Thread.sleep(100);
    }
    double avgLongitude = sum / total;
    return avgLongitude;
}
/**
 * Dummy metode for Gjennomsnittlig Latitude, Bruker samme verdien pga
refreshrate til GPS
 * @return avgLatitude
 */
public double getAvgLatitude() {
    double sum = 0;
    int total = 0;
    for (total = 0; total < maxAvgBase; total++) {
        sum = sum + getLatitude();
    }
    double avgLatitude = sum / total;
    // avgLatitude = Math.floor(avgLatitude * 100) / 100;
    return avgLatitude;
}
/**
 * Dummy metode for Gjennomsnittlig fart, Bruker samme verdien pga refreshrate
til GPS
 * @return avgSpeed
 */
public double getAvgSpeed() {
    double sum = 0;
    int total = 0;
    for (total = 0; total < maxAvgBase; total++) {
        sum = sum + getSpeed();
    }
    double avgSpeed = sum / total;
    return avgSpeed;
}
```

GPSData

}

GPSDataEventListener

```
package serialgpsserver;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.Date;

import gnu.io.SerialPortEvent;
import gnu.io.SerialPortEventListener;

public class GPSDataEventListener implements SerialPortEventListener {

    private SimpleDateFormat dateFormat = new SimpleDateFormat("HHmmssddMMyy");
    private InputStream inputStream;
    private GPSData locationData;
    private boolean connection = true;

    public GPSDataEventListener(InputStream inputStream) {
        super();

        this.setInputStream(inputStream);
    }

    @Override
    public void serialEvent(SerialPortEvent event) {
        try {
            if (event.getEventType() == SerialPortEvent.DATA_AVAILABLE) {
                BufferedReader reader = new BufferedReader(new
InputStreamReader(getInputStream()));
                try {
                    boolean GPRMC_next_round = false;
                    GPSData location = new GPSData();
                    //System.out.println("Reader ready = " + reader.ready());
                    if (!reader.ready()) {
                        setConnection(false);
                    }
                    while (reader.ready()) {
                        String currLine = reader.readLine();
                        //System.out.println(locationData.getTime());
                        //System.out.println("Curr line = " + currLine);
                        String[] st = currLine.split(",");
                        String type = st.length != 0 ? st[0] : null;
                        if (type.equals("$GPGGA")) {
                            location.setGpgga(currLine);
                        } else if (type.equals("$GPRMC")) {
                            location.setGprmc(currLine);
                            GPRMC_next_round = true;
                        } else if (type.equals("$GPGSA")) {
                            location.setGpgsa(currLine);
                        }
                    }
                }
            }
        }
    }
}
```

```

        GPSDataEventListener
    }
    if (GPRMC_next_round) {
        break;
    }
}
if (GPRMC_next_round) {
    setLocationData(location);
}
} catch (IOException e) {
    setConnection(false);
    e.printStackTrace();
} finally {
    try {
        reader.close();
    } catch (IOException e) {
        e.printStackTrace();
    }
}
} catch (Throwable e) {
    setConnection(false);
    e.printStackTrace();
}
}

private void setInputStream(InputStream inputStream) {
    this.inputStream = inputStream;
}

public InputStream getInputStream() {
    return inputStream;
}

private void setLocationData(GPSData locationData) {
    this.locationData = locationData;
}

public synchronized GPSData getLocationData() {
    return locationData;
}

/**
 * Convert data from DDMM.MMMM system to DD.DDDD system
 *
 * @param dat
 * @return
 */
public String convertGPSdata(double dat) {
    double deg = Math.floor(dat / 100);
    double min = dat * 100 % 10000 / 6000;
    double result = (double) Math.round((deg + min) * 1000000) / 1000000;
    return Double.toString(result);
}

```

GPSTDataEventListener

```
}

/**
 *
 * @param time format hhmmss e.g. 225446 - Time of fix 22:54:46 UTC
 * @param date format ddMMyy e.g. 191194 - UTC Date of fix, 19 November 1994
 * @return
 */
public Date convertStringToDate(String time, String date) {
    try {
        return dateFormat.parse(time + date);
    } catch (ParseException e) {
    }
    return null;
}

public void setConnection(boolean connection) {
    this.connection = connection;
}

public boolean isConnection() {
    return connection;
}
}
```


GPSReader

```
package serialgpsserver;

import gnu.io.CommPortIdentifier;
import gnu.io.PortInUseException;
import gnu.io.SerialPort;
import gnu.io.SerialPortEvent;
import gnu.io.SerialPortEventListener;
import gnu.io.UnsupportedCommOperationException;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.util.Enumeration;
import java.util.Timer;
import java.util.TooManyListenersException;
import java.util.concurrent.Semaphore;
import java.util.concurrent.TimeUnit;
import java.util.logging.Level;
import java.util.logging.Logger;
import org.json.JSONObject;

/**
 *
 * @Edits of several functions: Matias
 */
public class GPSReader extends Thread implements SerialPortEventListener {

    //"DEVICE","path":"/dev/ttyACM0"    GPS tilkobling fra terminal
    private GPSTData currentPosition;

    private long portScanTime = 2000;
    private long updateTime = 500;
    private boolean found = false;
    private InputStream inputStream;
    private GPSTDataEventListener dataListener;
    private SerialPort gpsPort = null;
    private boolean running = true;
//Kode lagt til:
    private StorageBoxGPS storageBoxGPS;
    private StorageBoxNorthEastGPS storageBoxNorthEast;
    private Semaphore semaphoreGPS;
    private Semaphore semaphoreNorthEast;
    private boolean available;
    private double xyLatBody;
    private double xyLonBody;
    private double latRef;
    private double lonRef;
    private boolean northEastAvailable;

    private long gpsUpdateTime = 600;
```

```

        GPSReader
    public GPSReader(StorageBoxGPS storageBoxGPS, StorageBoxNorthEastGPS
storageBoxNorthEastGPS, Semaphore semaphoreGPS, Semaphore semaphoreNorthEast) {
        this.storageBoxGPS = storageBoxGPS;
        this.storageBoxNorthEast = storageBoxNorthEastGPS;
        this.semaphoreGPS = semaphoreGPS;
        this.semaphoreNorthEast = semaphoreNorthEast;
        this.available = true;
        this.northEastAvailable = true;

    }

    private static final String regExp =
"((\\$GPGGA)|(\\$GPRMC)|(\\$GPGSA)|(\\$GPGLL)|(\\$GPGSV)|(\\$GPVTG)).*";

    //Må gå gjennom metoden under: findPort()
    /**
     *
     * @return
     */
    public boolean findPort() {
        // For linux(ubuntu)
        System.setProperty("gnu.io.rxtx.SerialPorts", "/dev/gpsDongle");
        System.out.println("GPS starts");
        Enumeration<CommPortIdentifier> enumer =
CommPortIdentifier.getPortIdentifiers();

        while (enumer.hasMoreElements()) {
            CommPortIdentifier port = enumer.nextElement();

            if (port.getPortType() == CommPortIdentifier.PORT_SERIAL &&
!port.isCurrentlyOwned()) {
                SerialPort serialPort = null;
                try {
                    serialPort = (SerialPort) port.open("MY_PORT_NAME", 2000);
                    System.out.println("GPS at " + serialPort );
                    inputStream = null;
                    inputStream = serialPort.getInputStream();
                    if (inputStream == null) {
                        System.out.println("no input stream");
                        return false;
                    }
                }
                serialPort.addEventListener(this);

                serialPort.notifyOnDataAvailable(true);

                serialPort.setSerialPortParams(4800, SerialPort.DATABITS_8,
                    SerialPort.STOPBITS_1, SerialPort.PARITY_NONE);

                Thread.sleep(getPortScanTime());
            } catch (PortInUseException e) {

```

```

        GPSReader
        e.printStackTrace();
    } catch (IOException e) {
        e.printStackTrace();
    } catch (TooManyListenersException e) {
        e.printStackTrace();
    } catch (UnsupportedOperationException e) {
        e.printStackTrace();
    } catch (InterruptedException e) {
        e.printStackTrace();
    }
}

if (this.isFound()) {
    System.out.println("Found Stream on com port " +
serialPort.getName());
    System.out.println("Waiting for client connection...");
    serialPort.removeEventListener();
    setDataListener(new GPSDataEventListener(inputStream));
    this.gpsPort = serialPort;
    try {
        serialPort.addEventListener(getDataListener());
        serialPort.notifyOnDataAvailable(true);
    } catch (TooManyListenersException e) {
        e.printStackTrace();
    }
    return true;
} else {
    try {
        if (inputStream != null) {
            inputStream.close();
        }
        if (serialPort != null) {
            serialPort.close();
        }
    } catch (IOException e) {
        e.printStackTrace();
    }
}

}

// System.out.println("No Stream found!");
return false;
}

@Override
public void run() {
    try {
        findPort();
        if(getDataListener() != null && getDataListener().isConnection()){
            setRunning(true);
        }
        while (isRunning()) {
            if (isFound() && getDataListener() != null &&

```

```

                                GPSReader
getDataListener().isConnection()) {
    GPSData data = getDataListener().getLocationData();
    setCurrentPosition(data);

//Prøver å aquire semaphore
    semaphoreGPS.acquire();
    semaphoreNorthEast.acquire();
// Tenker å putte GPS koordinater som string i StorageBoxGPS objektets variabel
    available = storageBoxGPS.getAvailable();
    northEastAvailable = storageBoxNorthEast.getAvailable();
    if (!available && !northEastAvailable && data != null &&
data.isConnection()) {
        double lat = data.getLatitude();
        double lon = data.getLongitude();
        double spd = data.getSpeed();
        storageBoxGPS.putGPSCoord(lat, lon, spd);
        xyLatBody = (lat * (Math.PI) / 180.0);
        xyLonBody = (lon * (Math.PI) / 180.0);
        //System.out.println("Latitude: "+ lat + " Longditude: "
+ lon);

        JSONObject xyNorthEast = new JSONObject();
        xyNorthEast.put("xyLatBody", xyLatBody);
        xyNorthEast.put("xyLonBody", xyLonBody);
        xyNorthEast.put("speed", spd);
        xyNorthEast.put("lat", lat);
        xyNorthEast.put("lon", lon);
        storageBoxNorthEast.putGPSCoord(xyNorthEast);

    } else if (data != null && !data.isConnection()) {
        System.out.println("No connection with GPS Satellites");
    }

// Slipper semaphore
    semaphoreGPS.release(); //Bruke finally block for å sikre
release selv om exception?
    semaphoreNorthEast.release();
//Print til terminal
    //
    } else {
        if (getDataListener() != null) {
            setDataListener(null);
            gpsPort.close();
            gpsPort.removeEventListener();
            gpsPort = null;
            setFound(false);
            System.out.println("Not found");
        }

    }
}

```

```

        GPSReader
        Thread.sleep(getUpdateTime());
    }

    } catch (InterruptedException e) {
        e.printStackTrace();
    } catch (Throwable t) {
        t.printStackTrace();
    }
}

@Override
public void serialEvent(SerialPortEvent event
) {

    if (event.getEventType() == SerialPortEvent.DATA_AVAILABLE && !found) {
        BufferedReader reader = new BufferedReader(new
InputStreamReader(getInputStream()));
        String line = null;
        try {
            if (reader.ready()) {
                line = reader.readLine();
                if (line != null && line.matches(regExp)) {
                    this.setFound(true);
                }
            }
        } catch (IOException e) {
            e.printStackTrace();
        } finally {
            try {
                reader.close();
            } catch (IOException e) {
                e.printStackTrace();
            }
        }
    }
}

public long getGPSshortUpdateTime(){
return gpsUpdateTime;
}

public long getUpdateTime() {
    return updateTime;
}

private void setInputStream(InputStream inputStream) {
    this.inputStream = inputStream;
}

public InputStream getInputStream() {
    return inputStream;
}

```

GPSReader

```
private void setFound(boolean found) {
    this.found = found;
}

public boolean isFound() {
    return found;
}

synchronized private void setCurrentPosition(GPSData currentPosition) {
    this.currentPosition = currentPosition;
}

synchronized public GPSData getCurrentPosition() {
    return currentPosition;
}

public void setPortScanTime(long portScanTime) {
    this.portScanTime = portScanTime;
}

public long getPortScanTime() {
    return portScanTime;
}

private void setDataListener(GPSDataEventListener dataListener) {
    this.dataListener = dataListener;
}

private GPSDataEventListener getDataListener() {
    return dataListener;
}

public void setRunning(boolean run) {
    this.running = run;
}

public boolean isRunning() {
    return running;
}

public boolean isConnection() {
    return isFound() && getCurrentPosition() != null &&
getCurrentPosition().isConnection();
}

}
```

LidarHandler

```
/*
 * 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 serialgpsserver;

/**
 *
 * @author pmlei
 */
public class LidarHandler {

    public LidarHandler(SerialHandler lidarArduino){

    }

}
```

ManualMode

```
/*
 * 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 serialgpsserver;

import java.util.concurrent.Semaphore;
import java.util.logging.Level;
import java.util.logging.Logger;
import no.ntnu.videostream.ImageStorageBox;
import no.ntnu.videostream.UDPCameraStream;
import org.json.JSONObject;

/**
 * This class provide manual control of the craft from a tablet
 * @author pmlei
 */
public class ManualMode extends Thread{

    ArduinoHandler ardHandler;
    private boolean isRunning;
    private StorageBoxRunningMode storageBoxRunningMode;
    private Semaphore semaphoreStorageBoxRunningMode;
    private JSONObject manualModeJson;

    /**
     * Manual mode construc
     * @param storageBoxRunningMode
     * @param semaphoreStorageBoxRunningMode

     * @param ardHandler
     */
    public ManualMode(StorageBoxRunningMode storageBoxRunningMode, Semaphore
semaphoreStorageBoxRunningMode, ArduinoHandler ardHandler){

        this.ardHandler = ardHandler;
        this.storageBoxRunningMode = storageBoxRunningMode;
        this.semaphoreStorageBoxRunningMode = semaphoreStorageBoxRunningMode;
    }

    @Override
    public void run() {

        //Boolean for å bryte while
        isRunning = true;

        while (isRunning) {
            try {
                //Tar semaforen for storageBoxAutoPilot
                semaphoreStorageBoxRunningMode.acquire();
                //Sjekker at storageBoxAutoPilot har fått koordinater fra
```


ManualMode

gps/gui

```

        boolean isAvailable = storageBoxRunningMode.getAvailable();

        if (isAvailable) {
            //Lagrer jsonobjektene i egne json objekter

            manualModeJson = new JSONObject();
            manualModeJson = storageBoxRunningMode.getGPSdata();
        }
        semaphoreStorageBoxRunningMode.release();

        if(ardHandler.isRemoteArdConnected()){
ardHandler.getArduino("remoteOperation").send(manualModeJson);
            System.out.println("Data in manual mode: " +manualModeJson);

        }
        else System.out.println("Platform contril arduino not
connected");
        if(ardHandler.isStabilityArdConnected()){
            double dataFromStabilityArduino =
ardHandler.getArduino("StabilityArduino").getArduinoData();
            semaphoreStorageBoxRunningMode.tryAcquire();
            //Sjekker at storageBoxAutoPilot har fått koordinater fra
gps/gui
            isAvailable = storageBoxRunningMode.getAvailable();
            if (isAvailable) {
                storageBoxRunningMode.putData("ArduinoHeading",
dataFromStabilityArduino);
            }
            semaphoreStorageBoxRunningMode.release();
        }
        } catch (InterruptedException ex) {
            Logger.getLogger(ManualMode.class.getName()).log(Level.SEVERE,
null, ex);
        }
    }
}

    public void close() {
        isRunning = false;
    }
}

```

NEDTransform

```
package serialgpsserver;

import static java.nio.file.Files.delete;
import java.util.logging.Level;
import java.util.logging.Logger;
import org.json.JSONException;
import org.json.JSONObject;

/**
 * Transformerer geodetiske koordinater for nå- og referanseverdi til
 * koordinater i NED koordinatsystemet
 *
 * @author Albert
 */
public class NEDTransform {

    //World Geodetic System 1984 konstanter
    //Lengste radius av jordens ellipsoide
    private final double semimajorAxis = 6378137.0;
    //korteste radius av jorden ellipsoide
    private final double semiminorAxis = 6356752.0;
    private final double flattening
        = (semimajorAxis - semiminorAxis) / semimajorAxis;
    private final double f;
    private final double R;

    public NEDTransform() {
        f = flattening;
        R = semimajorAxis;
    }

    /**
     * Flat Earth Coordinates, optimal for small changes in lat/lon used.
     *
     * @param latBody
     * @param lonBody
     * @param latRef
     * @param lonRef
     * @return
     */
    public double[] getFlatEarthCoordinates(double latBody, double lonBody,
        double latRef, double lonRef) {
        double dMy = latBody - latRef;
        double dL = lonBody - lonRef;

        double rN = (R / (Math.sqrt(1 - (2 * f - f * f)
            * Math.pow(Math.sin(latRef), 2))));
        double rM = rN * ((1 - (2 * f - f * f)) / (1 - (2 * f - f * f)
            * Math.pow(Math.sin(latRef), 2)));
        double dN = (dMy / Math.atan(1 / rM));
        double dE = (dL / Math.atan(1 / (rN * Math.cos(latRef))));
        return new double[]{dN, dE};
    }
}
```

NEDTransform

```
// }

public JSONObject getFlatEarthCoordinates(JSONObject jsonFlatEarth) {
    // Tar ut enkelte variabler av JSONObjektet
    try {
        double latBody = jsonFlatEarth.getDouble("xyLatBody");
        double lonBody = jsonFlatEarth.getDouble("xyLonBody");
        double latRef = jsonFlatEarth.getDouble("xyLatWaypoint");
        double lonRef = jsonFlatEarth.getDouble("xyLonWaypoint");
        //*****
        double dMy = latBody - latRef;
        double dL = lonBody - lonRef;

        double rN = (R / (Math.sqrt(1 - (2 * f - f * f)
            * Math.pow(Math.sin(latRef), 2))));
        double rM = rN * ((1 - (2 * f - f * f)) / (1 - (2 * f - f * f)
            * Math.pow(Math.sin(latRef), 2)));
        double dN = (dMy / Math.atan(1 / rM));
        double dE = (dL / Math.atan(1 / (rN * Math.cos(latRef))));

        jsonFlatEarth.remove("latBody");
        jsonFlatEarth.remove("lonBody");
        jsonFlatEarth.remove("latRef");
        jsonFlatEarth.remove("lonRef");

//        jsonFlatEarth.getJSONObject("latBody").remove("latBody");
//        jsonFlatEarth.getJSONObject("lonBody").remove("lonBody");
//        jsonFlatEarth.getJSONObject("latRef").remove("latRef");
//        jsonFlatEarth.getJSONObject("lonRef").remove("lonRef");

        jsonFlatEarth.put("dNorth", dN);
        jsonFlatEarth.put("dEast", dE);
    } catch (JSONException ex) {
        Logger.getLogger(NEDTransform.class.getName()).log(Level.SEVERE,
null, ex);
    }

    return jsonFlatEarth;
}

//Denne gjelder for jord som IKKE er "flat" ikke brukt pga mer kompleks
//flat jord er godt for DP
/**
 * Works for all distances.
 *
 * @param latitudeBody
 * @param longitudeBody
 * @param latitudeReference
 * @param longitudeReference
 * @return
 */
// public double[] getBodyCInNEDByGeodeticBodyPosAndRef(double latitudeBody,
```

```

                                NEDTransform
//      double longitudeBody, double latitudeReference,
//      double longitudeReference) {
//      double NRef = getN(latitudeReference, longitudeReference);
//      double[] xyzRef = llh2ECEF(latitudeReference, longitudeReference,
//      0, NRef);
//      double NBody = getN(latitudeReference, longitudeReference);
//      double[] xyzBody = llh2ECEF(latitudeBody, longitudeBody, 0, NBody);
//      double dx = xyzBody[0] - xyzRef[0];
//      double dy = xyzBody[1] - xyzRef[1];
//      double dz = xyzBody[2] - xyzRef[2];
//
//      double cosPhi = Math.cos(latitudeReference);
//      double sinPhi = Math.sin(latitudeReference);
//      double cosLambda = Math.cos(longitudeReference);
//      double sinLambda = Math.sin(longitudeReference);
//
//      double t = cosLambda * dx + sinLambda * dy;
//
//      double dxEast = -sinLambda * dx + cosLambda * dy;
//      double dzUp = cosPhi * t + sinPhi * dz;
//      double dyNorth = -sinPhi * t + cosPhi * dz;
//
//      double xNorth = dyNorth;
//      double yEast = dxEast;
//      double zDown = -dzUp;
//      return new double[]{xNorth, yEast, zDown};
//  }
//
//  private double getN(double lat, double lon) {
//      double a = semimajorAxis;
//      double b = semiminorAxis;
//      double acos = a * Math.cos(lat);
//      double bsin = b * Math.cos(lon);
//      double aa = a * a;
//      return aa / Math.sqrt(acos * acos + bsin * bsin);
//  }
/**
 * Converts latitude, longitude and height to ECEF coordinate system
 *
 * @param latitude
 * @param longitude
 * @param height
 * @return var-index: x-0, y-1, z-2
 */
//  private double[] llh2ECEF(double lat,
//      double lon, double height, double N) {
//      double x = (N + height) * Math.cos(lat) * Math.cos(lon);
//      double y = (N + height) * Math.cos(lat) * Math.sin(lon);
//      double bOvera = semiminorAxis / semimajorAxis;
//      double z = (N * bOvera * bOvera + height) * Math.sin(lat);
//      double[] xyz = new double[]{x, y, z};
//      return xyz;

```

NEDTransform

```
//  
}
```

PIDController

```
package serialgpsserver;

/**
 * Pid kontroller for Autopilot og Dynamic Positioning
 * @author Albert
 */
public class PIDController {
    //
    //IO Variables

    private double outputVariable;

    //Gains
    private double Kp;
    private double Ki;
    private double Kd;

    //Class variables
    private double integralTerm;
    private double lastError;
    private final double cycleTimeInSeconds;

    private double maxOutput;
    private double minOutput;

    public PIDController() {
        outputVariable = 0.0;
        integralTerm = 0.0;
        lastError = 0.0;

        maxOutput = 2 * 110.0; //Maks output jaging og tverrskipss
        minOutput = 2 * -110.0; //Minimum output jaging og tverrskipss

        Kp = 1.0;
        Ki = 0.0;
        Kd = 0.0;
        cycleTimeInSeconds = 0.2;
    }

    /**
     * Beregner output for regulatoren
     *
     * @param newInput
     * @param referenceVariable
     * @param continuous
     * @return
     */
    public double computeOutput(double newInput, double referenceVariable,
                                boolean continuous) {

        double error = referenceVariable - newInput;
```

```

                                PIDController
// Dersom kontinuerlig kan wrap around
if (continuous) {
    maxOutput = 121.0; //torque(yaw)
    minOutput = -121.0; //torque(yaw)
    if (Math.abs(error) > 180) {
        if (error > 0) {
            error = error - 360.0;
        } else {
            error = error + 360.0;
        }
    }
}
// Integrator anti-windup og integrator ledd
if ((integralTerm + error * cycleTimeInSeconds * Ki) < maxOutput
    && (integralTerm + error * cycleTimeInSeconds * Ki)
    > minOutput) {
    integralTerm += Ki * error * cycleTimeInSeconds;
}
double dError = (error - lastError) / cycleTimeInSeconds;
//Beregn PID Output
outputVariable = Kp * error + integralTerm + Kd * dError;
limitOutputVariable();
lastError = error;
return outputVariable;
}

/**
 * resetter feil
 */
public void resetErrors() {
    integralTerm = 0;
    lastError = 0;
}

/**
 * setter forsterkningskonstant for denne regulatoren
 *
 * @param Kp
 * @param Ki
 * @param Kd
 */
public void setTunings(float Kp, float Ki, float Kd) {
    this.Kp = Kp;
    this.Ki = Ki;
    this.Kd = Kd;
}

/**
 * returnerer forsterkningskonstantene for denne regulatoren
 *
 * @return

```

```

                                PIDController
    */
    public double[] getTunings() {
        return new double[]{Kp, Ki, Kd};
    }

    /**
     * setter forsterkningskonstantene for vdenne regulatoren
     *
     * @param gainChanged
     * @param newControllerGain
     */
    void setGain(int gainChanged, double newControllerGain) {
        switch (gainChanged) {
            case 1:
            case 4:
            case 7:
                Kp = newControllerGain;
                break;
            case 2:
            case 5:
            case 8:
                Ki = newControllerGain;
                break;
            case 3:
            case 6:
            case 9:
                Kd = newControllerGain;
                break;
        }
    }

    /**
     * begrenser output variablene
     */
    private void limitOutputVariable() {
        if (outputVariable > maxOutput) {
            outputVariable = maxOutput;
        } else if (outputVariable < minOutput) {
            outputVariable = minOutput;
        }
    }
}

```


PlatformMode

```
/*
 * 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 serialgpsserver;

import java.util.concurrent.Semaphore;
import java.util.logging.Level;
import java.util.logging.Logger;

/**
 *
 * @author pmlei
 */
public class PlatformMode extends Thread {

    private ArduinoHandler ardHandler;
    private StorageBoxPlatformMode storageBoxPlatformMode;
    private final StorageBoxRunningMode storageBoxRunningMode;
    private Semaphore semaphoreStorageBoxPlatformMode;
    private final Semaphore semaphoreStorageBoxRunningMode;
    private int currentMode = 2;

    private AutoPilot autoPilot;
    private ManualMode manualMode;

    public PlatformMode(ArduinoHandler ardHandler, StorageBoxPlatformMode
storageBoxPlatformMode, StorageBoxRunningMode storageBoxRunningMode, Semaphore
semaphoreStorageBoxPlatformMode, Semaphore semaphoreStorageBoxRunningMode){

        this.ardHandler = ardHandler;
        this.storageBoxPlatformMode = storageBoxPlatformMode;
        this.storageBoxRunningMode = storageBoxRunningMode;
        this.semaphoreStorageBoxPlatformMode = semaphoreStorageBoxPlatformMode;
        this.semaphoreStorageBoxRunningMode = semaphoreStorageBoxRunningMode;

    }

    /**
     *
     */
    @Override
    public void run(){
        System.out.println("Ready to choose mode");
        boolean isRunning = true;
        startMode(currentMode);

        while(isRunning){
            try {
                // System.out.println("aqu platform mode");
                semaphoreStorageBoxPlatformMode.acquire();
            }
        }
    }
}
```

```

        PlatformMode
// System.out.println("Got aqu platform mode");
boolean isAvailable = storageBoxPlatformMode.getAvailable();
if(isAvailable){
    // System.out.println("has mode");
    int mode = storageBoxPlatformMode.getMode();
    if(currentMode != mode){
        startMode(mode);
        currentMode = mode;
    }
}
semaphoreStorageBoxPlatformMode.release();
} catch (InterruptedException ex) {
    Logger.getLogger(PlatformMode.class.getName()).log(Level.SEVERE,
null, ex);
}
}

private void startMode(int mode){
    switch(mode){

        case 1:
            if(autoPilot != null && autoPilot.isAlive()) autoPilot.close();
            manualMode = new ManualMode(storageBoxRunningMode,
semaphoreStorageBoxRunningMode, ardHandler);
            manualMode.setName("Manual mode");
            manualMode.start();
            System.out.println("Starting Manual mode");
            break;

        case 2:
            if(manualMode != null && manualMode.isAlive())
manualMode.close();
            autoPilot = new AutoPilot(storageBoxRunningMode,
semaphoreStorageBoxRunningMode, ardHandler);
            autoPilot.setName("Autopilot thread");
            autoPilot.start();
            System.out.println("Starting Autopilot");
            break;

        default:
            break;
    }
}
}

```

RotationMatrix

```
package serialgpsserver;

import org.apache.commons.math3.linear.ArrayRealVector;
import org.apache.commons.math3.linear.BlockRealMatrix;
import org.apache.commons.math3.linear.RealVector;

/**
 * Klasse for å opprette rotasjonsmatrise mellom BODY og NED
 *
 * @author Albert
 */
public class RotationMatrix {

    private BlockRealMatrix Rz;
    private double[][] raw;
    ArrayRealVector vector;

    public RotationMatrix(double headingDegrees) {
        double headingRadians = headingDegrees * (double) Math.PI / 180.0f;
        raw = new double[][]{
            {c(headingRadians), -s(headingRadians), 0},
            {s(headingRadians), c(headingRadians), 0},
            {0, 0, 1}};
        Rz = new BlockRealMatrix(raw);
    }

    /**
     * transponerer rotasjonsmatrisen og multipliserer den med input vektor
     *
     * @param u
     * @param v
     * @param w
     * @return
     */
    public double[] multiplyRzwithV(double u, double v, double w) {
        vector = new ArrayRealVector(new double[]{u, v, w});
        //Rz'*returnVector 3x3 * 3x1 = 3x1
        RealVector returnVector = Rz.transpose().operate(vector);
        return returnVector.toArray();
    }

    /**
     * cosinus funksjon
     *
     * @param radians
     * @return
     */
    private double c(double radians) {
        return (double) Math.cos(radians);
    }
}
```

RotationMatrix

```
/**
 * sinusfunksjon
 *
 * @param radians
 * @return
 */
private double s(double radians) {
    return (double) Math.sin(radians);
}
}
```

SerialGPSServer

```
/*
 * 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 serialgpsserver;

import java.io.IOException;
import java.util.concurrent.Semaphore;

/**
 *
 * @author Matias
 */
public class SerialGPSServer {
    static {

        System.load("/home/odroid/NetbeansProjects/Plattform/libs/soFiles/librxtxSerial.
so");
        System.load("/home/odroid/NetbeansProjects/Plattform/libs/soFiles/librxtxParalle
l.so");
        System.load("/home/odroid/NetbeansProjects/Plattform/libs/soFiles/libopencv_2413
.so");
    }

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) throws IOException {

        int portNumber;
        portNumber = 7777;

        int numberOfPermits = 1;
        // int numberOfProducers = 1; Fra eksempel, brukes ikke her(enda)
        boolean fairness = true; //Java semaphore
        String comport = "";
        //Nødvendige Semaphorer
        Semaphore semaphoreStorageBoxGPS = new Semaphore(numberOfPermits,
fairness);
        Semaphore semaphoreStorageBoxSocketSend = new Semaphore(numberOfPermits,
fairness);
        Semaphore semaphoreStorageBoxSocketReceive = new
Semaphore(numberOfPermits, fairness);
        Semaphore semaphoreStorageBoxPlatformMode = new
Semaphore(numberOfPermits, fairness);
        Semaphore semaphoreStorageBoxNorthEastGPS = new
Semaphore(numberOfPermits, fairness);
        Semaphore semaphoreStorageBoxRunningMode = new
Semaphore(numberOfPermits, fairness);
        // Semaphore semaphoreArduinoComm = new Semaphore(numberOfPermits,
fairness); // "numberOfPermits" må kanskje endres for denne semaforen.
        // Semaphore semaphoreSocketSend = new Semaphore(numberOfPermits,
```

SerialGPSServer

```
fairness);
//Nødvendige fellesobjekt(storagebox)
    StorageBoxGPS storageBoxGPS = new StorageBoxGPS();
    StorageBoxSocketSend storageBoxSocketSend = new StorageBoxSocketSend();
    StorageBoxSocketReceive storageBoxSocketReceive = new
StorageBoxSocketReceive();
    StorageBoxPlatformMode storageBoxPlatformMode = new
StorageBoxPlatformMode();
    StorageBoxNorthEastGPS storageBoxNorthEastGPS = new
StorageBoxNorthEastGPS();
    StorageBoxRunningMode storageBoxRunningMode = new
StorageBoxRunningMode();
    // StorageBoxArduinoComm storageBoxArduinoComm = new
StorageBoxArduinoComm();
    //Oppretter "TrådObjekter"
    ArduinoHandler arduinoHandler = new ArduinoHandler();
    ServerHandler serverHandler = new ServerHandler(storageBoxSocketSend,
storageBoxSocketReceive, semaphoreStorageBoxSocketSend,
semaphoreStorageBoxSocketReceive, portNumber );
    GPSReader reader = new GPSReader(storageBoxGPS, storageBoxNorthEastGPS,
semaphoreStorageBoxGPS, semaphoreStorageBoxNorthEastGPS);
    PlatformMode platformMode = new PlatformMode(arduinoHandler,
storageBoxPlatformMode, storageBoxRunningMode, semaphoreStorageBoxPlatformMode,
semaphoreStorageBoxRunningMode);
    LidarHandler lidarHandler = new
LidarHandler(arduinoHandler.getArduino("LidarHandler"));
    Conductor conductor = new Conductor(storageBoxGPS,
storageBoxNorthEastGPS, storageBoxSocketSend, storageBoxSocketReceive,
storageBoxPlatformMode, storageBoxRunningMode,
        semaphoreStorageBoxGPS, semaphoreStorageBoxNorthEastGPS,
semaphoreStorageBoxSocketSend, semaphoreStorageBoxSocketReceive,
semaphoreStorageBoxPlatformMode, semaphoreStorageBoxRunningMode);

// ServerSocketWorker serverSocketWorker = new ServerSocketWorker(storageBoxGPS,
semaphoreStorageBoxGPS, portNumber);//test
    //Setter navn og Starter Tråder
    conductor.setName("Conductor Thread");
    conductor.start();

    Thread serverHandlerThread = new Thread(serverHandler);
    serverHandlerThread.setName("Server Handler Thread");
    serverHandlerThread.start();

    reader.setName("GPS-Reader Thread");
    reader.start();//Flytt til ServerHandler i IF(), likt som
ServerSocketWorker
    platformMode.setName("Platform Mode Thread");
    platformMode.start();
}
}
```

SerialHandler

```
/*
 * 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 serialgpsserver;

import gnu.io.CommPortIdentifier;
import gnu.io.SerialPort;
import gnu.io.SerialPortEvent;
import gnu.io.SerialPortEventListener;
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.OutputStream;
import java.util.logging.Level;
import java.util.logging.Logger;
import org.json.JSONException;
import org.json.JSONObject;

/**
 *
 * @author RobinBergset
 */
public class SerialHandler implements SerialPortEventListener {

    //Felt
    //private Enumeration portList;
    //private CommPortIdentifier portId;
    // private HashMap<String, CommPortIdentifier> comList;
    private String portName;
    private SerialPort serialPort;
    private CommPortIdentifier portId;
    private BufferedReader input;
    private OutputStream output;
    private boolean serialReady = false;
    private static final int TIME_OUT = 2000;
    private int DATA_RATE = 9600;
    private boolean isConnected = false;

    private double dataFromArduino;
    private long time;
    private long testTime;

    /**
     * Initialiserer variabler
     */
    public SerialHandler(String serialPort, int baudRate, CommPortIdentifier
portId) {

        this.portName = serialPort;
        this.DATA_RATE = baudRate;
    }
}
```

```

                                SerialHandler
this.portId = portId;

connect(portName);

System.out.println("Arduino up and running");

}

/**
 *
 * @param port
 */
public void connect(String port) {
    try {
        serialPort = (SerialPort) portId.open(this.getClass().getName(),
TIME_OUT);
        serialPort.setSerialPortParams(DATA_RATE,
            SerialPort.DATABITS_8,
            SerialPort.STOPBITS_1,
            SerialPort.PARITY_NONE);
        input = new BufferedReader(new
InputStreamReader(serialPort.getInputStream()));
        output = serialPort.getOutputStream();
        serialReady = true;
        serialPort.addEventListener(this);
        serialPort.notifyOnDataAvailable(true);
        System.out.println("COM-port in use: " + serialPort.getName());
        isConnected = true;

    } catch (Exception e) {
        System.err.println(e.toString());
        System.out.println("Not connected");
        isConnected = false;
    }
}

@Override
public void serialEvent(SerialPortEvent spe) {
    //System.out.println("SerialEvent metode kjøres");
    if (spe.getEventType() == SerialPortEvent.DATA_AVAILABLE) {
        try {
            boolean inputTest = input.ready();
            // System.out.println("Is event ready: " +inputTest);
            if(inputTest){
                String tempDataFromArduino = input.readLine();
                double headingFromArduino =
Double.parseDouble(tempDataFromArduino);
                // System.out.println(headingFromArduino);

                dataFromArduino=headingFromArduino;

                // System.out.println("Raw data: " +tempDataFromArduino);

```



```

        SerialHandler
        //double value = Double.parseDouble(tempDataFromArduino);
        //double calheading = ((value -100));
        // System.out.println("Modulus heading: " + calheading);
    }
} catch (IOException ex) {

Logger.getLogger(SerialHandler.class.getName()).log(Level.SEVERE, null, ex);
    System.out.println("SerialPortEvent readLine error");
}
}

}

public boolean send(JSONObject tempJSON){

    boolean isSent = false;
    time = System.currentTimeMillis();
    if(time > testTime){
        testTime = time + 1000;
    try {
        if (serialReady) {
            String tempString = tempJSON.toString();
            byte[] valuesByte = tempString.getBytes();
            output.write(valuesByte);
            output.flush();
            isSent = true;
        }
    }
    catch (IOException ex) {

Logger.getLogger(SerialHandler.class.getName()).log(Level.SEVERE, null, ex);
        isSent = false;
        System.out.println("isSent = " + isSent);
    }
}
return isSent;
}

public BufferedReader getInputReader() {
    return input;
}

public void close(){
    serialPort.close();
}

    public double getArduinoData(){
        return dataFromArduino;
    }

    public boolean isConnected(){
        return isConnected;
    }
}

```

```
}                                SerialHandler
```

ServerHandler

```
/*
 * 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 serialgpsserver;

import java.io.IOException;
import java.net.*;
import java.util.concurrent.Semaphore;
import java.util.logging.Level;
import java.util.logging.Logger;

/**
 *
 * @author Matias
 */
public class ServerHandler implements Runnable {
    //Div felt. Husk å initialisere alle felter i konstruktør(god kodelstil)

    private StorageBoxSocketSend storageBoxSocketSend;
    private StorageBoxSocketReceive storageBoxSocketReceive;
    private Semaphore semaphoreStorageBoxSocketReceive;
    private Semaphore semaphoreStorageBoxSocketSend;
    private int portNumber;

    public ServerHandler(StorageBoxSocketSend storageBoxSocketSend,
StorageBoxSocketReceive storageBoxSocketReceive, Semaphore
semaphoreStorageBoxSocketSend, Semaphore semaphoreStorageBoxSocketReceive, int
portNumber) {
        this.storageBoxSocketSend = storageBoxSocketSend;
        this.storageBoxSocketReceive = storageBoxSocketReceive;
        this.semaphoreStorageBoxSocketSend = semaphoreStorageBoxSocketSend;
        this.semaphoreStorageBoxSocketReceive =
semaphoreStorageBoxSocketReceive;

        this.portNumber = portNumber;
    }

    @Override
    public void run() {
        try {
            ServerSocket serverSock = new ServerSocket(7777); //Setter opp port
på server

            while (true) {
                Socket socket = serverSock.accept(); //Godtar
tilkopling til server
                System.out.println("Client connection established on port " +
socket.getLocalPort())
            }
        } catch (IOException ex) {
            Logger.getLogger(ServerHandler.class.getName()).log(Level.SEVERE, null, ex);
        }
    }
}
```

```

        ServerHandler
            + " from IP " +
socket.getInetAddress().getHostAddress());
        ServerSocketWorkerSend serverWorkerSend = new
ServerSocketWorkerSend(storageBoxSocketSend, semaphoreStorageBoxSocketSend,
portNumber, socket);
        ServerSocketWorkerReceive serverWorkerReceive = new
ServerSocketWorkerReceive(storageBoxSocketReceive,
semaphoreStorageBoxSocketReceive, portNumber, socket);

        Thread serverWorkerSendThread = new Thread(serverWorkerSend);
serverWorkerSendThread.setName("Server Send Thread");
serverWorkerSendThread.start();

        Thread serverWorkerReceiveThread = new
Thread(serverWorkerReceive);
serverWorkerReceiveThread.setName("Server Receive Thread");
serverWorkerReceiveThread.start();

    }
    } catch (IOException ex) {
        Logger.getLogger(ServerHandler.class.getName()).log(Level.SEVERE,
null, ex);
    }
}
}

```

```
1
2 <RelativeLayout xmlns:android="http://
  schemas.android.com/apk/res/android"
3     xmlns:map="http://schemas.android.com/
  apk/res-auto"
4     xmlns:tools="http://schemas.android.
  com/tools"
5     android:id="@+id/
  relativeLayoutFragment"
6     android:layout_width="match_parent"
7     android:layout_height="fill_parent" >
8
9
10    <LinearLayout
11        android:orientation="horizontal"
12        android:layout_width="wrap_content
13        "
14        android:layout_height="
  wrap_content"
15        android:id="@+id/linearLayout">
16
17    </LinearLayout>
18
19    <fragment
20        android:id="@+id/map"
21        android:name="com.google.android.
  gms.maps.SupportMapFragment"
22        android:layout_width="match_parent
23        "
24        android:layout_height="
  match_parent"
25        tools:context="com.example.bakken.
```

```
24 platformcontroller2.AutopilotActivity"
25
26         android:layout_alignParentTop="
    true"
27         android:layout_alignParentStart="
    true" />
28
29     <ListView
30         android:id="@+id/listViewMarkers"
31         android:layout_width="200dp"
32         android:layout_height="
    match_parent"
33         android:divider="@color/
    colorPrimaryDark"
34         android:dividerHeight="3dp"
35         android:background="@drawable/
    rectangular_main_dark_border"
36         android:layout_alignParentRight="
    true"
37         android:layout_alignParentEnd="
    true"
38         android:layout_below="@+id/
    linearLayout"
39         android:layout_alignParentBottom="
    true" />
40
41     <Button
42         android:text="Manual Mode"
43         android:layout_width="120dp"
44         android:layout_height="70dp"
45         android:gravity="center"
46         android:textSize="20dp"
47         android:background="@drawable/
```

```
47 rectangular_main_dark_border"
48         android:id="@+id/buttonManuelMode"
49         android:layout_marginLeft="14dp"
50         android:layout_marginStart="18dp"
51         android:layout_alignParentTop="
true"
52         android:layout_toEndOf="@+id/
linearLayout" />
53
54
55
56     <TextView
57         android:text=""
58         android:layout_width="150dp"
59         android:layout_height="60dp"
60         android:gravity="center"
61         android:background="@drawable/
white_textview"
62         android:id="@+id/
textViewConnectionStatus"
63         android:layout_alignParentBottom="
true"
64         android:layout_toRightOf="@+id/
linearLayout"
65         android:layout_toEndOf="@+id/
linearLayout"
66         android:layout_marginLeft="53dp"
67         android:layout_marginStart="53dp"
/>
68
69     <TextView
70         android:text=""
71         android:layout_width="150dp"
```

```
72         android:layout_height="60dp"
73         android:gravity="center"
74         android:background="@drawable/
white_textview"
75         android:id="@+id/textViewPosition
"
76         android:layout_marginRight="132dp
"
77         android:layout_marginEnd="132dp"
78         android:layout_alignParentBottom=
"true"
79         android:layout_toLeftOf="@+id/
textViewHeading"
80         android:layout_toStartOf="@+id/
textViewHeading" />
81
82     <TextView
83         android:text=""
84         android:layout_width="150dp"
85         android:layout_height="60dp"
86         android:gravity="center"
87         android:background="@drawable/
white_textview"
88         android:id="@+id/textViewDistance
"
89         android:layout_marginLeft="104dp"
90         android:layout_marginStart="104dp
"
91         android:layout_alignParentBottom=
"true"
92         android:layout_toRightOf="@+id/
textViewConnectionStatus"
93         android:layout_toEndOf="@+id/
```



```
93  textViewConnectionStatus" />
94
95      <TextView
96          android:text=""
97          android:layout_width="150dp"
98          android:layout_height="60dp"
99          android:gravity="center"
100         android:background="@drawable/
white_textview"
101         android:id="@+id/textViewHeading"
102         android:layout_alignParentBottom=
"true"
103         android:layout_toLeftOf="@+id/
listViewMarkers"
104         android:layout_toStartOf="@+id/
listViewMarkers"
105         android:layout_marginRight="65dp"
106         android:layout_marginEnd="65dp"
/>
107
108      <Button
109          android:id="@+id/buttonStart"
110          android:layout_width="120dp"
111          android:layout_height="70dp"
112          android:text="Start"
113          android:textSize="20dp"
114          android:gravity="center"
115          android:background="@drawable/
rectangular_main_dark_border"
116          android:layout_marginLeft="14dp"
117          android:layout_below="@+id/
linearLayout"
118          android:layout_toEndOf="@+id/
```

```
118 textViewConnectionStatus"
119         android:layout_marginStart="50dp"
120     />
121     <Button
122         android:id="@+id/
buttonClearWaypoints"
123         android:layout_width="120dp"
124         android:layout_height="70dp"
125         android:gravity="center"
126         android:textSize="19dp"
127         android:text="Clear Waypoints"
128         android:background="@drawable/
rectangular_red_button"
129         android:layout_marginLeft="14dp"
130         android:layout_marginEnd="45dp"
131         android:layout_alignParentTop="
true"
132         android:layout_toStartOf="@+id/
textViewHeading" />
133
134     <Button
135         android:id="@+id/
buttonDPVisualization"
136         android:layout_width="120dp"
137         android:layout_height="70dp"
138         android:background="@drawable/
rectangular_main_dark_border"
139         android:gravity="center"
140         android:layout_marginRight="26dp"
141         android:text="DP Visualization"
142         android:layout_marginStart="26dp"
143         android:layout_below="@+id/
```

```
143 linearLayout"
144         android:layout_toEndOf="@+id/
    textViewDistance" />
145
146     <Button
147         android:id="@+id/buttonConnect"
148         android:layout_width="120dp"
149         android:layout_height="70dp"
150         android:background="@drawable/
    rectangular_main_dark_border"
151         android:layout_below="@+id/
    linearLayout"
152         android:layout_marginEnd="27dp"
153         android:layout_toStartOf="@+id/
    listViewMarkers"
154         android:text="Connect" />
155
156
157 </RelativeLayout>
158
```

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <RelativeLayout xmlns:android="http://
  schemas.android.com/apk/res/android"
3     xmlns:app="http://schemas.android.com/
  apk/res-auto"
4     xmlns:tools="http://schemas.android.
  com/tools"
5     android:id="@+id/activity_manual"
6     android:layout_width="match_parent"
7     android:background="@color/
  colorPrimaryDark"
8     android:layout_height="match_parent"
9     android:paddingBottom="@dimen/
  activity_vertical_margin"
10    android:paddingLeft="@dimen/
  activity_horizontal_margin"
11    android:paddingRight="@dimen/
  activity_horizontal_margin"
12    android:paddingTop="@dimen/
  activity_vertical_margin"
13    tools:context="com.example.bakken.
  platformcontroller2.ManualActivity">
14
15
16    <Button
17        android:text="Autopilot"
18        android:layout_width="100dp"
19        android:layout_height="70dp"
20        android:background="@drawable/
  button_rectangular"
21        android:id="@+id/buttonAutopilot"
22        android:textColor="@color/
  cast_expanded_controller_text_color"
```

```
23         android:layout_alignParentTop="
    true"
24         android:layout_alignParentStart="
    true"
25         android:layout_marginStart="54dp"
    />
26
27     <ImageButton
28         android:layout_width="70dp"
29         android:layout_height="70dp"
30         app:srcCompat="@drawable/
    ic_action_arrowleft"
31         android:layout_centerVertical="
    true"
32         android:layout_alignParentLeft="
    true"
33         android:background="@drawable/
    circular_button"
34         android:layout_alignParentStart="
    true"
35         android:id="@+id/imageButtonLeft"
    />
36
37     <ImageButton
38         android:layout_width="70dp"
39         android:layout_height="70dp"
40         app:srcCompat="@drawable/
    ic_action_arrowup"
41         android:layout_above="@+id/
    imageButtonLeft"
42         android:background="@drawable/
    circular_button"
43         android:layout_toRightOf="@+id/
```

```
43 ImageButtonLeft"
44         android:layout_toEndOf="@+id/
ImageButtonLeft"
45         android:id="@+id/
ImageButtonForward" />
46
47     <ImageButton
48         android:layout_width="70dp"
49         android:layout_height="70dp"
50         app:srcCompat="@drawable/
ic_action_arrowright"
51         android:layout_alignBottom="@+id/
ImageButtonLeft"
52         android:background="@drawable/
circular_button"
53         android:layout_toRightOf="@+id/
ImageButtonForward"
54         android:layout_toEndOf="@+id/
ImageButtonForward"
55         android:id="@+id/imageButtonRight"
    />
56
57     <ImageButton
58         android:layout_width="70dp"
59         android:layout_height="70dp"
60         app:srcCompat="@drawable/
ic_action_arrowdown"
61         android:layout_below="@+id/
ImageButtonRight"
62         android:layout_toRightOf="@+id/
ImageButtonLeft"
63         android:background="@drawable/
circular_button"
```

```
64         android:layout_toEndOf="@+id/  
imageButtonLeft"  
65         android:id="@+id/  
imageButtonBackward" />  
66  
67     <ImageButton  
68         android:layout_width="70dp"  
69         android:layout_height="70dp"  
70         android:background="@drawable/  
circular_button"  
71         app:srcCompat="@drawable/  
ic_action_rotateright"  
72         android:id="@+id/  
imageButtonRotateRight"  
73         android:layout_above="@+id/  
imageButtonForward"  
74         android:layout_alignParentEnd="  
true"  
75         android:layout_marginBottom="27dp"  
/>  
76  
77     <ImageButton  
78         android:layout_width="70dp"  
79         android:layout_height="70dp"  
80         android:background="@drawable/  
circular_button"  
81         app:srcCompat="@drawable/  
ic_action_rotateleft"  
82         android:id="@+id/  
imageButtonRotateLeft"  
83         android:layout_alignTop="@+id/  
imageButtonRotateRight"  
84         android:layout_alignStart="@+id/
```

```
84 seekBarDirection" />
85
86     <ImageButton
87         android:layout_width="70dp"
88         android:layout_height="70dp"
89         android:background="@drawable/
circular_button"
90         app:srcCompat="@drawable/
ic_action_horizontalright"
91         android:layout_alignTop="@+id/
ImageButtonSidewaysLeft"
92         android:layout_alignParentRight="
true"
93         android:layout_alignParentEnd="
true"
94         android:id="@+id/
ImageButtonSidewaysRight" />
95
96     <TextView
97         android:text=""
98         android:layout_width="150dp"
99         android:layout_height="60dp"
100        android:id="@+id/
textViewConnectionStatus"
101        android:gravity="center"
102        android:background="@drawable/
white_textview"
103        android:textColor="#000000"
104        android:hint="Connection"
105        android:layout_alignParentBottom=
"true"
106        android:layout_alignStart="@+id/
imageViewVideofeed"
```



```
107         android:layout_marginStart="81dp"
108     />
109     <ImageButton
110         android:layout_width="70dp"
111         android:layout_height="70dp"
112         android:background="@drawable/
circular_button"
113         app:srcCompat="@drawable/
ic_action_horizontalleft"
114         android:id="@+id/
ImageButtonSidewaysLeft"
115         android:layout_alignBottom="@+id/
ImageButtonRight"
116         android:layout_alignStart="@+id/
ImageButtonRotateLeft"
117         android:layout_marginBottom="24dp
" />
118
119     <SeekBar
120         android:layout_width="250dp"
121         android:layout_height="30dp"
122         android:background="@drawable/
button_rectangular"
123         android:id="@+id/seekBarDirection
"
124         android:layout_alignBottom="@+id/
buttonAutopilot"
125         android:layout_alignParentEnd="
true" />
126
127     <TextView
128         android:text=""
```

```
129         android:hint="Position"
130         android:layout_width="150dp"
131         android:layout_height="60dp"
132         android:gravity="center"
133         android:id="@+id/textViewPosition
    "
134         android:background="@drawable/
white_textview"
135         android:layout_marginLeft="81dp"
136         android:layout_below="@+id/
textView"
137         android:layout_alignEnd="@+id/
imageViewVideofeed"
138         android:layout_marginEnd="106dp"
/>
139
140     <TextView
141         android:text=""
142         android:hint="Speed"
143         android:layout_width="150dp"
144         android:layout_height="60dp"
145         android:background="@drawable/
white_textview"
146         android:gravity="center"
147         android:id="@+id/textViewSpeed"
148
149         android:layout_above="@+id/
textView3"
150         android:layout_alignStart="@+id/
textViewConnectionStatus" />
151
152     <TextView
153         android:id="@+id/textViewHeading"
```

```
154         android:layout_width="150dp"
155         android:layout_height="60dp"
156         android:gravity="center"
157         android:background="@drawable/
white_textview"
158         android:text="TextView"
159         android:layout_alignParentBottom=
"true"
160         android:layout_alignLeft="@+id/
textViewPosition"
161         android:layout_alignStart="@+id/
textViewPosition" />
162
163     <TextView
164         android:text="Speed"
165         android:layout_width="
wrap_content"
166         android:layout_height="
wrap_content"
167         android:layout_marginLeft="34dp"
168         android:textSize="20dp"
169         android:id="@+id/textView"
170         android:textColor="@color/
cast_expanded_controller_text_color"
171         android:layout_alignBaseline="@+
id/textView2"
172         android:layout_alignBottom="@+id/
textView2"
173         android:layout_alignStart="@+id/
textView3"
174         android:layout_marginStart="11dp"
175     />
```

```
176         <TextView
177             android:text="Position"
178             android:textSize="20dp"
179             android:layout_width="
wrap_content"
180             android:layout_height="
wrap_content"
181             android:id="@+id/textView2"
182             android:textColor="@color/
cast_expanded_controller_text_color"
183             android:layout_marginStart="26dp"
184             android:layout_above="@+id/
textViewSpeed"
185             android:layout_alignStart="@+id/
textViewPosition" />
186
187         <TextView
188             android:id="@+id/textView3"
189             android:textSize="20dp"
190             android:layout_width="
wrap_content"
191             android:layout_height="
wrap_content"
192             android:textColor="@color/
cast_expanded_controller_text_color"
193             android:text="Connection"
194             android:layout_above="@+id/
textViewConnectionStatus"
195             android:layout_alignLeft="@+id/
textViewConnectionStatus"
196             android:layout_alignStart="@+id/
textViewConnectionStatus"
197             android:layout_marginLeft="22dp"
```

```
198         android:layout_marginStart="22dp"
199     />
200     <TextView
201         android:id="@+id/textView4"
202         android:textSize="20dp"
203         android:layout_width="
wrap_content"
204         android:layout_height="
wrap_content"
205         android:textColor="@color/
cast_expanded_controller_text_color"
206         android:text="Heading"
207         android:layout_marginLeft="15dp"
208         android:layout_below="@+id/
textViewPosition"
209         android:layout_alignEnd="@+id/
textViewHeading"
210         android:layout_marginEnd="30dp"
211     />
212     <ImageView
213         android:layout_width="600dp"
214         android:layout_height="500dp"
215         android:id="@+id/
imageViewVideofeed"
216         android:src="@drawable/
ic_action_name"
217         android:layout_centerHorizontal="
true"
218         android:layout_alignBottom="@+id/
imageButtonBackward" />
219
```

```
220
221     <ImageButton
222         android:id="@+id/
imageButtonLantern"
223         android:layout_width="80dp"
224         android:layout_height="80dp"
225         android:background="@drawable/
button_rectangular"
226         app:srcCompat="@drawable/
ic_action_light"
227         android:layout_marginStart="13dp"
228         android:layout_alignTop="@+id/
imageButtonAlarm"
229         android:layout_alignStart="@+id/
imageButtonSidewaysLeft" />
230
231
232     <ImageButton
233         android:id="@+id/imageButtonAlarm
"
234         android:layout_width="80dp"
235         android:layout_height="80dp"
236         android:background="@drawable/
button_rectangular"
237         app:srcCompat="@drawable/
ic_action_alarm"
238         android:layout_alignBottom="@+id/
textViewPosition"
239         android:layout_alignParentEnd="
true"
240         android:layout_marginBottom="42dp
" />
241
```

242

243 </RelativeLayout>

244

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <RelativeLayout xmlns:android="http://
  schemas.android.com/apk/res/android"
3     xmlns:app="http://schemas.android.com/
  apk/res-auto"
4     xmlns:tools="http://schemas.android.
  com/tools"
5     android:layout_width="match_parent"
6     android:layout_height="match_parent"
7     android:background="@color/
  colorPrimaryDark"
8     tools:context="com.example.bakken.
  platformcontroller2.StartupActivity">
9
10    <Button
11        android:id="@+id/
  buttonStartManualMode"
12        android:layout_width="300dp"
13        android:layout_height="100dp"
14        android:text="Manual Mode"
15        android:background="@drawable/
  button_rectangular"
16        android:layout_marginBottom="69dp"
17        android:layout_above="@+id/
  imageView2"
18        android:layout_alignStart="@+id/
  buttoStartAutopilot" />
19
20    <Button
21        android:id="@+id/
  buttoStartAutopilot"
22        android:layout_width="300dp"
23        android:layout_height="100dp"
```



```
24         android:background="@drawable/
button_rectangular"
25         android:text="Autopilot"
26         android:layout_marginBottom="43dp"
27         android:layout_above="@+id/
buttonStartManualMode"
28         android:layout_centerHorizontal="
true" />
29
30     <ImageView
31         android:id="@+id/imageView2"
32         android:layout_width="551dp"
33         android:layout_height="138dp"
34         android:src="@drawable/ntnulogo"
35         android:layout_marginBottom="46dp"
36         android:layout_alignParentBottom="
true"
37         android:layout_centerHorizontal="
true" />
38
39     <TextView
40         android:id="@+id/textView5"
41         android:textColor="@color/
cast_expanded_controller_text_color"
42         android:layout_width="wrap_content
"
43         android:layout_height="
wrap_content"
44         android:text="Platform Controller"
45         android:textSize="60dp"
46         android:gravity="center"
47         android:textStyle="bold"
48         android:layout_marginTop="24dp"
```

```
49         android:layout_marginRight="366dp"
50         android:layout_alignParentTop="
    true"
51         android:layout_centerHorizontal="
    true" />
52
53
54
55
56
57
58 </RelativeLayout>
59
```

```
1 package com.example.bakken.  
  platformcontroller2;  
2  
3 import android.content.Intent;  
4 import android.graphics.Color;  
5 import android.support.v4.app.  
  FragmentActivity;  
6 import android.os.Bundle;  
7 import android.view.View;  
8 import android.widget.AdapterView;  
9 import android.widget.Button;  
10 import android.widget.ListView;  
11 import android.widget.TextView;  
12 import android.widget.Toast;  
13  
14 import com.example.bakken.  
  platformcontroller2.android.widget.  
  DataStorage;  
15 import com.example.bakken.  
  platformcontroller2.android.widget.  
  SocketHandler;  
16 import com.example.bakken.  
  platformcontroller2.android.widget.  
  StorageBoxReceived;  
17 import com.google.android.gms.maps.  
  CameraUpdateFactory;  
18 import com.google.android.gms.maps.  
  GoogleMap;  
19 import com.google.android.gms.maps.  
  OnMapReadyCallback;  
20 import com.google.android.gms.maps.  
  SupportMapFragment;  
21 import com.google.android.gms.maps.model.
```

```
21 BitmapDescriptorFactory;
22 import com.google.android.gms.maps.model.
    LatLng;
23 import com.google.android.gms.maps.model.
    MarkerOptions;
24 import com.google.android.gms.maps.model.
    Polyline;
25 import com.google.android.gms.maps.model.
    PolylineOptions;
26
27 import org.json.JSONArray;
28 import org.json.JSONException;
29 import org.json.JSONObject;
30
31 import java.io.BufferedReader;
32 import java.io.IOException;
33 import java.io.InputStreamReader;
34 import java.io.PrintWriter;
35 import java.net.Socket;
36 import java.util.ArrayList;
37 import java.util.Timer;
38 import java.util.TimerTask;
39
40 public class AutopilotActivity extends
    FragmentActivity implements
    OnMapReadyCallback {
41
42     private GoogleMap mMap;
43     private ListView markerList;
44     Button manualModeButton;
45     TextView connectionStatusTextView;
46     TextView headingTextView;
47     private Socket socket;
```

```
48     PolylineOptions line;
49     private JSONArray listOfLatitudes =
new JSONArray();
50     private JSONArray listOfLongitudes =
new JSONArray();
51     TextView positionTextView;
52     JSONObject jsonObject1;
53     JSONObject jsonObjectSend;
54     JSONObject jsonObjectRecieved2;
55     DataStorage ds = new DataStorage();
56     StorageBoxReceived sbr = new
StorageBoxReceived();
57     Button startButton;
58     Button clearMarkersButton;
59     Button connectButton;
60     double latitude;
61     double longitude;
62     LatLng position;
63     LatLng operationArea;
64     private boolean isAutoModeRunning;
65     private boolean isRecieveRunning;
66
67
68
69     @Override
70     protected void onCreate(Bundle
savedInstanceState) {
71         super.onCreate(savedInstanceState)
;
72         setContentView(R.layout.
activity_autopilot);
73
74         /** Obtain the SupportMapFragment
```

```

74  and get notified when the map is ready to
    be used.*/
75      SupportMapFragment mapFragment =
        (SupportMapFragment)
        getSupportFragmentManager()
76          .findFragmentById(R.id.
            map);
77      mapFragment.getMapAsync(this);
78
79      /**Initialize and connect xml to
        javacode*/
80      positionTextView = (TextView)
        findViewById(R.id.textViewPosition);
81      headingTextView = (TextView)
        findViewById(R.id.textViewHeading);
82      markerList = (ListView)
        findViewById(R.id.listViewMarkers);
83      startButton = (Button)
        findViewById(R.id.buttonStart);
84      clearMarkersButton = (Button)
        findViewById(R.id.buttonClearWaypoints);
85      manualModeButton = (Button)
        findViewById(R.id.buttonManuelMode);
86      positionTextView = (TextView)
        findViewById(R.id.textViewPosition);
87      connectionStatusTextView = (
        TextView) findViewById(R.id.
        textViewConnectionStatus);
88      connectButton = (Button)
        findViewById(R.id.buttonConnect);
89
90
91      /** By pressing the Manual mode

```

```

91 button, activity is changed to ManualMode
    */
92         manualModeButton.
            setOnClickListener(new View.
                OnClickListener() {
93                 @Override
94                 public void onClick(View v) {
95                     goToManualMode();
96                 }
97             });
98
99
100
101
102     }
103
104
105     /**
106      * Manipulates the map once available
107      *
108      * This callback is triggered when
109      * the map is ready to be used.
110      * This is where we can add markers
111      * or lines, add listeners or move the
112      * camera. In this case,
113      * we just add a marker near Sydney,
114      * Australia.
115      * If Google Play services is not
116      * installed on the device, the user will be
117      * prompted to install
118      * it inside the SupportMapFragment.
119      * This method will only be triggered once
120      * the user has

```

```

112      * installed Google Play services and
      returned to the app.
113      */
114      @Override
115      public void onMapReady(final
      GoogleMap googleMap) {
116
117          jsonObjectRecieved2= new
      JSONObject();
118          jsonObject1 = new JSONObject();
119          mMap = googleMap;
120
121
122          /**A Timer task to update
      position on the Google Map and info in
      the textViews*/
123          TimerTask tasknew = new TimerTask
      () {
124              @Override
125              public void run() {
126                  jsonObjectRecieved2 = sbr
      .getMyRecivedData();
127
128                  AutopilotActivity.this.
      runOnUiThread(new Runnable() {
129                      @Override
130                      public void run() {
131
132                          if (
      jsonObjectRecieved2 != null&&
      jsonObjectRecieved2.length()>0) {
133                              try {
134                                  if(

```



```
134 jsonObjectRecieved2.has("lat")) {
135                                     //
    posMarker.remove();
136
    latitude = jsonObjectRecieved2.getDouble(
        "lat");
137
    longitude = jsonObjectRecieved2.getDouble(
        "lon");
138
    position = new LatLng(latitude, longitude
    );
139
    positionTextView.setText("" + position);
140                                     //
    posMarker = mMap.addMarker(new
    MarkerOptions().position(position).icon(
    BitmapDescriptorFactory.defaultMarker(
    BitmapDescriptorFactory.HUE_GREEN)));
141                                     mMap.
    addMarker(new MarkerOptions().position(
    position).icon(BitmapDescriptorFactory.
    defaultMarker(BitmapDescriptorFactory.
    HUE_GREEN)));
142                                     }else{
143
144                                     }
145                                     if(
    jsonObjectRecieved2.has("heading")) {
146
    headingTextView.setText("" +
    jsonObjectRecieved2.get("heading"));
147                                     }else
```

```

147 {
148     headingTextView.setText("No Heading Data"
149                             );
150
151     } catch (
152         JSONException e) {
153         e.
154         printStackTrace();
155     }
156     } else {
157     positionTextView.setText("No GPS data");
158     }
159     });
160 }
161
162     /**Creates a new Timer which
163     executes the task repeatedly*/
164     Timer timer = new Timer();
165     timer.scheduleAtFixedRate(tasknew
166     , 1000, 1000);
167
168     /**An Array used to display
169     marker location data in a List View on
170     the app*/
171     final ArrayList<LatLng> locations
172     = new ArrayList();
173
174     /**An Array of polylines to be

```

```
169  used on the GUI*/
170      final ArrayList<Polyline>
polylines = new ArrayList<>();
171
172      /**creates an adapter to make
display of data possible*/
173      final ArrayAdapter<LatLng>
adapter = new ArrayAdapter<LatLng>(
getApplicationContext(),
174          android.R.layout.
simple_list_item_1, android.R.id.text1,
locations);
175
176      /** sets the adapter on the GUI,
now the list is visible */
177      markerList.setAdapter(adapter);
178
179      /**A position of the wanted area
on the screen. This position is NTNU
Ålesund*/
180      operationArea = new LatLng(62.
47207,6.235422);
181
182
183      /**Sets the camera view to center
the operation area. This */
184      mMap.moveCamera(
CameraUpdateFactory.newLatLngZoom(
operationArea, 10f));
185
186
187      /***/
188      TimerTask task2 = new TimerTask()
```

```
188 {
189     @Override
190     public void run() {
191         jsonObjectRecieved2 = sbr
            .getMyRecivedData();
192         AutopilotActivity.this.
            runOnUiThread(new Runnable() {
193             @Override
194             public void run() {
195                 if (
                    jsonObjectRecieved2 != null&&
                    jsonObjectRecieved2.length()>0) {
196                     try {
197                         if(
                            jsonObjectRecieved2.has("lat")) {
198                             latitude = jsonObjectRecieved2.getDouble(
                                "lat");
199                             longitude = jsonObjectRecieved2.getDouble
                                ("lon");
200                             operationArea = new LatLng(latitude,
                                longitude);
201                             mMap.
                                moveCamera(CameraUpdateFactory.
                                    newLatLngZoom(operationArea, 17.5f));
202                             }
203
204
205                             } catch (
                                JSONException e) {
206                                 e.
```

```

206 printStackTrace();
207
208
209
210
211
212
213
214     Timer timer2 = new Timer();
215
216     timer2.schedule(task2, 2000);
217
218     /** makes it possible to add
    markers on the map by long-clicking it.
    Also adds polyline.*/
219     mMap.setOnMapLongClickListener(
    new GoogleMap.OnMapLongClickListener() {
220         @Override
221         public void onMapLongClick(
    LatLng latlng) {
222
223
224             /**Adds marker to the map
    */
225             mMap.addMarker(new
    MarkerOptions().position(latlng));
226             locations.add(latlng);
227
228             /**Converts the LatLng
    object to Double, and stores the separate
    data in a HashMap.
229             *We do this to be able
    to send data to the server later.*/

```

```
230                Double lat = latlng.  
                latitude;  
231                Double lng = latlng.  
                longitude;  
232  
233                listOfLatitudes.put(lat);  
234                listOfLongitudes.put(lng)  
                ;  
235  
236  
237                try {  
238  
239                    jsonObject1.put("LatitudeList", listOfLatitudes);  
240                    jsonObject1.put("LongitudeList", listOfLongitudes);  
241                    jsonObject1.put("NewWaypointList", true);  
242  
243                    } catch (JSONException e)  
                {  
244                        e.printStackTrace();  
245                    }  
246  
247  
248                ds.setMyWaypoints(  
                jsonObject1);  
249  
250                line = new  
                PolylineOptions();  
251                /**Sets polyline between  
                the markers*/  
252                for (LatLng location :
```

```
252 locations) {
253
254         polylines.add(mMap.
addPolyline(line.add(location).width(5).
color(Color.RED)));
255
256     }
257
258 }
259 });
260
261     clearMarkersButton.
setOnClickListener(new View.
OnClickListener() {
262         @Override
263         public void onClick(View v) {
264             try {
265                 locations.clear();
266
267                 for (int i = 0; i <
listOfLatitudes.length(); i++) {
268                     listOfLatitudes.
remove(i);
269                     listOfLongitudes.
remove(i);
270                 }
271                 jsonObject1.remove("
LatitudeList");
272                 jsonObject1.remove("
LongitudeList");
273                 jsonObject1.put("
NewWaypointList", false);
274                 ds.setMyWaypoints(
```

```

274 jsonObject1);
275             polylines.clear();
276             line = new
PolylineOptions();
277             adapter.clear();
278             markerList.setAdapter
(adapter);
279             mMap.clear();
280             startButton.
setBackgroundDrawable(getResources().
getDrawable(R.drawable.
rectangular_main_dark_border));
281
282
283             Toast.makeText(
AutopilotActivity.this, "Waypoints
cleared!", Toast.LENGTH_LONG).show();
284             } catch (JSONException e)
{
285                 e.printStackTrace();
286             }
287         }
288     });
289
290
291
292         /**We create a new Thread to
handle the socket communication. Android
Studio does not approve of
293         network communication in the
Main Thread*/
294
295         new Thread() {

```



```
296         @Override
297         public void run() {
298
299             this.setName("TCP Send
300 Thread");
301
302             try {
303                 jsonObjectSend = new
304                 JSONObject();
305
306                 /**Tries to find a
307 connection to the remote server on a
308 specified port and address*/
309                 socket =
310                 SocketHandler.getSocket();
311
312                 /**Prints formatted
313 representations of objects to a text-
314 output stream.
315
316                 * This class
317 implements all of the print methods found
318 in PrintStream.*
319
320                 final PrintWriter
321                 printWriter = new PrintWriter(socket.
322                 getOutputStream(), true);
323
324                 jsonObjectSend.put("
325 Mode", 2);
326
327                 printWriter.println(
328                 jsonObjectSend);
329
330                 isAutoModeRunning =
```

```
315 false;
316             while (!
    isAutoModeRunning) {
317
318
319
320             startButton.
    setOnClickListener(new View.
    OnClickListener() {
321
322                 @Override
323                 public void
    onClick(View v) {
324
325                     try {
326
    startButton.setBackgroundDrawable(
    getResources().getDrawable(R.drawable.
    rectangular_main_dark_greenboarder));
327
    jsonObjectSend = ds.getMyWaypoints();
328
    jsonObjectSend.put("Mode", 2);
329
    printWriter.println(jsonObjectSend);
330
331                 } catch (
    JSONException e) {
332                     e.
    printStackTrace();
333                 }
334
335
```

```

336
337     AutopilotActivity.this.runOnUiThread(new
        Runnable() {
338                                     @
        Override
339     public void run() {
340         Toast.makeText(AutopilotActivity.this, "
        Sending waypoints, starting vessel...",
        Toast.LENGTH_LONG).show();
341
342                                     }
343                                     });
344     }
345     });
346
347     connectButton.
        setOnClickListener(new View.
        OnClickListener() {
348                                     @Override
349                                     public void
        onClick(View v) {
350                                     socket =
        SocketHandler.getSocket();
351
        connectButton.setBackgroundDrawable(
        getResources().getDrawable(R.drawable.
        rectangular_main_dark_border));
352
        connectionStatusTextView.setText("No
        Connection");

```

```

353     connectionStatusTextView.
        setBackgroundDrawable(getResources().
            getDrawable(R.drawable.white_textview));
354
355         }
356     });
357
358     }
359
360     } catch (IOException e)
361     {
362         e.printStackTrace();
363     } catch (JSONException e)
364     {
365         e.printStackTrace();
366     }
367
368     }.start();
369
370
371
372     new Thread() {
373         @Override
374         public void run() {
375
376             this.setName("TCP Receive
377 Thread");
378
379             /**Reads the input stream
380 and adds the data to a JSONObject*/

```

```
379         try {
380
381             socket =
382             SocketHandler.getSocket();
383             JSONObject
384             jsonObjectReceived = null;
385             InputStreamReader
386             inRead = new InputStreamReader(socket.
387             getInputStream());
388             BufferedReader
389             bufRead = new BufferedReader(inRead);
390
391             isRecieveRunning =
392             false;
393             while (true) {
394
395                 String
396                 jsonDataMsg = "";
397
398                 jsonDataMsg =
399                 bufRead.readLine();
400
401                 if (jsonDataMsg
402                 != null && jsonDataMsg.length() > 0) {
403                     jsonObjectReceived = new JSONObject(
404                     jsonDataMsg);
405                     //
406                 }
407             }
408         }
```

```
401             if (
                jsonObjectReceived != null &&
                jsonObjectReceived.length() > 0) {
402                 sbr.
                setMyRecivedData(jsonObjectReceived);
403
404             }
405
406
407         }
408
409         } catch (IOException e) {
410             e.printStackTrace();
411         } catch (JSONException e)
        {
412             e.printStackTrace();
413         }
414
415     }
416
417     }.start();
418
419 }
420
421 /**
422     * When Manual Mode button is pressed
    , the intent starts and changes the
    current activity to
423     * ManualActivity
424     */
425     public void goToManualMode() {
426
427         Intent intent = new Intent(this,
```

```
427 ManualActivity.class);  
428         isAutoModeRunning = true;  
429         isRecieveRunning = true;  
430         startActivity(intent);  
431         finish();  
432  
433     }  
434  
435  
436 }  
437
```

```
1 package com.example.bakken.  
  platformcontroller2.android.widget;  
2  
3 import org.json.JSONObject;  
4  
5 import java.io.Serializable;  
6 import java.util.HashMap;  
7  
8 /**  
9  * Created by Bakken on 02.03.2017.  
10 */  
11  
12 public class DataStorage{  
13  
14  
15     private JSONObject myWaypoints = new  
    JSONObject();  
16  
17  
18     public JSONObject getMyWaypoints() {  
19         return myWaypoints;  
20     }  
21  
22     public void setMyWaypoints(JSONObject  
    jsonObject) {  
23         this.myWaypoints = jsonObject;  
24     }  
25  
26 }  
27
```



```
1 package com.example.bakken.  
  platformcontroller2;  
2  
3 import android.content.Intent;  
4 import android.graphics.Bitmap;  
5 import android.graphics.BitmapFactory;  
6 import android.support.v7.app.  
  AppCompatActivity;  
7 import android.os.Bundle;  
8 import android.util.Log;  
9 import android.view.MotionEvent;  
10 import android.view.View;  
11 import android.widget.Button;  
12 import android.widget.ImageButton;  
13 import android.widget.ImageView;  
14 import android.widget.SeekBar;  
15 import android.widget.TextView;  
16  
17 import com.example.bakken.  
  platformcontroller2.android.widget.  
  SocketHandler;  
18 import com.example.bakken.  
  platformcontroller2.android.widget.  
  StorageBoxReceived;  
19 import com.google.android.gms.maps.model.  
  LatLng;  
20  
21 import org.json.JSONException;  
22 import org.json.JSONObject;  
23  
24 import java.io.BufferedReader;  
25 import java.io.IOException;  
26 import java.io.InputStreamReader;
```

```
27 import java.io.PrintWriter;
28 import java.net.DatagramPacket;
29 import java.net.DatagramSocket;
30 import java.net.Socket;
31 import java.net.SocketException;
32 import java.util.Timer;
33 import java.util.TimerTask;
34
35 public class ManualActivity extends
    AppCompatActivity {
36
37     Button autopilotButton;
38     ImageButton forwardButton;
39     ImageButton backwardsButton;
40     ImageButton turnRightButton;
41     ImageButton turnLeftButton;
42     ImageButton rotateLeftButton;
43     ImageButton rotateRightButton;
44     ImageButton horizontalLeftButton;
45     ImageButton horizontalRightButton;
46     SeekBar setSpeedSeekBar;
47     TextView speedTextView;
48     TextView headingTextView;
49     TextView connectionTextView;
50     TextView positionTextView;
51     int speedValue;
52     JSONObject jsonObjectValues = new
    JSONObject();
53     JSONObject jsonObjectReceived;
54     private Socket socket;
55     ImageView videoImageView;
56     private DatagramSocket serverSocket;
57     private static final String
```

```

57 debugString = "debug";
58     private boolean isUDPRunning;
59     private boolean isManualModeRunning;
60     private boolean isRecieveThreadRunning
    ;
61     StorageBoxReceived sbr = new
StorageBoxReceived();
62     ImageButton lanternButton;
63     ImageButton alarmButton;
64     Boolean lightsOn = false;
65     PrintWriter outputStream;
66
67
68     @Override
69     protected void onCreate(Bundle
savedInstanceState) {
70         super.onCreate(savedInstanceState)
    ;
71         setContentView(R.layout.
activity_manual);
72
73         /**Initialize and connect xml to
javacode*/
74         autopilotButton = (Button)
findViewById(R.id.buttonAutopilot);
75         forwardButton = (ImageButton)
findViewById(R.id.imageButtonForward);
76         backwardsButton = (ImageButton)
findViewById(R.id.imageButtonBackward);
77         turnRightButton = (ImageButton)
findViewById(R.id.imageButtonRight);
78         turnLeftButton = (ImageButton)
findViewById(R.id.imageButtonLeft);

```

```
79         rotateLeftButton = (ImageButton)
        findViewById(R.id.imageButtonRotateLeft);
80         rotateRightButton = (ImageButton)
        findViewById(R.id.imageButtonRotateRight
        );
81         horizontalLeftButton = (
        ImageButton) findViewById(R.id.
        imageButtonSidewaysLeft);
82         horizontalRightButton = (
        ImageButton) findViewById(R.id.
        imageButtonSidewaysRight);
83         setSpeedSeekBar = (SeekBar)
        findViewById(R.id.seekBarDirection);
84         speedTextView = (TextView)
        findViewById(R.id.textViewSpeed);
85         headingTextView = (TextView)
        findViewById(R.id.textViewHeading);
86         positionTextView = (TextView)
        findViewById(R.id.textViewPosition);
87         connectionTextView = (TextView)
        findViewById(R.id.
        textViewConnectionStatus);
88         videoImageView = (ImageView)
        findViewById(R.id.imageViewVideofeed);
89         alarmButton = (ImageButton)
        findViewById(R.id.imageButtonAlarm);
90         lanternButton = (ImageButton)
        findViewById(R.id.imageButtonLantern);
91
92
93         /**Set start value to
        directionSeekBar*/
94         setSpeedSeekBar.setMax(3200);
```

```
95         setSpeedSeekBar.setProgress(0);
96
97         /**Set start value on the text
98         views*/
99         speedTextView.setText(" Speed: ")
100        ;
101        speedTextView.setTextSize(30);
102        headingTextView.setText("Not
103        moving ");
104        headingTextView.setTextSize(25);
105        connectionTextView.setText("Not
106        Connected");
107        connectionTextView.setTextSize(23
108        );
109        connectionTextView.
110        setBackgroundDrawable(getResources().
111        getDrawable(R.drawable.red_textview));
112
113        /**When autopilot button is
114        pressed, goToAutopilot methods is called
115        */
116        autopilotButton.
117        setOnClickListener(new View.
118        OnClickListener() {
119            @Override
120            public void onClick(View v) {
121                autopilotButton.
122                setBackgroundDrawable(getResources().
123                getDrawable(R.drawable.
124                rectangular_main_dark_greenboarder));
125                goToAutopilot();
126            }
127        });
```

```
114         }
115     });
116
117     TimerTask tasknew = new TimerTask
118     () {
119         @Override
120         public void run() {
121             jsonObjectReceived = sbr.
122             getMyRecivedData();
123
124             ManualActivity.this.
125             runOnUiThread(new Runnable() {
126                 @Override
127                 public void run() {
128                     if (
129                     jsonObjectReceived != null&&
130                     jsonObjectReceived.length()>0) {
131                         try {
132                             if(
133                             jsonObjectReceived.has("lat")) {
134
135                             double latitude = jsonObjectReceived.
136                             getDouble("lat");
137
138                             double longitude = jsonObjectReceived.
139                             getDouble("lon");
140
141                             LatLng currentPos = new LatLng(latitude,
142                             longitude);
143
144                             positionTextView.setText("" + currentPos)
145                             ;
146                         }
147                     }
148                 }
149             });
150         }
151     };
152 }
```

```
133
134                                     }
135                                     if (
    jsonObjectReceived.has("heading")) {
136    headingTextView.setTextSize(23);
137    headingTextView.setText("" +
    jsonObjectReceived.get("heading"));
138                                     }
139
140                                     } catch (
    JSONException e) {
141                                     e.
    printStackTrace();
142                                     }
143                                     } else {
144    headingTextView.setTextSize(23);
145    headingTextView.setText("No heading data"
    );
146                                     }
147                                     }
148                                     });
149    }
150    };
151
152    Timer timer = new Timer();
153
154    timer.scheduleAtFixedRate(tasknew
    , 1000, 1000);
155
```

```
156
157         /**
158         * A separate thread to handle
159         the network communication and data
160         transfer.
161         */
162         new Thread() {
163             @Override
164             public void run() {
165                 this.setName("Control
166 Thread");
167                 isManualModeRunning =
168 false;
169                 try {
170                     socket =
171 SocketHandler.getSocket();
172                     runOnUiThread(new
173 Runnable() {
174                         @Override
175                         public void run()
176                         {
177                             if (socket.
178 isConnected()) {
179                                 connectionTextView.setText("" + "
180 Connected");
181                                 connectionTextView.setTextSize(25);
182                             }
183                         }
184                     }
185                 } catch (IOException e) {
186                     e.printStackTrace();
187                 }
188             }
189         }
190     }
191 }
```



```
177 connectionTextView.setBackgroundDrawable(  
    getResources().getDrawable(R.drawable.  
    green_textview));  
178                                     } else {  
179  
    connectionTextView.setText("" + "Not  
    Connected");  
180  
    connectionTextView.setTextSize(25);  
181  
    connectionTextView.setBackgroundDrawable(  
    getResources().getDrawable(R.drawable.  
    red_textview));  
182                                     }  
183                                     }  
184                                     });  
185  
186  
187                                     outputStream = new  
    PrintWriter(socket.getOutputStream(),  
    true);  
188                                     setSpeedSeekBar.  
    setProgress(0);  
189  
190                                     jsonObjectValues.put(  
    "Mode", 1);  
191                                     jsonObjectValues.put(  
    "lights", 0);  
192                                     jsonObjectValues.put(  
    "horn", 0);  
193                                     outputStream.println(  
    jsonObjectValues);  
194
```

```
195
196             while (!
197         isManualModeRunning) {
198
199             /** A seekBar
200         that sets a double value from 0 -> 3200
201         This value is used to
202             * control the
203         speed of the platform
204             */
205         setSpeedSeekBar.
206         setOnSeekBarChangeListener(new SeekBar.
207         OnSeekBarChangeListener() {
208
209             @Override
210             public void
211         onProgressChanged(SeekBar seekBar, int
212         progress, boolean fromUser) {
213
214         speedValue = 0;
215
216         speedValue = progress;
217
218         speedTextView.setText("" + speedValue);
219
220             }
221
222             @Override
223             public void
224         onStartTrackingTouch(SeekBar seekBar) {
225
226             }
227         }
```

```

216                                     @Override
217                                     public void
    onStopTrackingTouch(SeekBar seekBar) {
218
219
220                                     }
221                                     });
222
223                                     /**
224                                     * A touch button
    which sends data to server. (forward
    command)
225                                     */
226
227
228                                     forwardButton.
    setOnTouchListener(new View.
    OnTouchListener() {
229                                     @Override
230                                     public
    boolean onTouch(View v, MotionEvent event
    ) {
231                                     int
    eventAction = event.getAction();
232
233
234                                     switch (
    eventAction) {
235                                     case
    MotionEvent.ACTION_DOWN:
236
    forwardButton.setBackgroundDrawable(
    getResources().getDrawable(R.drawable.

```

```
236 circular_button_green));
237
    try {
238
239         jsonObjectValues.put("smcSerial1", 0);
240
241         jsonObjectValues.put("smcSerial2", (
speedValue * -1));
242
243         jsonObjectValues.put("smcSerial3", 0);
244
245         jsonObjectValues.put("smcSerial4", (
speedValue * -1));
246
247         outputStream.println(jsonObjectValues)
;
248
249     }
250
251     catch (JSONException e) {
252
253         e.printStackTrace();
254
255     }
256
257     break;
258
259     case
MotionEvent.ACTION_UP:
260
261     try {
262
263         jsonObjectValues.put("smcSerial1", 0);
```

```
254         jsonObjectValues.put("smcSerial2", 0);
255         jsonObjectValues.put("smcSerial3", 0);
256         jsonObjectValues.put("smcSerial4", 0);
257         outputStream.println(jsonObjectValues)
        ;
258         forwardButton.setBackgroundDrawable(
        getResources().getDrawable(R.drawable.
        circular_button));
259
260     }
        catch (JSONException e) {
261
        e.printStackTrace();
262     }
263
        break;
264
265     }
266     return
        false;
267     }
268     });
269
270
271     /**
272     * A touch button
        which sends data to server. (turn right
        command)
```

```
273                                     */
274                                     turnRightButton.
    setOnTouchListener(new View.
    OnTouchListener() {
275                                     @Override
276                                     public
    boolean onTouch(View v, MotionEvent event
    ) {
277                                     int
    eventAction = event.getAction();
278
279    turnRightButton.setBackgroundDrawable(
    getResources().getDrawable(R.drawable.
    circular_button_green));
280
281                                     switch (
    eventAction) {
282                                     case
    MotionEvent.ACTION_DOWN:
283
    turnRightButton.setBackgroundDrawable(
    getResources().getDrawable(R.drawable.
    circular_button_green));
284
    try {
285
286        jsonObjectValues.put("smcSerial1",
    speedValue);
287
    jsonObjectValues.put("smcSerial2", (
    speedValue * -1));
```

```
288         jsonObjectValues.put("smcSerial3", 0);
289         jsonObjectValues.put("smcSerial4", ((
speedValue * -1) * 0.5));
290
291         outputStream.println(jsonObjectValues)
;
292
293     }
    catch (JSONException e) {
294
        e.printStackTrace();
295
    }
296
297     break;
298
299         case
MotionEvent.ACTION_UP:
300
301         turnRightButton.setBackgroundDrawable(
getResources().getDrawable(R.drawable.
circular_button));
302
303     try {
304
        jsonObjectValues.put("smcSerial1", 0);
305
        jsonObjectValues.put("smcSerial2", 0);
```

```

306         jsonObjectValues.put("smcSerial3", 0);
307         jsonObjectValues.put("smcSerial4", 0);
308         outputStream.println(jsonObjectValues)
309         ;
310
311     }
312     catch (JSONException e) {
313         e.printStackTrace();
314     }
315     break;
316     }
317     return
318     false;
319     }
320     });
321     /**
322     * A touch button
323     which sends data to server. (turn left
324     command)
325     */
326     turnLeftButton.
    setOnTouchListener(new View.
    OnTouchListener() {
        @Override
        public

```



```
326 boolean onTouch(View v, MotionEvent event
    ) {
327                                     int
    eventAction = event.getAction();
328
329                                     switch (
    eventAction) {
330                                     case
    MotionEvent.ACTION_DOWN:
331
    turnLeftButton.setBackgroundDrawable(
    getResources().getDrawable(R.drawable.
    circular_button_green));
332
    try {
333
        jsonObjectValues.put("smcSerial1", (
    speedValue * -1));
334
        jsonObjectValues.put("smcSerial2", ((
    speedValue * -1) * 0.5));
335
        jsonObjectValues.put("smcSerial3", 0);
336
        jsonObjectValues.put("smcSerial4", (
    speedValue * -1));
337
        outputStream.println(jsonObjectValues)
    ;
338
339
340                                     }

    catch (JSONException e) {
```

```
341         e.printStackTrace();
342     }
343
344
345     break;
346
347     case
    MotionEvent.ACTION_UP:
348
    turnLeftButton.setBackgroundDrawable(
    getResources().getDrawable(R.drawable.
    circular_button));
349
350
    try {
351
        jsonObjectValues.put("smcSerial1", 0);
352
        jsonObjectValues.put("smcSerial2", 0);
353
        jsonObjectValues.put("smcSerial3", 0);
354
        jsonObjectValues.put("smcSerial4", 0);
355
        outputStream.println(jsonObjectValues)
    ;
356
357     }
    catch (JSONException e) {
358
        e.printStackTrace();
```

```
359                                     }
360
    break;
361
362                                     }
363                                     return
    false;
364                                     }
365                                     });
366
367                                     /**
368                                     * A touch button
    which sends data to server. (backwards
    command)
369                                     */
370                                     backwardsButton.
    setOnTouchListener(new View.
    OnTouchListener() {
371                                     @Override
372                                     public
    boolean onTouch(View v, MotionEvent event
    ) {
373                                     int
    eventAction = event.getAction();
374
375                                     switch (
    eventAction) {
376                                     case
    MotionEvent.ACTION_DOWN:
377
    backwardsButton.setBackgroundDrawable(
    getResources().getDrawable(R.drawable.
    circular_button_green));
```

```
378
    try {
379
        jsonObjectValues.put("smcSerial1", 0);
380
        jsonObjectValues.put("smcSerial2",
speedValue);
381
        jsonObjectValues.put("smcSerial3", 0);
382
        jsonObjectValues.put("smcSerial4",
speedValue);
383
        outputStream.println(jsonObjectValues)
        ;
384
385                                     }
        catch (JSONException e) {
386
            e.printStackTrace();
387
388                                     }
389
390
        break;
391
392                                     case
MotionEvent.ACTION_UP:
393
        backwardsButton.setBackgroundDrawable(
        getResources().getDrawable(R.drawable.
        circular_button));
394
```

```
395
    try {
396
        jsonObjectValues.put("smcSerial1", 0);
397
        jsonObjectValues.put("smcSerial2", 0);
398
        jsonObjectValues.put("smcSerial3", 0);
399
        jsonObjectValues.put("smcSerial4", 0);
400
        outputStream.println(jsonObjectValues)
        ;
401
402
        }
        catch (JSONException e) {
403
            e.printStackTrace();
404
        }
405
        break;
406
407
        }
408
        return
        false;
409
        }
410
        });
411
412
        /**
413
        * A touch button
        which sends data to server. (rotate left
        command)
414
        */
```

```
415 rotateLeftButton.  
    setOnTouchListener(new View.  
        OnTouchListener() {  
416             @Override  
417             public  
                boolean onTouch(View v, MotionEvent event  
        ) {  
418                 int  
                eventAction = event.getAction();  
419  
420                 switch (eventAction) {  
421                     case  
                        MotionEvent.ACTION_DOWN:  
422  
                            rotateLeftButton.setBackgroundDrawable(  
                                getResources().getDrawable(R.drawable.  
                                    circular_button_green));  
423  
                            try {  
424  
                                jsonObjectValues.put("smcSerial1", (  
                                    speedValue * -1));  
425  
                                jsonObjectValues.put("smcSerial2", 0);  
426  
                                jsonObjectValues.put("smcSerial3", (  
                                    speedValue * -1));  
427  
                                jsonObjectValues.put("smcSerial4", 0);  
428  
                                outputStream.println(jsonObjectValues)  
                                ;
```

```
429
430                                     }
    catch (JSONException e) {
431        e.printStackTrace();
432                                     }
433
434
435    break;
436
437                                     case
    MotionEvent.ACTION_UP:
438
439        rotateLeftButton.setBackgroundDrawable(
        getResources().getDrawable(R.drawable.
        circular_button));
440
441    try {
442        jsonObjectValues.put("smcSerial1", 0);
443        jsonObjectValues.put("smcSerial2", 0);
444        jsonObjectValues.put("smcSerial3", 0);
445        jsonObjectValues.put("smcSerial4", 0);
446        outputStream.println(jsonObjectValues)
        ;
447
```

```

448                                                                                                     }
    catch (JSONException e) {
449
        e.printStackTrace();
450                                                                                                     }
451
    break;
452
453                                                                                                     }
454                                                                                                     return
    false;
455                                                                                                     }
456                                                                                                     });
457
458                                                                                                     /**
459                                                                                                     * A touch button
        which sends data to server. (rotate
        right command)
460                                                                                                     */
461                                                                                                     rotateRightButton
        .setOnTouchListener(new View.
        OnTouchListener() {
462                                                                                                     @Override
463                                                                                                     public
        boolean onTouch(View v, MotionEvent event
        ) {
464                                                                                                     int
        eventAction = event.getAction();
465
466                                                                                                     switch (
        eventAction) {
467                                                                                                     case
        MotionEvent.ACTION_DOWN:

```



```
468 rotateRightButton.setBackgroundDrawable(  
    getResources().getDrawable(R.drawable.  
    circular_button_green));  
469  
    try {  
470  
        jsonObjectValues.put("smcSerial1",  
        speedValue);  
471  
        jsonObjectValues.put("smcSerial2", 0);  
472  
        jsonObjectValues.put("smcSerial3",  
        speedValue);  
473  
        jsonObjectValues.put("smcSerial4", 0);  
474  
        outputStream.println(jsonObjectValues)  
        ;  
475  
476                                     }  
        catch (JSONException e) {  
477  
            e.printStackTrace();  
478                                     }  
479  
480  
481  
        break;  
482  
483                                     case  
        MotionEvent.ACTION_UP:  
484
```

```
485 rotateRightButton.setBackgroundDrawable(  
    getResources().getDrawable(R.drawable.  
    circular_button));  
486  
487 try {  
488     jsonObjectValues.put("smcSerial1", 0);  
489     jsonObjectValues.put("smcSerial2", 0);  
490     jsonObjectValues.put("smcSerial3", 0);  
491     jsonObjectValues.put("smcSerial4", 0);  
492     outputStream.println(jsonObjectValues)  
493     ;  
494     }  
    catch (JSONException e) {  
495         e.printStackTrace();  
496     }  
497     break;  
498  
499     }  
500     return  
    false;  
501     }  
502     });  
503
```

```

504                                     /**
505                                     * A touch button
506                                     * which sends data to server. (go
507                                     horizontal left command)
508                                     */
509                                     horizontalLeftButton.setOnTouchListener(
510                                     new View.OnTouchListener() {
511                                     @Override
512                                     public
513                                     boolean onTouch(View v, MotionEvent event
514                                     ) {
515                                     int
516                                     eventAction = event.getAction();
517                                     switch (
518                                     eventAction) {
519                                     case
520                                     MotionEvent.ACTION_DOWN:
521                                     horizontalLeftButton.
522                                     setBackgroundDrawable(getResources().
523                                     getDrawable(R.drawable.
524                                     circular_button_green));
525                                     try {
526                                     jsonObjectValues.put("smcSerial1", (
527                                     speedValue * -1));
528                                     jsonObjectValues.put("smcSerial2", 0);
529                                     jsonObjectValues.put("smcSerial3",

```

```
518 speedValue);
519
    jsonObjectValues.put("smcSerial4", 0);
520
    outputStream.println(jsonObjectValues)
    ;
521
522                                     }
    catch (JSONException e) {
523
        e.printStackTrace();
524
525                                     }
526
527
    break;
528
529                                     case
    MotionEvent.ACTION_UP:
530
531
        horizontalLeftButton.
        setBackgroundDrawable(getResources().
        getDrawable(R.drawable.circular_button));
532
533
        try {
534
            jsonObjectValues.put("smcSerial1", 0);
535
            jsonObjectValues.put("smcSerial2", 0);
536
            jsonObjectValues.put("smcSerial3", 0);
```

```

537         jsonObjectValues.put("smcSerial4", 0);
538         outputStream.println(jsonObjectValues)
539         ;
540     }
541     catch (JSONException e) {
542         e.printStackTrace();
543     }
544     break;
545     }
546     return
547     false;
548     });
549
550     /**
551     * A touch button
552     which sends data to server. (go
553     horizontal right command)
554     */
555     horizontalRightButton.setOnTouchListener(
556     new View.OnTouchListener() {
557         @Override
558         public
559         boolean onTouch(View v, MotionEvent event
560         ) {
561             int

```

```
556 eventAction = event.getAction();
557
558                                     switch (
eventAction) {
559                                     case
MotionEvent.ACTION_DOWN:
560
horizontalRightButton.
setBackgroundDrawable(getResources().
getDrawable(R.drawable.
circular_button_green));
561
try {
562
    jsonObjectValues.put("smcSerial1",
speedValue);
563
    jsonObjectValues.put("smcSerial2", 0);
564
    jsonObjectValues.put("smcSerial3", (
speedValue * -1));
565
    jsonObjectValues.put("smcSerial4", 0);
566
    outputStream.println(jsonObjectValues)
;
567
568                                     }
catch (JSONException e) {
569
    e.printStackTrace();
570
571                                     }
```

```
572
573
    break;
574
575                                     case
    MotionEvent.ACTION_UP:
576
577
        horizontalRightButton.
        setBackgroundDrawable(getResources().
        getDrawable(R.drawable.circular_button));
578
579
    try {
580
        jsonObjectValues.put("smcSerial1", 0);
581
        jsonObjectValues.put("smcSerial2", 0);
582
        jsonObjectValues.put("smcSerial3", 0);
583
        jsonObjectValues.put("smcSerial4", 0);
584
        outputStream.println(jsonObjectValues)
    ;
585
586                                     }
        catch (JSONException e) {
587
            e.printStackTrace();
588
589                                     }
        break;
```

```
590
591                                     }
592                                     return
    false;
593                                     }
594                                     });
595
596
597         lanternButton.
    setOnClickListener(new View.
    OnClickListener() {
598                                     @Override
599                                     public void
    onClick(View v) {
600                                     if (
    lightsOn == false) {
601                                     try {
602
    lanternButton.setBackgroundDrawable(
    getResources().getDrawable(R.drawable.
    rectangular_red_button));
603
604
    jsonObjectValues.put("lights", 1);
605
    outputStream.println(jsonObjectValues);
606
    lightsOn = true;
607                                     }
    catch (JSONException e) {
608                                     e
    .printStackTrace();
609                                     }
```



```
610                                     } else {
611                                     try {
612
        lanternButton.setBackgroundDrawable(
        getResources().getDrawable(R.drawable.
        button_rectangular));
613
614        jsonObjectValues.put("lights", 0);
615        outputStream.println(jsonObjectValues);
616        lightsOn = false;
617                                     }
        catch (JSONException e) {
618                                     e
        .printStackTrace();
619                                     }
620                                     }
621                                     }
622                                     });
623
624        alarmButton.
        setOnTouchListener(new View.
        OnTouchListener() {
625                                     @Override
626                                     public
        boolean onTouch(View v, MotionEvent event
        ) {
627                                     int
        eventAction = event.getAction();
628
629                                     switch (
```

```
629 eventAction) {
630                                     case
        MotionEvent.ACTION_DOWN:
631
632
        try {
633
            alarmButton.setBackgroundDrawable(
                getResources().getDrawable(R.drawable.
                rectangular_main_dark_greenboarder));
634
            jsonObjectValues.put("horn", 1);
635
            outputStream.println(jsonObjectValues)
        ;
636
637                                     }
        catch (JSONException e) {
638
            e.printStackTrace();
639                                     }
640
641
642
        break;
643
644                                     case
        MotionEvent.ACTION_UP:
645
        try {
646
            alarmButton.setBackgroundDrawable(
                getResources().getDrawable(R.drawable.
```

```
646 button_rectangular));
647
        jsonObjectValues.put("horn", 0);
648
        outputStream.println(jsonObjectValues)
        ;
649
650                                     }
        catch (JSONException e) {
651
            e.printStackTrace();
652                                     }
653
        break;
654
655                                     }
656                                     return
        false;
657                                     }
658                                     });
659
660                                     }
661                                     } catch (IOException e) {
662                                     Log.e(debugString, e.
        getMessage());
663                                     } catch (JSONException e)
        {
664                                     e.printStackTrace();
665                                     } finally {
666
667                                     }
668                                     }
669
```

```

670         }.start();
671
672
673         /**A separate thread to handle
UDP stream of images.**/
674         new Thread() {
675             @Override
676             public void run() {
677                 this.setName("UDP Thread"
678
679                 );
680
681                 try {
682
683                     serverSocket = new
684                     DatagramSocket(5555);
685                     isUDPRunning = false;
686                     while (!isUDPRunning)
687                     {
688                         byte[]
689                         receiveData = new byte[57654];
690                         DatagramPacket
691                         receivePacket = new DatagramPacket(
692                         receiveData, receiveData.length);
693                         serverSocket.
694                         receive(receivePacket);
695
696                         final Bitmap bp =
697                         BitmapFactory.decodeByteArray(
698                         receiveData, 0, receiveData.length);
699                         runOnUiThread(new
700                         Runnable() {
701
702                             @Override
703                             public void
704                             run() {

```

```
691     videoImageView.setImageBitmap(bp);
692         }
693     });
694
695
696     }
697     } catch (SocketException
698 e) {
699         e.printStackTrace();
700     } catch (IOException e) {
701         e.printStackTrace();
702     } finally {
703     }
704 }
705 }.start();
706
707
708     /**Separate thread to handle TCP
incoming data */
709     new Thread() {
710         @Override
711         public void run() {
712
713             this.setName("TCP Receive
Thread Manual Mode");
714
715             /**Reads the input stream
and adds the data to a JSONObject*/
716             try {
717
718                 socket =
```

```
718 SocketHandler.getSocket();
719         JSONObject
    jsonObjectReceived = null;
720         InputStreamReader
    inRead = new InputStreamReader(socket.
    getInputStream());
721         BufferedReader
    bufRead = new BufferedReader(inRead);
722
723
    isRecieveThreadRunning = false;
724         while (!
    isRecieveThreadRunning) {
725
726
727             String
    jsonDataMsg = "";
728
729
730             jsonDataMsg =
    bufRead.readLine();
731
732
733             if (jsonDataMsg
    != null && jsonDataMsg.length() > 0) {
734
    jsonObjectReceived = new JSONObject(
    jsonDataMsg);
735
    }
736
737             if (
    jsonObjectReceived != null &&
    jsonObjectReceived.length() > 0) {
```

```
738                                     sbr.
    setMyRecivedData(jsonObjectReceived);
739                                     }
740
741                                     if (jsonDataMsg
    == null) {
742                                     runOnUiThread
    (new Runnable() {
743                                     @Override
744                                     public
    void run() {
745
746
    connectionTextView.setText("" + "Not
    Connected");
747
    connectionTextView.setTextSize(25);
748
    connectionTextView.setBackgroundDrawable(
    getResources().getDrawable(R.drawable.
    red_textview));
749
750                                     }
751                                     });
752                                     }
753                                     }
754
755                                     } catch (IOException e) {
756                                     e.printStackTrace();
757                                     } catch (JSONException e)
    {
758                                     e.printStackTrace();
759                                     }
```

```
760
761         }
762
763         }.start();
764
765     }
766
767
768     /**
769      * Creates an intent to change
770      activity to AutoPilotActivity
771      */
772     public void goToAutopilot() {
773
774         Intent intent = new Intent(this,
775         AutoPilotActivity.class);
776         isUDPRunning = true;
777         isManualModeRunning = true;
778         startActivity(intent);
779         finish();
780
781     }
782
783
784 }
785
```



```
1 package com.example.bakken.  
  platformcontroller2.android.widget;  
2  
3 import java.net.Socket;  
4  
5 /**  
6  * Created by Christer on 19.05.2017.  
7  */  
8  
9 public class SocketHandler {  
10  
11     private static Socket socket;  
12  
13     public static synchronized Socket  
14     getSocket() {  
15         return socket;  
16     }  
17  
18     public static synchronized void  
19     setSocket(Socket socket) {  
20         SocketHandler.socket = socket;  
21     }  
22 }
```

```
1 package com.example.bakken.  
  platformcontroller2;  
2  
3 import android.content.Intent;  
4 import android.support.v7.app.  
  AppCompatActivity;  
5 import android.os.Bundle;  
6 import android.view.View;  
7 import android.widget.Button;  
8 import android.widget.Toast;  
9  
10 import com.example.bakken.  
  platformcontroller2.android.widget.  
  SocketHandler;  
11  
12 import java.io.IOException;  
13 import java.net.Socket;  
14  
15 public class StartupActivity extends  
  AppCompatActivity {  
16  
17     Button startManualModeButton;  
18     Button startAutopilotButton;  
19     SocketHandler socketHandler;  
20  
21     private static Socket socket;  
22     private static final int portnumber =  
  7777;  
23     private static final String hostname =  
  "192.168.0.102";  
24  
25     @Override  
26     protected void onCreate(Bundle
```

```

26 savedInstanceState) {
27     super.onCreate(savedInstanceState)
28     ;
29     setContentView(R.layout.
activity_startup);
30
31     /**Initialize and connect xml to
javacode*/
32     startAutopilotButton = (Button)
findViewById(R.id.buttoStartAutopilot);
33     startManualModeButton = (Button)
findViewById(R.id.buttonStartManualMode);
34
35     /**sets a onClicklistener on
startManualModeButton. Takes the user to
ManualActivity Mode if the socket
36     has connection to the server.*/
37     startManualModeButton.
setOnClickListener(new View.
OnClickListener() {
38         @Override
39         public void onClick(View v) {
40
41             if(socket !=null) {
42                 startManualModeButton.
setBackgroundDrawable(getResources().
getDrawable(R.drawable.
rectangular_main_dark_greenboarder));
43                 startManualMode();
44             }else{
45
46                 Toast.makeText(

```

```

46  getApplicationContext(), "Socket is not
    connected to server", Toast.LENGTH_LONG).
    show();
47
48          }
49      }
50      });
51
52      /**sets a onClicklistener on
    startAutopilotButton. Takes the user to
    AutiopilotActivity if the socket
53          has connection to the server.*/
54      startAutopilotButton.
    setOnClickListener(new View.
    OnClickListener() {
55          @Override
56          public void onClick(View v) {
57              if(socket !=null) {
58                  startAutopilot();
59              }else{
60                  Toast.makeText(
    getApplicationContext(), "Socket is not
    connected to server", Toast.LENGTH_LONG).
    show();
61              }
62          }
63      });
64
65
66
67      new Thread() {
68          @Override
69          public void run() {

```

```
70
71         try {
72
73             /**Connects the
74             client socket to the server socket*/
75             socket = new Socket(
76                 hostname,portnumber);
77             SocketHandler.
78             setSocket(socket);
79             runOnUiThread(new
80                 Runnable() {
81                     @Override
82                     public void run()
83                     {
84                         if(socket.
85                             isConnected()) {
86                             Toast.
87                             makeText(getApplicationContext(),"
88                             Connected to server",Toast.LENGTH_LONG).
89                             show();
90                         }
91                     }
92                 });
93             } catch (IOException e) {
94                 e.printStackTrace();
95             }
96         }
97     }.start();
98
99
```

```
94     }
95
96     /**An intent to start ManualActivity
97     */
98     public void startManualMode() {
99         Intent intentM = new Intent(this,
100             ManualActivity.class);
101         startActivity(intentM);
102     }
103     /**An intent to start
104     AutopilotActivity*/
105     public void startAutopilot() {
106         Intent intentA = new Intent(this,
107             AutopilotActivity.class);
108         startActivity(intentA);
109         finish();
110     }
111 }
112
```

```
1 package com.example.bakken.  
  platformcontroller2.android.widget;  
2  
3 import org.json.JSONObject;  
4  
5 /**  
6  * Created by Christer on 11.05.2017.  
7  */  
8  
9 public class StorageBoxReceived {  
10  
11     private JSONObject myReceivedData =  
12         new JSONObject();  
13  
14     public JSONObject getMyRecivedData() {  
15         return myReceivedData;  
16     }  
17  
18     public void setMyRecivedData(  
19         JSONObject jsonObject) {  
20         this.myReceivedData = jsonObject;  
21     }  
22 }
```

uten navn

```
#include <Wire.h>
#include <Adafruit_Sensor.h>
#include <Adafruit_LSM303_U.h>
#include <Adafruit_L3GD20_U.h>
#include <Adafruit_9DOF.h>
#include <PID_v1.h>
#include <Kalman.h>
#include <ArduinoJson.h>
#include <Filters.h>
#define TCAADDR 0x70
#define UNO 13

/* Assign a unique ID to this sensor at the same time */
Adafruit_9DOF dof = Adafruit_9DOF();
Adafruit_LSM303_Accel_Unified accel1 = Adafruit_LSM303_Accel_Unified(1);
Adafruit_LSM303_Mag_Unified mag1 = Adafruit_LSM303_Mag_Unified(2);
Adafruit_L3GD20_Unified gyro1 = Adafruit_L3GD20_Unified(9);
Adafruit_LSM303_Accel_Unified accel2 = Adafruit_LSM303_Accel_Unified(3);
Adafruit_LSM303_Mag_Unified mag2 = Adafruit_LSM303_Mag_Unified(4);
Adafruit_L3GD20_Unified gyro2 = Adafruit_L3GD20_Unified(10);
Adafruit_LSM303_Accel_Unified accel3 = Adafruit_LSM303_Accel_Unified(5);
Adafruit_LSM303_Mag_Unified mag3 = Adafruit_LSM303_Mag_Unified(6);
Adafruit_L3GD20_Unified gyro3 = Adafruit_L3GD20_Unified(11);
Adafruit_LSM303_Accel_Unified accel4 = Adafruit_LSM303_Accel_Unified(7);
Adafruit_LSM303_Mag_Unified mag4 = Adafruit_LSM303_Mag_Unified(8);
Adafruit_L3GD20_Unified gyro4 = Adafruit_L3GD20_Unified(12);

void tcselect(uint8_t i) {
  if (i > 6) return;

  Wire.beginTransmission(TCAADDR);
  Wire.write(1 << i);
  Wire.endTransmission();
}

// ***** PID init *****
// One PID for each task: Hight, Roll and Pitch.

// ***** Roll PID *****
//Define Variables we'll be connecting to
double RollSetpoint, RollInput, RollOutput; // Setpoint = The value you want. In
our case 0. Input = value from sensor. Output = the controll signal.

//Define the aggressive and conservative Tuning Parameters
double RaggKp = 4, RaggKi = 0.2, RaggKd = 1; // Kp = constant, Ki = integral,
Kd = derivative
double RconsKp = 10, RconsKi = 0.01, RconsKd = 0.001; // Kp = constant, Ki =
integral, Kd = derivative

//Specify the links and initial tuning parameters
PID rollPID(&RollInput, &RollOutput, &RollSetpoint, RconsKp, RconsKi, RconsKd,
```


uten navn

```
DIRECT); // PID initiate
```

```
// ***** Pitch PID *****
```

```
//Define Variables we'll be connecting to
```

```
double PitchSetpoint, PitchInput, PitchOutput; // Setpoint = The value you want.  
In our case 0. Input = value from sensor. Output = the controll signal.
```

```
//Define the aggressive and conservative Tuning Parameters
```

```
double PaggKp = 4, PaggKi = 0.2, PaggKd = 1; // Kp = constant, Ki = integral,  
Kd = derivative
```

```
double PconsKp = 10, PconsKi = 0.01, PconsKd = 0.001; // Kp = constant, Ki =  
integral, Kd = derivative
```

```
//Specify the links and initial tuning parameters
```

```
PID pitchPID(&PitchInput, &PitchOutput, &PitchSetpoint, PconsKp, PconsKi,  
PconsKd, DIRECT); // PID initiate
```

```
double PIDtimer;
```

```
double PIDStopTime;
```

```
// ***** IMU init *****
```

```
/* Update this with the correct SLP for accurate altitude measurements */  
float seaLevelPressure = SENSORS_PRESSURE_SEALEVELHPA;
```

```
double avgRoll; // Holds the average roll
```

```
double avgPitch; // Holds the average pitch
```

```
double angularVRollRad; // Holds the angular roll velocity in rad/s
```

```
double angularVPitchRad; // Holds the angular pitch velocity in rad/s
```

```
double angularVRollDeg; // Holds the angular roll velocity in deg/s
```

```
double angularVPitchDeg; // Holds the angular pitch velocity in deg/s
```

```
double heading; // Holds the IMU heading in degrees.
```

```
// ***** Pressure sensors *****
```

```
int column1PressureSensor = A0; // Pin for reading the pressure sensor inside  
the columns
```

```
int column2PressureSensor = A1; // Pin for reading the pressure sensor  
outside column 2
```

```
int column3PressureSensor = A2; // Pin for reading the pressure sensor inside  
the columns
```

```
int column4PressureSensor = A3; // Pin for reading the pressure sensor inside  
the columns
```

```
int column5PressureSensor = A4; // Pin for reading the pressure sensor  
outside column 5
```

```
int column6PressureSensor = A5; // Pin for reading the pressure sensor inside  
the columns
```

```
double DraftSetpoint;
```

```
double DraftInputPort; // Pressure on port side
```

```
double DraftInputStarboard; // Pressure on starboard side
```

```
double draft; // Holds the average draft between port and starboard sides
```

```

                                uten navn

int draftCM; // Draft in cm

double column1WaterLevel; // Hight of water level in column 1 in cm.
double column2WaterLevel; // Hight of water level outside column 2 in cm.
double column3WaterLevel; // Hight of water level in column 3 in cm.
double column4WaterLevel; // Hight of water level in column 4 in cm.
double column5WaterLevel; // Hight of water level outside column 5 in cm.
double column6WaterLevel; // Hight of water level in column 6 in cm.

// ***** Relays *****

int column1Inn = 6;
int column1Out = 2;
int column3Inn = 7;
int column3Out = 3;
int column4Inn = 8;
int column4Out = 4;
int column6Inn = 9;
int column6Out = 5;

// ***** Edge detect *****

boolean draftCorrDone;
boolean pitchCorrDone;
boolean rollCorrDone;

// ***** Empty tanks *****

int EmptyTanks = 13;
int startEmptyingTanks;

// ***** Timer *****

uint32_t timer;

// ***** Kalman filter *****

Kalman kalmanX; // Create Kalman instances
Kalman kalmanY;

double kalAngleRoll, kalAnglePitch; // Holds the calculated angel using Kalman
filter

// ***** Filter *****

float filterFrequency = 0.02;
double rollFiltered;
double pitchFiltered;
double headingFiltered;
double draftFiltered;
FilterOnePole lowPassFilterRoll(LOWPASS, filterFrequency);
FilterOnePole lowPassFilterPitch(LOWPASS, filterFrequency);

```

```

                                uten navn
FilterOnePole lowPassFilterHeading(LOWPASS, filterFrequency);
FilterOnePole lowPassFilterDraft(LOWPASS, filterFrequency);

// ***** Timer *****

double timePassed;
double nowTime;
double waitTime;

void setup() {

    // ***** Serial setup *****
    Serial.begin( 9600 );    // start the serial port

    // ***** Start I2C *****
    Wire.begin();

    // ***** Initialise the sensors *****
    initSensors();

    // ***** Empty tanks *****

    pinMode(EmptyTanks, INPUT);

    // ***** Starts timer *****

    timer = micros();

    // ***** Kalman filter *****

    kalmanX.setAngle(avgRoll);
    kalmanY.setAngle(avgPitch);

    // ***** Setpoints *****

    RollInput = kalAngleRoll; // The roll angle sendt to rollPID
    PitchInput = kalAnglePitch; // The pitch angle sendt to pitchPID

    DraftSetpoint = 30 ; // (30) Setpoint for platform draft in cm
    RollSetpoint = 0; // Setpoint for angle in degrees
    PitchSetpoint = 0; // Setpoint for angle in degrees

    //turn the roll PID on
    rollPID.SetMode(AUTOMATIC);
    rollPID.SetOutputLimits(0, 100);

    //turn the pitch PID on
    pitchPID.SetMode(AUTOMATIC);
    pitchPID.SetOutputLimits(0, 100);

    PIDtimer = millis();

```

```

                                uten navn

PIDStopTime = 0;

// ***** Configure digital pins *****

for (int i = 2; i <= 9; i++) {
    pinMode(i, OUTPUT);
    digitalWrite(i, HIGH);
}
// ***** Timer *****

nowTime = millis();
waitTime = 1000;

}

void loop() {
    Serial.print("Pitch: "); Serial.print(pitchFiltered); Serial.print(" Roll: ");
    Serial.print(rollFiltered); Serial.print(" Depth: "); Serial.print(draftCM);
    Serial.print(" PIDtimer: "); Serial.println(PIDtimer);
    getGyroData();
    kalman();
    getPressureData();
    SendDataToServer();
    sendDataToGimbal();

    RollInput = abs(kalAngleRoll) * (-1); // The roll angle sendt to rollPID
    PitchInput = abs(kalAnglePitch) * (-1); // The pitch angle sendt to pitchPID
    rollPID.Compute();
    pitchPID.Compute();

    startEmptyingTanks = true; //digitalRead(EmptyTanks);

    if (startEmptyingTanks) {
        while (true) {
            digitalWrite(column1Out, LOW);
            digitalWrite(column3Out, LOW);
            digitalWrite(column4Out, LOW);
            digitalWrite(column6Out, LOW);
            Serial.println("Emptying tanks");
        }
    }

    // Tests if platform is within limits (Roll, pitch and draft)
    if (draftFiltered > (DraftSetpoint + 5) || draftFiltered < (DraftSetpoint -
5))
    {
        draftCorrDone = false;

        while (!draftCorrDone) {
            getGyroData();
            getPressureData();

```

```

                                uten navn

    SendDataToServer();
    sendDataToGimbal();
    kalman();
    // Serial.print(" draft corr ");
    //Serial.println(draft);
    correctDraft(); // Starts draft correction
    if (draft < (DraftSetpoint + 1) && draft > (DraftSetpoint - 1) ||
columnsEmpty())
    {
        draftCorrDone = true;
        // Serial.println("Stopping pumps");
        stopPumps();
    }
}

if (rollFiltered > (RollSetpoint + 0.8) || rollFiltered < (RollSetpoint -
0.8))
{
    rollCorrDone = false;

    while (!rollCorrDone) {
        getGyroData();
        // Serial.print("Roll corr. KalAngleRoll: ");
Serial.println(kalAngleRoll);
        getPressureData();
        SendDataToServer();
        sendDataToGimbal();
        kalman();
        correctRoll(); // Starts roll correction
        if (levelAlert()) {
            stopPumps();
        }
        if (kalAngleRoll > (RollSetpoint - 0.6) && kalAngleRoll < (RollSetpoint +
0.6))
        {
            rollCorrDone = true;
            //Serial.println("Stopping pumps");
            stopPumps();
            lowPassFilterRoll.setToNewValue(0);
        }
    }
}

else if (pitchFiltered > (PitchSetpoint + 0.8) || pitchFiltered <
(PitchSetpoint - 0.8))
{
    pitchCorrDone = false;

    while (!pitchCorrDone) {
        getGyroData();
        //Serial.print(" Pitch corr, kalAnglePitch: ");

```

uten navn

```
Serial.println(kalAnglePitch);
    getPressureData();
    SendDataToServer();
    sendDataToGimbal();
    kalman();
    correctPitch(); // Starts pitch correction
    //Serial.print(" PitchOutput: "); Serial.println(PitchOutput);
    if (levelAlert()) {
        stopPumps();
    }
    if (kalAnglePitch > (PitchSetpoint - 0.6) && kalAnglePitch <
(PitchSetpoint + 0.6))
    {
        pitchCorrDone = true;
        // Serial.println("Stopping pumps");
        stopPumps();
        lowPassFilterPitch.setToNewValue(0);
    }
}
}
```

```
String getGyroData() {
    /* Get a new sensor event */
    sensors_event_t accel_event;
    sensors_event_t mag_event;
    sensors_vec_t    orientation;
    sensors_event_t gyro_event;

    double roll1; double pitch1; double head1;
    double roll2; double pitch2; double head2;
    double roll3; double pitch3; double head3;
    double roll4; double pitch4; double head4;
    double angularVRollRad1; double angularVRollRad2; double angularVRollRad3;
double angularVRollRad4;
    double angularVPitchRad1; double angularVPitchRad2; double angularVPitchRad3;
double angularVPitchRad4;
    double angularVRollDeg1; double angularVRollDeg2; double angularVRollDeg3;
double angularVRollDeg4;
    double angularVPitchDeg1; double angularVPitchDeg2; double angularVPitchDeg3;
double angularVPitchDeg4;

    tcselect(2);
    accel1.getEvent(&accel_event);
    if (dof.accelGetOrientation(&accel_event, &orientation))
    {
        roll1 = orientation.roll;
        pitch1 = orientation.pitch;
    }
    mag1.getEvent(&mag_event);
    if (dof.magGetOrientation(SENSOR_AXIS_Z, &mag_event, &orientation))
```

```

                                uten navn
{
    head1 = orientation.heading;
}
gyro1.getEvent(&gyro_event);
angularVRollRad1 = gyro_event.gyro.x;
angularVPitchRad1 = gyro_event.gyro.y;
angularVRollDeg1 = (angularVRollRad * 4068) / 71;
angularVPitchDeg1 = (angularVPitchRad * 4068) / 71;

tcaselect(3);
accel2.getEvent(&accel_event);
if (dof.accelGetOrientation(&accel_event, &orientation))
{
    roll2 = orientation.roll;
    pitch2 = orientation.pitch;
}
mag2.getEvent(&mag_event);
if (dof.magGetOrientation(SENSOR_AXIS_Z, &mag_event, &orientation))
{
    head2 = orientation.heading;
}
gyro2.getEvent(&gyro_event);
angularVRollRad2 = gyro_event.gyro.x;
angularVPitchRad2 = gyro_event.gyro.y;
angularVRollDeg2 = (angularVRollRad * 4068) / 71;
angularVPitchDeg2 = (angularVPitchRad * 4068) / 71;

tcaselect(4);
accel3.getEvent(&accel_event);
if (dof.accelGetOrientation(&accel_event, &orientation))
{
    roll3 = orientation.roll;
    pitch3 = orientation.pitch;
}
mag3.getEvent(&mag_event);
if (dof.magGetOrientation(SENSOR_AXIS_Z, &mag_event, &orientation))
{
    head3 = orientation.heading;
}
gyro1.getEvent(&gyro_event);
angularVRollRad3 = gyro_event.gyro.x;
angularVPitchRad3 = gyro_event.gyro.y;
angularVRollDeg3 = (angularVRollRad * 4068) / 71;
angularVPitchDeg3 = (angularVPitchRad * 4068) / 71;

tcaselect(5);
accel4.getEvent(&accel_event);
if (dof.accelGetOrientation(&accel_event, &orientation))
{
    roll4 = orientation.roll;
    pitch4 = orientation.pitch;
}

```

```

                                uten navn
mag4.getEvent(&mag_event);
if (dof.magGetOrientation(SENSOR_AXIS_Z, &mag_event, &orientation))
{
    head4 = orientation.heading;
}
gyro1.getEvent(&gyro_event);
angularVRollRad4 = gyro_event.gyro.x;
angularVPitchRad4 = gyro_event.gyro.y;
angularVRollDeg4 = (angularVRollRad * 4068) / 71;
angularVPitchDeg4 = (angularVPitchRad * 4068) / 71;

angularVRollDeg = ((angularVRollDeg1 + angularVRollDeg2 + angularVRollDeg3 +
angularVRollDeg4) / 4);
angularVPitchDeg = ((angularVPitchDeg1 + angularVPitchDeg2 + angularVPitchDeg3
+ angularVPitchDeg4) / 4);

// Average values from four IMU's
avgRoll = ((roll1 + roll2 + roll3 + roll4) / 4);
avgPitch = ((pitch1 + pitch2 + pitch3 + pitch4) / 4);
heading = ((head1 + head2 + head3 + head4) / 4);

lowPassFilterRoll.input(avgRoll);
lowPassFilterPitch.input(avgPitch);
lowPassFilterHeading.input(heading);

rollFiltered = lowPassFilterRoll.output();
pitchFiltered = lowPassFilterPitch.output();
headingFiltered = lowPassFilterHeading.output();

//    Serial.print(head1);
//    Serial.print(" ");
//    Serial.print(head2);
//    Serial.print(" ");
//    Serial.print(head3);
//    Serial.print(" ");
//    Serial.println(head4);
}

void initSensors()
{
    /* Initialise the 1st sensor */
    tcselect(2);
    accel1.begin();
    mag1.begin();
    gyro1.begin();
    tcselect(2);
    if (!accel1.begin())
    {
        Serial.println(F("Ooops, accel1....."));
        while (1);
    }
    if (!mag1.begin())

```


uten navn

```
{
  Serial.println("Ooops, no LSM303 detected ... Check your wiring!");
  while (1);
}
if (!gyro1.begin())
{
  Serial.print("Ooops, no L3GD20 detected ... Check your wiring or I2C
ADDR!");
  while (1);
}

/* Initialise the 2nd sensor */
tcselect(3);
accel2.begin();
mag2.begin();
gyro2.begin();
tcselect(3);
if (!accel2.begin())
{
  Serial.println(F("Ooops, accel1....."));
  while (1);
}
if (!mag2.begin())
{
  Serial.println("Ooops, no LSM303 detected ... Check your wiring!");
  while (1);
}
if (!gyro2.begin())
{
  Serial.print("Ooops, no L3GD20 detected ... Check your wiring or I2C
ADDR!");
  while (1);
}

/* Initialise the 3rd sensor */
tcselect(4);
accel3.begin();
mag3.begin();
gyro3.begin();
tcselect(4);
if (!accel3.begin())
{
  Serial.println(F("Ooops, accel1....."));
  while (1);
}
if (!mag3.begin())
{
  Serial.println("Ooops, no LSM303 detected ... Check your wiring!");
  while (1);
}
if (!gyro3.begin())
{
```

```

                                uten navn
    Serial.print("Oops, no L3GD20 detected ... Check your wiring or I2C
ADDR!");
    while (1);
}

/* Initialise the 4th sensor */
tcaselect(5);
accel4.begin();
mag4.begin();
gyro4.begin();
tcaselect(5);
if (!accel4.begin())
{
    Serial.println(F("Oops, accel1....."));
    while (1);
}
if (!mag4.begin())
{
    Serial.println("Oops, no LSM303 detected ... Check your wiring!");
    while (1);
}
if (!gyro4.begin())
{
    Serial.print("Oops, no L3GD20 detected ... Check your wiring or I2C
ADDR!");
    while (1);
}
}

void getPressureData()
{
    double sensorValue1 = 0; double mappedValue1; double constValue1; // variable
to store the value coming from the sensor
    double sensorValue3 = 0; double mappedValue3; double constValue3; // variable
to store the value coming from the sensor
    double sensorValue4 = 0; double mappedValue4; double constValue4; // variable
to store the value coming from the sensor
    double sensorValue6 = 0; double mappedValue6; double constValue6; // variable
to store the value coming from the sensor
    double sensorValuePortside = 0; double mappedValuePortside; double
constValuePortside; // variable to store the value coming from the sensor
    double sensorValueStarboardSide = 0; double mappedValueStarboardSide; double
constValueStarboardSide; // variable to store the value coming from the sensor

    // read the value from the sensor:
    sensorValue1 = analogRead(column1PressureSensor);
    sensorValuePortside = analogRead(column2PressureSensor);
    sensorValue3 = analogRead(column3PressureSensor);
    sensorValue4 = analogRead(column4PressureSensor);
    sensorValueStarboardSide = analogRead(column3PressureSensor);
    sensorValue6 = analogRead(column6PressureSensor);

```

```

        uten navn
    //Serial.print(sensorValue1); Serial.print(" ");
    Serial.print(sensorValuePortside); Serial.print(" ");
    Serial.print(sensorValue3); Serial.print(" "); Serial.print(sensorValue4);
    Serial.print(" "); Serial.print(sensorValueStarboardSide); Serial.print(" ");
    Serial.println(sensorValue6);

    // Mapes values to correct scale and constrain them between max and min
    mappedValue1 = map(sensorValue1, 270, 440, 0, 172); constValue1 =
    constrain(mappedValue1, 0, 1000);
    mappedValuePortside = map(sensorValuePortside, 278, 419, 0, 141);
    constValuePortside = constrain(mappedValuePortside, 0, 1000);
    mappedValue3 = map(sensorValue3, 271, 425, 0, 155); constValue3 =
    constrain(mappedValue3, 0, 1000);
    mappedValue4 = map(sensorValue4, 313, 472, 0, 159); constValue4 =
    constrain(mappedValue4, 0, 1000);
    mappedValueStarboardSide = map(sensorValueStarboardSide, 283, 425, 0, 142);
    constValueStarboardSide = constrain(mappedValueStarboardSide, 0, 1000);
    mappedValue6 = map(sensorValue6, 272, 442, 0, 170); constValue6 =
    constrain(mappedValue6, 0, 1000);

    // Calculates the water hight in cm
    column1WaterLevel = constValue1 * 0.349;
    column2WaterLevel = constValuePortside * 0.39;
    column3WaterLevel = constValue3 * 0.387;
    column4WaterLevel = constValue4 * 0.377;
    column5WaterLevel = constValueStarboardSide * 0.387;
    column6WaterLevel = constValue6 * 0.353;

    draft = (column2WaterLevel + column5WaterLevel) / 2;
    lowPassFilterDraft.input(draft);
    draftFiltered = lowPassFilterDraft.output();
    // Serial.print( "Draft: "); Serial.println(draft);
    draftCM = (draft * 0.29167) - 22;

    //Serial.print(column1WaterLevel); Serial.print(" ");
    Serial.print(column2WaterLevel); Serial.print(" ");
    Serial.print(column3WaterLevel); Serial.print(" ");
    Serial.print(column4WaterLevel); Serial.print(" ");
    Serial.print(column5WaterLevel); Serial.print(" ");
    Serial.println(column6WaterLevel);
}

void correctDraft() // Corrects the draft of the platform
{
    if (draft < DraftSetpoint) { // Tests if platform is to high in the water
        digitalWrite(column1Inn, LOW);
        digitalWrite(column3Inn, LOW);
        digitalWrite(column4Inn, LOW);
        digitalWrite(column6Inn, LOW);
    }
}

```

```

                                uten navn
    if (draft > DraftSetpoint && !columnsEmpty()) { // Tests if platform is to low
in the water
        digitalWrite(column1Out, LOW);
        digitalWrite(column3Out, LOW);
        digitalWrite(column4Out, LOW);
        digitalWrite(column6Out, LOW);
    }
}

void correctRoll()
{
    RollInput = abs(kalAngleRoll) * (-1); // The roll angle sendt to rollPID
    rollPID.Compute();
    if (kalAngleRoll > (RollSetpoint)) {
        // Serial.print("ROLL OVER SET -> Output: ");
        // Serial.print(RollOutput);
        if (PIDtime(RollOutput)) {
            stopPumps();
        } else {
            digitalWrite(column1Inn, LOW);
            digitalWrite(column3Inn, LOW);
        }
    }

    if (kalAngleRoll < (RollSetpoint)) {
        // Serial.print("ROLL UNDER SET -> Output: ");
        // Serial.print(RollOutput);
        if (PIDtime(RollOutput)) {
            stopPumps();
        } else {
            digitalWrite(column4Inn, LOW);
            digitalWrite(column6Inn, LOW);
        }
    }
}

void correctPitch()
{
    PitchInput = abs(kalAnglePitch) * (-1); // The pitch angle sendt to pitchPID
    pitchPID.Compute();
    if (kalAnglePitch > (PitchSetpoint)) {
        // Serial.print("PITCH OVER SET -> Output: ");
        // Serial.print(PitchOutput);
        if (PIDtime(PitchOutput)) {
            stopPumps();
        } else {
            digitalWrite(column3Inn, LOW);
            digitalWrite(column4Inn, LOW);
        }
    }

    if (kalAnglePitch < (PitchSetpoint)) {

```

```

        uten navn
    // Serial.print("PITCH UNDER SET -> Output: ");
    // Serial.print(PitchOutput);
    if (PIDtime(PitchOutput)) {
        stopPumps();
    } else {
        digitalWrite(column1Inn, LOW);
        digitalWrite(column6Inn, LOW);
    }
}
}

void stopPumps()
{
    // If platform has correct hight then:
    // Serial.println(" STOP PUMPS ");
    digitalWrite(column1Inn, HIGH);
    digitalWrite(column3Inn, HIGH);
    digitalWrite(column4Inn, HIGH);
    digitalWrite(column6Inn, HIGH);
    digitalWrite(column1Out, HIGH);
    digitalWrite(column3Out, HIGH);
    digitalWrite(column4Out, HIGH);
    digitalWrite(column6Out, HIGH);
}

// Sends data to server.
void SendDataToServer()
{
    // Data starts with a "S" to identify the subsystem. "S" for Stability system.
    StaticJsonBuffer<500> jsonBuffer;
    JsonObject& dataMessage = jsonBuffer.createObject();
    dataMessage["address"] = "S";
    dataMessage["heading"] = headingFiltered;
    //dataMessage.printTo(Serial);
    // Serial.println(head1);
}

void sendDataToGimbal() {
    int intKalAngleRoll = (int)kalAngleRoll;
    int intKalAnglePitch = (int)kalAnglePitch;

    Wire.beginTransaction(UNO);
    Wire.write(intKalAngleRoll);
    Wire.write((intKalAngleRoll >> 8));
    Wire.write(intKalAnglePitch);
    Wire.write((intKalAnglePitch >> 8));
    Wire.endTransmission(UNO);
}

void kalman() {
    double dt = (double)(micros() - timer) / 1000000; // Calculate delta time

```

```

                                uten navn

timer = micros();

kalAngleRoll = kalmanX.getAngle(avgRoll, angularVRollDeg, dt);
kalAnglePitch = kalmanY.getAngle(avgPitch, angularVPitchDeg, dt);
}

boolean levelAlert() {
    boolean levelAlert = false;
    if (column1WaterLevel > 40 || column3WaterLevel > 40 || column4WaterLevel > 40
|| column5WaterLevel > 40) {
        levelAlert = true;
    }
    else levelAlert = false;
    return levelAlert;
}

boolean PIDtime(double PIDOutput) {
    boolean StopPID = false;
    //Serial.print("PIDtimer: "); Serial.println(PIDtimer);
    PIDtimer = millis();

    // Serial.print(" PIDtimer: "); Serial.print(PIDtimer); Serial.print("
PIDStopTime: "); Serial.println(PIDStopTime);

    if (PIDtimer > PIDStopTime) {
        PIDStopTime = PIDtimer + (PIDOutput * 2.5);
        // Serial.print(" PIDStopTime: "); Serial.println(PIDStopTime);
        StopPID = true;
    }
    return StopPID;
}

boolean columnsEmpty() {
    boolean isEmpty = false;
    if (column1WaterLevel < 2 || column3WaterLevel < 2 || column4WaterLevel < 2 ||
column6WaterLevel < 2) {
        isEmpty = true;
    }
    return isEmpty;
}

```

uten navn

```
#include <SoftwareSerial.h>
#include <ArduinoJson.h>

#define rxPin 3 // pin 3 connects to smcSerial TX (not used in this example)
#define txPin1 4 // pin 4 connects to smcSerial RX
#define txPin2 5 // pin 4 connects to smcSerial RX
#define txPin3 6 // pin 4 connects to smcSerial RX
#define txPin4 7 // pin 4 connects to smcSerial RX
SoftwareSerial smcSerial1 = SoftwareSerial(rxPin, txPin1); //rightThruster
SoftwareSerial smcSerial2 = SoftwareSerial(rxPin, txPin2); //frontThruster
SoftwareSerial smcSerial3 = SoftwareSerial(rxPin, txPin3); //leftThruster
SoftwareSerial smcSerial4 = SoftwareSerial(rxPin, txPin4); //backThruster

#define lightPin 8 // Lanterns
#define hornPin 9 // Horn

//Verdi for farten vi setter til thrusterne
int frontThrusterSpeedInt;
int backThrusterSpeedInt;
int rightThrusterSpeedInt;
int leftThrusterSpeedInt;
String readString;
char incomingBytes[8];
char c;

// required to allow motors to move
// must be called when controller restarts and after any error
void exitSafeStart()
{
    smcSerial1.write(0x83);
    smcSerial2.write(0x83);
    smcSerial3.write(0x83);
    smcSerial4.write(0x83);
}

// speed should be a number from -3200 to 3200
void frontThrusterSpeed(int speed)
{
    if (speed < 0)
    {
        smcSerial2.write(0x86);
        speed = -speed; // make speed positive
    }
    else
    {
        smcSerial2.write(0x85); // motor forward command
    }
    smcSerial2.write(speed & 0x1F);
    smcSerial2.write(speed >> 5);
}

void rightThrusterSpeed(int speed)
```

uten navn

```
{
  if (speed < 0)
  {
    smcSerial1.write(0x86);
    speed = -speed; // make speed positive
  }
  else
  {
    smcSerial1.write(0x85); // motor forward command
  }
  smcSerial1.write(speed & 0x1F);
  smcSerial1.write(speed >> 5);
}

void leftThrusterSpeed(int speed) {
  if (speed < 0)
  {
    smcSerial3.write(0x86);
    speed = -speed; // make speed positive
  }
  else
  {
    smcSerial3.write(0x85); // motor forward command
  }
  smcSerial3.write(speed & 0x1F);
  smcSerial3.write(speed >> 5);
}

void backThrusterSpeed(int speed) {
  if (speed < 0)
  {
    smcSerial4.write(0x86);
    speed = -speed; // make speed positive
  }
  else
  {
    smcSerial4.write(0x85); // motor forward command
  }
  smcSerial4.write(speed & 0x1F);
  smcSerial4.write(speed >> 5);
}

void setup()
{
  // initialize software serial object with baud rate of 19.2 kbps
  smcSerial1.begin(19200);
  smcSerial2.begin(19200);
  smcSerial3.begin(19200);
  smcSerial4.begin(19200);
  Serial.begin(9600);

  // the Simple Motor Controller must be running for at least 1 ms
```



```

                                uten navn
// before we try to send serial data, so we delay here for 5 ms
delay(5);
// if the Simple Motor Controller has automatic baud detection
// enabled, we first need to send it the byte 0xAA (170 in decimal)
// so that it can learn the baud rate

smcSerial1.write(0xAA);
smcSerial2.write(0xAA); // send baud-indicator byte
smcSerial3.write(0xAA);
smcSerial4.write(0xAA);

// next we need to send the Exit Safe Start command, which
// clears the safe-start violation and lets the motor run
exitSafeStart(); // clear the safe-start violation and let the motor run
frontThrusterSpeed(0);
backThrusterSpeed(0);
leftThrusterSpeed(0);
rightThrusterSpeed(0);

pinMode(lightPin, OUTPUT);
pinMode(hornPin, OUTPUT);
}

void loop() {

    DynamicJsonBuffer jsonBuffer;
    if (Serial.available() > 0) {
        c = Serial.read();
        if (c == '{') { // start of packet
            readString = c;
            while (c != '}') { // until end of packet
                if (Serial.available() > 0) {
                    c = Serial.read();
                    readString += c;
                }
            }
        }
    }
}

JsonObject& root = jsonBuffer.parseObject(readString);

String fremS = root["smcSerial2"];
String bakS = root["smcSerial4"];
String rightS = root["smcSerial1"];
String venstreS = root["smcSerial3"];
String lightsS = root["lights"];
String hornS = root["horn"];

int frem = fremS.toInt();
int bak = bakS.toInt();
int right = rightS.toInt();
int venstre = venstreS.toInt();

```

uten navn

```
int lights = lightsS.toInt();
int horn = hornS.toInt();

frontThrusterSpeed(frem); // Green, second engine controller (Colour of wire
and engine controller from bottom)
backThrusterSpeed(bak); // White, third engine controller
rightThrusterSpeed(right); // Yellow, top engine controller
leftThrusterSpeed(venstre); // Red, first engine controller

if (lights == 1) {
    digitalWrite(lightPin, LOW);
}
else digitalWrite(lightPin, HIGH);

if (horn == 1) {
    digitalWrite(hornPin, LOW);
}
else digitalWrite(hornPin, HIGH);
}
```