INITIAL DEVELOPMENT OF AN AUTONOMOUS UNDERWATER VEHICLE

Jeppe Dam
Nick Østergaard
Lasse Høj Nielsen
Simon Als Nielsen
Brian Bach Nielsen
Rasmus Lundgaard Christensen

11gr502
Aalborg University 2011
Department of Electronic Systems

AALBORG UNIVERSITY

Institute of Electronic Systems
Electronics and Information Technology
Fredrik Bajers vej 7
9220 Aalborg Øst
Phone 99 40 86 00
http://es.aau.dk

Title:
Initial development of an Autonomous Underwater Vehicle

Theme:
Distributed embedded systems for physical systems

Projectperiod:
P5, fall semester 2011

Projectgroup:
502

Participants:
Jeppe Dam
Nick Østergaard
Lasse Høj Nielsen
Simon Als Nielsen
Brian Bach Nielsen
Rasmus Lundgaard Christensen

Supervisor:
Jesper Abildgaard Larsen

Number of printed copies: 8

Number of pages: 152

Appendices: 12 + 1 CD

Finished on: May 3, 2012

Synopsis:

This project documents the development of an Autonomous Underwater Vehicle (AUV). The AUV is based on a remote controlled submersible that is modified to navigate a predetermined three-dimensional route. The AUV's electronics are based on an ARM7 development board and a custom shield developed throughout this project. The shield includes an Inertial Measurement Unit (IMU), a pressure sensor and a radio communication module. The measurements of the IMU are used for calculating the position and orientation of the AUV, while a manometer is used for further input for depth calculations; the radio communication module is used to send way points to the AUV. A model of the physical system has been created and a control system for this has been made. This has been simulated and verified. In the model the AUV is able to manoeuvre in three dimensions.

Preface

This report documents the development of a platform for underwater research in the form of an AUV. This project has been researched and written by group 502 at 5th semester electronics and IT at Aalborg University (AAU). The theme of the semester is "Distributed embedded systems for physical systems". The project have been researched and written between the 2nd of September and the 22nd of December 2011.

Thanks to

Throughout this project some people have been specially helpful, and therefore a special thanks is given to:

- AASI Svømmeklub for use of their pool to test the AUV
- Aalborg Energie Technick A/S for a computer to use as ground station
- Department of Civil Engineering for using their wave laboratory

Brian Bach Nielsen

Rasmus Lundgaard Christensen

Jeppe Dam

Lasse Høj Nielsen

Simon Als Nielsen

Nick Østergaard

Reading guide

The following report is divided into parts. The parts divide the report into the general working phases of the project i.e. *preliminary analysis, design, implementation, summation* and lastly *appendix* has its own part as well.

The parts are divided into chapters according to different aspects of the project.

Citations in the report is done by the Harvard method and the list of references can be found on page 138. The elements of the list of references are sorted by author.

Acronyms are written to their full extend with the acronym in parentheses, thereafter only the acronym are used. The list of acronyms can be found on page xiii.

Attached the report is a CD, which contains copies of web references and other digital files (source code, scripts and raw measurement data) that could be of interest to the reader. In some places in the report there will be a reference to the CD; this will look like this: `/path-to-file`.

In this project the submarine will be referred to as a stock submersible when it is in its initial configuration, described in section 1.3 on page 4, and the AUV in the modified configuration.

Movement terms used in the report

During this report, different terms are used to represent the way in which a maritime vessel moves. This is illustrated in figure 1 on page vii and are described as follows.

The vessel is put into a three-dimensional diagram where the vessel is placed with the centre of gravity in $[0, 0, 0]$. See figure 1 on page vii.

- The movement along the x-axis (denoted ${}^{s}x$) is called surge, and rotation about it is called roll.
- The movement along the y-axis (denoted ${}^{s}y$) is called sway, and rotation about it is called pitch.
- The movement along the z-axis (denoted ${}^{s}z$) is called heave, and rotation about it is called yaw.

These notations describes the vessels position in a cartesian coordinate system as well as the vessels orientation along the axises.

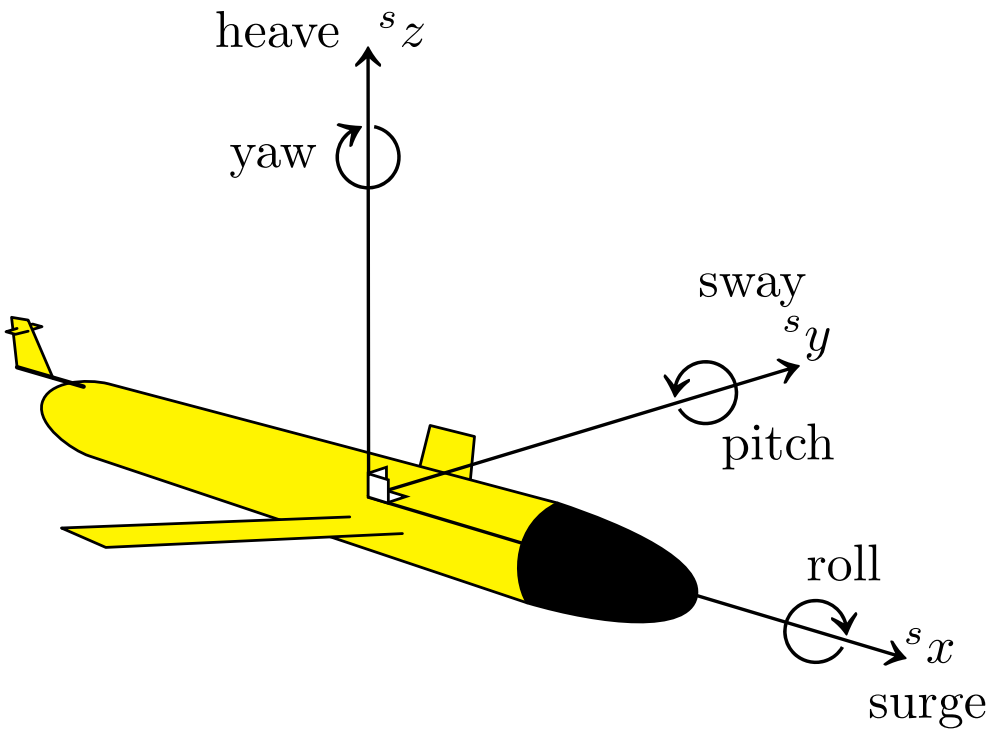


Figure 1: A vessel inserted in a cartesian coordinate system with the centre of gravity as $[0, 0, 0]$

Contents

List of acronyms

AAU Aalborg University

ADC Analog to Digital Converter

AUV Autonomous Underwater Vehicle

CR Carriage Return

EOD End Of Data

FreeRTOS Free Real Time Operating System

GPS Global Positioning System

GSM Global System for Mobile Communications

GUI Graphical User Interface

IMU Inertial Measurement Unit

IO Input Output

LED Light Emitting Diode

LF Line Feed

MISO Master In Slave Out

MOSI Master Out Slave In

MSB Most Significant Bit

NSS Slave Select

PID Proportional-Integral-Derivative

PWM Pulse Width Modulation

RPS rounds per second

SPCK Serial Clock

SPI Serial Peripheral Interface

UART Universal Asynchronous Receiver/Transmitter

UNESCO United Nations Educational, Scientific and Cultural Organization

VHF Very High Frequency

Part I

Preliminary analysis

This part describes the preliminary analysis leading up to the design of the AUV. This includes analysis of the problem, requirement specification and use cases.

1 Introduction

In 2005 a submarine crashed into an underwater mountain, killing one and injuring several others. This could have been avoided if sufficient amounts of data existed on the ocean. Even though the ocean covers roughly 2/3 of the earths surface, 90 % of it remains unexplored[1].

In recent years several companies have begun exploring the ocean. For instance oil companies want to know where the oil is located, environmentalists want to know how climate changes effect the lives of fish and mammals living in the sea, and lastly biologists would like to know how the change of pH-value or salinity changes the lives of the plants in the area; all of which would benefit from an exploration of the ocean. The scope of this project is to develop a prototype platform that eventually is able to survey the North Sea.

1.1 Preliminary analysis

To do research and inspection on the bottom of the sea, there are several ways that can be used where the different advantages and the disadvantages are listed in this chapter. One of the methods could be to use sonar equipment on a surface vessel. The advantages in this case, are that it is easy to implement on vessels that are already sailing the oceans, whereas areas that are not navigated frequently requires a research vessel for analysis of these areas. The downside clearly outweighs the upside as it would take an entire fleet to do maritime environmental research of these areas, and the only upside being that it is easy to mount on the vessels that already navigates the sea.

Another method is to use satellites or aircraft for geomapping of the ocean floor, where satellites in orbit maps the sea. A downside to this is that the satellites are expensive to build and launch. Furthermore it is hard to measure the environmental parameters from space - for instance the pH value, the currents, the salinity of the oceans and so forth. This makes it hard for aircraft and satellites to fulfil the criteria as to do research of the oceans.

A third method is to develop an autonomous underwater drone also called an AUV. This vehicle is equipped with the appropriate sensory equipment, and are sent on a mission to collect the specified data. These data can then be wirelessly transmitted to a ground station. The biggest advantage of an AUV, is that it does not require any man-hours when it is sailing, and transmits the data, whenever it surfaces.

As this projects goal fits perfectly into developing an AUV, it is chosen to look deeper into this idea, meaning that an AUV will be described and developed throughout this report.

[1] `http://news.nationalgeographic.com/news/2005/02/0217_050217_seamap.html`, September 12, 2011

1.2 Initial problem

The following problem is derived from the previous section.

What will it take to develop an AUV that can sail across the North Sea, whilst measuring internal and external parameters?

This means that the AUV will need to control itself during the sailing and always know its direction. It needs to measure internal parameters, e.g. battery level, and it needs to connect with external measurement equipment. All of this will be taken into account through the further analysis.

1.3 Mechanical hardware

The mechanical hardware used in this project is presented here, to give an overview of the submersible that is used. The stock submersible is named the *Neptune SB-1* by the manufacturer—Thunder Tiger Corporation. It has got an outer hull and a static diving system meaning that it changes its buoyancy. The system is driven by a ballast tank with a pump and motor unit which can control the buoyancy. It is controlled via a standard hobby remote control system.

Project relevant physical specifications of the stock submersible are given below. For specific applications in this project, these specifications are needed. These specifications are outlined in table 1.1. Some of these hard specifications given by the manufacturer have been changed including the mass.

Property	**Value**
Mass	7.3 kg
Volume of cylinder	4.85 l
Overall length	770 mm
Width	230 mm (170 mm without beam)
Depth	200 mm
Height	240 mm
Propulsion motor	12 V
Speed	0.56 m/s submerged
Maximum diving depth	10 m (Mechanical limit)
Power supply	12 V, 2.2 Ah (sealed lead-acid battery)
Ballast tank	240 ml
Ballast pump	6 ml/s

Table 1.1: Basic specifications for the stock submersible

1.4 Initial requirements

For the AUV to complete the tasks set up in the previous sections, this section will list some basic functional requirements the AUV has to fulfil and why they are important.

Structural requirements

For the AUV to be able to research the ocean in a productive manner it needs to be able to dive as far down as the ocean reaches, and deliver power to potential scientific payload

- **Diving capability** – to dive to the the entire depth of about 200 m maximum for the North Sea[2].
- **Power system** – to deliver sufficient power to manoeuvring the AUV, propulsion and delivering power to a potential science payload.

Navigation requirements

To navigate autonomously, the AUV has to estimate its position, attitude, speed and direction.

- **Position** – to estimate the current position.
- **Attitude** – to estimate which way it is pointing.
- **Direction** – to estimate which way it is going, this may differ from its attitude due to currents in the sea.
- **Speed** – to give an estimate of arrival time and to calculate the position as exact as possible.

Manoeuvring

The AUV will need a high level of autonomy to be able to navigate under water where communication with a ground station can not be assumed to be functional.

- **Vertical movement** – to get from the surface to the ocean floor and back, heave.
- **Forward/backwards** – to travel along a horisontal line, surge.
- **Turn left and right** – to turn about the vertical axis, yaw.
- **Autonomy** – to a degree that allows the AUV to travel by it self while out of radio contact.

[2] `http://www.worldatlas.com/aatlas/infopage/northsea.htm`, September 27, 2011

Communication

The AUV will need means to communicate with a ground station to exchange data e.g. measured data, error reports and new waypoints.

- **Data communication** – to communicate with a ground station when at the water surface.

1.5 Limitations

The limitations are set up to make the project more tangible and fit the projects timespan. As mentioned in section 1.3 on page 4, a stock submersible is used for this project.

As a consequence of the AUV's mechanical system, which forms the mechanical platform for the electronics, is a stock submersible, it introduces some constraints that will limit the specfications of the hardware platform. These limitations will affect the requirements of this project.

Maximum depth of 10 metres While the AUV has a mechanical depth constraint of 10 metres, there is no need to design sensors or other systems that is capable of operating at greater depths. Below a depth of 10 metres there is a risk of leakage.

Standing water As this is an initial design of an AUV, the factor of moving water is eliminated by testing it in a pool with standing water.

Type of water As a consequence of the AUV being tested in a swimming pool, the water type is of interest, since the density of saltwater and fresh water are different. The swimming pool is chloric water, but the amount of chloride is negligible, so the parameters of fresh water will be used for this system.

Power management As a consequence of this being an initial design, the AUV will be tested under controlled conditions, which are limited compared to the North Sea. Therefore the power management does not need to be optimized.

Radio navigation In the large scale implementation the platform should have a Global Positioning System (GPS) capability. There are some constraints of using a GPS on an AUV. The first thing is that the GPS device is not able to get a GPS fix under water, and it takes time to get a GPS fix when it surfaces. Therefore GPS will be omitted when tested in an indoor pool.

Communication As the AUV is an autonomous device which does as instructed, it will be convenient to have a sporadic communication link to

the operator. This means that it should have a communication link to the ground station. Since a wire is impractical when crossing the North Sea, a radio link will be used. As this initial platform should be tested in a swimming pool, the radio link can be any short range or even land based service available.

Requirements 2

In this chapter the requirements for the project will be specified and explained. The requirements are seperated into major requirements and minor requirements. Major requirements will be the primary focus during the development of this project. Section 2.2 on the next page lists how the requirements should be tested and the criterion for a successful acceptance testing.

2.1 Requirement specification

In these requirements it is predetermined that all the tests are preformed in a controlled environment, such as a swimmingpool.

Major requirements:

- *Requirement 1*: Must be able to manoeuvre autonomously along a predefined three-dimensional route, which can be defined by the AUV being able to do the following:
  a) Move for- or backward with ±1 m margins
  b) Turn left or right with a maximum turning radius of 1 m
  c) Move vertically up or down with ±1 m margins
  d) Move up or down, while moving forwards with ±1 m margins
  e) Maintain given position with ±1 m margins
  f) Reach way point within a radius of 1 m
- *Requirement 2*: Must measure internal and external parameters, including:
  a) Measure velocity with an accuracy of 10 %
  b) Measure depth with an accuracy of ±0.1 m
  c) Measure direction relative to the Earth's magnetic field (compass) with an accuracy of ±2.5°
  d) Measure battery voltage with an accuracy of ±0.01 V

Minor requirements:

- *Requirement 3*: Must have an emergency routine so the AUV surfaces if the battery level becomes critical.
- *Requirement 4*: Must be able to wirelessly send logfiles to a ground station and receive a new route within 100 metres line of sight.

- *Requirement 5*: Rise to the water surface to communicate with the ground station wirelessly to receive new waypoints.

The requirement specification is further specified in the chapter 3 on page 13 requirement analysis, where further description of the requirements as well as to what sensors to use if the end product is to comply with the requirements.

2.2 Acceptance testing

Through this section it is described how each requirement is tested and the expected result for a successful test. All the specificed tests below are conducted in a controlled environment, where there is no wind or current to take into account.

Test 1: Must be able to manoeuvre autonomously along a predefined three-dimensional route, which can be defined by the AUV being able to do the follwing:

While the AUV is submerged into water, the below specified tests are conducted seperatly, followed by a combined test where the tests below are joined in a sequence, and carried out one after the other.
Succes: The AUV carries out the below mentioned tests in a sequence.

Test 1a: Move for- or backward with ±1 m margins.

The AUV is submerged into water and programmed to sail 10 metres and stop.
Succes: The AUV moves along the line for 10 metres (withing ±1 m margin)

Test 1b: Turn left or right with a maximum turning radius of 1 m

The AUV is instructed to go go to a waypoint 10 metres away, and then to go back to its starting position to make it turn at the smallest radius possible.
Succes: The turning radius is less than 1 metre

Test 1c: Move vertically up or down with ±1 m margins

The AUV is submerged in water, and instructed to dive for 2 metres vertically and afterwards go up for 2 metres.
Succes: The AUV dives 2 metres, maintains the depth for 60 seconds (within ±1 m margin), and returns to the surface in the original position (within ±1 m margin).

Test 1d: Move up or down while moving forward with ±1 m margins

The AUV is submerged into water, and is instructed to move forward 2 metres while diving 2 metres.
Succes: The AUV ends up with a depth of 2 metres (within ±1 m margin) and has moved forward 2 metres (within ±1 m margin).

Test 1e: Maintain given position within ±1 m margins

The AUV is submerged into water, and is instructed to move to a way point. When destination is reached, the AUV maintains position in the water.
Succes: The AUV is able to maintain its position within ±1 m margins.

Test 1f: Reach way point within a radius of 1 m

The AUV is submerged into water, and is instructed to move to a waypoint. When destination is reached, the AUV should be within a radius of 1 m from the destination.
Succes: The AUV ends up within a radius of 1 metre from the destination.

Test 2: Must measure internal and external parameters, including:

The AUV is submerged into water, instructed to dive and sail forward for 5 metres and then surface.
Succes: It is possible to pass the following tests and read the data either from a log file, or through radio communication.

Test 2a: Measure velocity with an accuracy of 10 %

The AUV is submerged into water, instructed to dive and sail forward for 5 metres then surface.
Succes: A reading of the speed can be read either from a log or through radio communication with an accuracy of 10 %.

Test 2b: Measure depth with an accuracy of ±0.1 m

The AUV is submerged into water, instructed to dive and sail forward for 5 metres then surface.
Succes: A reading of the water pressure can be read either from a log through radio communication with an accuracy of ±0.1 m.

Test 2c: Measure direction relative to the Earth's magnetic field (compass) with an accuracy of $\pm 2.5^{\circ}$.

The AUV is submerged into water, instructed to dive and sail forward for 5 metres then surface.
Succes: A reading of the direction can be read either from a log or through radio communication with an accuracy of $\pm 2.5^{\circ}$.

Test 2d: Measure battery voltage with an accuracy of ± 0.01 V

The AUV is submerged into water, instructed to dive and sail forward for 5 metres then surface.
Succes: A reading of the battery voltage can be read either from a log or through radio communication with an accuracy of 0.01 V.

Test 3: Must have an emergency routine so the AUV surfaces if the battery level becomes critical.

The AUV is connected to a variable voltage supply, and the voltage is gradually decreased.
Succes: The AUV's onboard pump empties the ballast when the voltage are critical.

Test 4: Must be able to wirelessly send logfiles to a ground station and receive a new route within 100 metres line of sight.

The AUV is placed at water surface within 100 metres from the ground station. The software running on the ground station is set up to exchange data with the AUV.
Succes: A successful transfer without errors is completed three times.

Test 5: Rise to the water surface to communicate with the ground station wirelessly to receive new waypoints.

The AUV is programmed to sail forward for 5 metres, and wait for new instructions.
Succes: The new instructions make the AUV sail a new route which is transferred to it.

Sensor analysis 3

This chapter describes and discusses the different sensor types of whose some are needed for the AUV to work properly.

The hardware of this project is based on a remote controlled stock submersible, as seen on figure 3.1. The stock submersible already contains stock servos and control surfaces for controlling pitch and yaw, a DC-motor with a propeller, and a water ballast tank to adjust the buoyancy of the submarine.

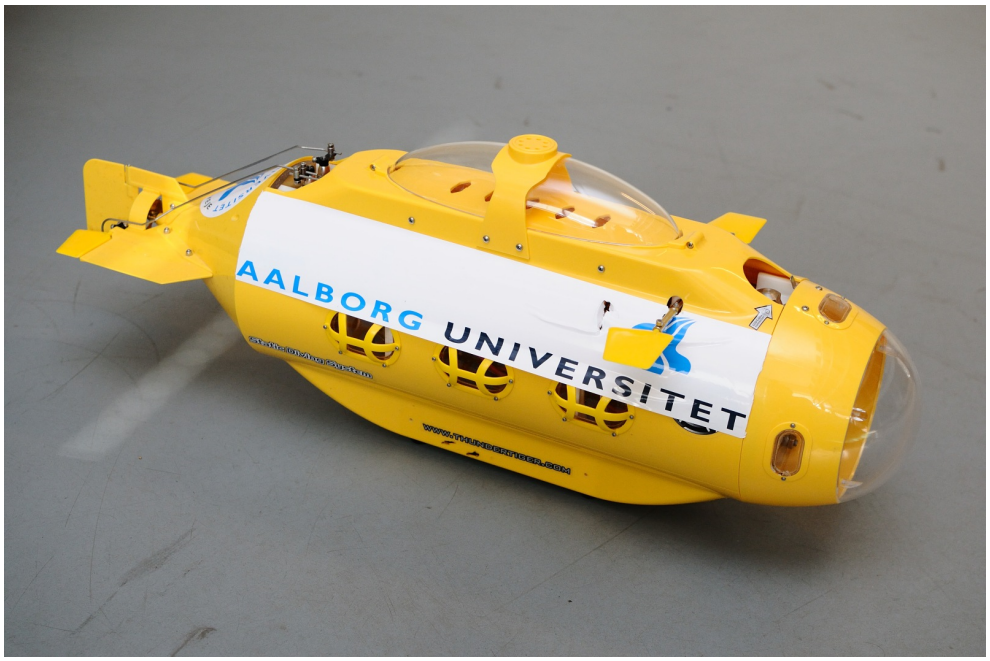


Figure 3.1: The stock submersible used in this project.

In this project the remote controller will be replaced with a microcontroller unit. The microcontroller unit is interfaced with sensors to sense the surrounding environment and actuators for reacting on the preprogrammed mission and the environment. In this section the sensors and actuators needed for certain functions are discussed.

The actuators needed are already present in the AUV, but to use them autonomously the AUV needs sensory equipment.

In the following, sensors will be associated with the requirements that they are to fulfil.

Requirement 1: Must be able to manoeuvre autonomously along a predefined three-dimensional route, which can be defined by the AUV being able to do the following:

Move for- or backward with ±1 m margins
This can be done by controlling the stock DC-motor in the AUV just with some regulation applied.

Turn left or right with a maximum turning radius of 1 m
Using the stock actuators on the AUV along with control of the rudders, this can be accomplished.

Move vertically up or down with ±1 m margins
Using the stock ballast tank aboard, the AUV is able to move up or down.

Move up or down, while moving forward with ±1 m margins
This can be done by using the stock DC-motor to surge ahead while using the stock ballast tank.

Maintain given position with ± 1 m margins
This is possible by using readouts of all the sensors and combining these with the actuators and the pump.

Reach way point within a radius of 1 m
All the above functions and requirements should be able to accomplish this.

Requirement 2: Must measure internal and external parameters, including:

Measure velocity
This can be accomplished in several ways – one could be to calculate it from the readouts of an accelerometer, and integrate over these. Another way would be to implement a water flux sensor to measure the water speed around the AUV. The flux sensor is relative to the water flowing around the AUV, which dictates that the water should be still, if an absolute velocity is to be calculated.

The accelerometer-method of measuring velocity compared to using a water flux sensor has a disadvantage; if an accelerometer is used to measure velocity, some challenges are to be taken into account. The most important challenge is that small errors may accumulate and result in a large error in the velocity. This can also result in static velocity readout that differs from zero.

If a water flux sensor is used, small errors can occur. If tested in a pool, whilst the AUV is maintaining a fixed position, the flux will be zero. It would be optimal to use a water flux sensor for this project, however it has not been possible to aquire such a device.

Measure depth

Depth measurement in water can be accomplished with a manometer. The requirements for this, is to be able to measure pressure down to a depth of 10 metres, as according to the limitations in section 1.5 on page 6. Furthermore the accuracy should be within ±0.1 m according to the margins from the requirement specification in chapter 2 on page 9.

As the depth can be calculated from the pressure, a requirement for the resolution can be calculated so a specific sensor can be chosen. First the maximum pressure will be calculated as the pressure rises with increasing depth. The pressure also depends on the density of the water, which again depends on the type of water. According to the limitations, the water is fresh water which have a density $\rho_{\text{water}} = 998.2$ kg/m^3, and the general equation to calculate the absolute pressure is, $p = \rho \cdot g \cdot h + p_{\text{surface}}$, so the absolute pressure at a depth of 10 metres is as seen in equation 3.1c.

$$p_{10\text{ m}} = \rho_{\text{water}} \cdot g \cdot h + p_{\text{surface}} \tag{3.1a}$$
$$= 998.2 \cdot 9.82 \cdot 10 + 101325 \tag{3.1b}$$
$$= 199348.24 \approx 199 \text{ kPa} = 1.99 \text{ bar} \tag{3.1c}$$

Where:

$p_{10\text{ m}}$ = the pressure at a depth of 10 meters
ρ_{water} = the density of the water
g = the gravity of the Earth
h = the depth
p_{surface} = the atmospheric pressure above the water

Next the minimum resolution has to be determined, as the accuracy should be within 10 cm, then the pressure difference for 10 cm has to be found.

$$\Delta p = \rho_{\text{water}} \cdot g \cdot \Delta h \tag{3.2a}$$
$$= 998.2 \cdot 9.82 \cdot 0.1 \tag{3.2b}$$
$$= 980.2324 \approx 980 \text{ Pa} \approx 0.01 \text{ bar} \tag{3.2c}$$

Where:

Δp = the pressure difference per 10 cm
ρ_{water} = the density of the water
g = the gravitational acceleration of the Earth
Δh = the difference depth of 10 cm

Therefore the sensor should be able to measure up to 2 bars of absolute pressure and have a resolution better than 0.01 bar.

Measure direction relative to the Earths magnetic field (compass)
This can be done with a magnetometer. To comply with the requirement specification in chapter 2 on page 9, the precision of the magnetometer, must be within $\pm 2.5°$, hence this then can tell the yaw of the AUV.

As an aid to help control of the AUV, a combination of a gyrometer and a compass could be used. This, however is not necessary due to the slow rate of turn in an underwater vehicle.

Measure battery voltage
This can be achieved by using a AD-converter on the microcontroller, and have the necessary resolution to give an idea of the battery voltage. Also to know when the battery voltage falls to a critical level.

Requirement 3: Must have an emergency routine so the AUV surfaces if the battery level becomes critical. The AUV must be able to measure the voltage of the battery. This can be done by reading the volt meter, and initiate the emergency procedure.

Requirement 4: Must be able to wirelessly send log files to a ground station and receive a new route within 100 metres line of sight. This can be done in several ways. As previously stated, the tests are conducted in a swimming pool, whereas maritime Very High Frequency (VHF) communication can be used. There are no greater advantages and disadvantages in either solution, therefore a VHF communication module is chosen. The implementation can be done with a module such as the APC220 `/datasheets/VHF/`.

Requirement 5: Rise to the water surface to communicate with the ground station wirelessly to receive new way points. The software must rise the AUV to the surface when no more waypoints are present, to receive a new route.

3.1 Summary

To make an AUV that can sail a predefined route through a swimming pool and comply with the requirement specification the following sensor types are needed.

- Fluxmeter
- Magnotometer
- Manometer

- VHF module

This have been determined from the analysis of the sensors in which calculations of the depth and requirements from the specification are taken into account.

4 Use cases

The main goal with this chapter is to analyze how the product will work best in practice. The analysis will focus on setting up the AUV unit before the actual dive as it will be developed as an autonomous device that requires no interaction at all.

This use case analysis is divided into several descriptions of the interactions between the AUV, sensors and actuators respectively or between the AUV and the wireless communication through which the user can setup the AUV. This has further been illustrated on figure 4.1.

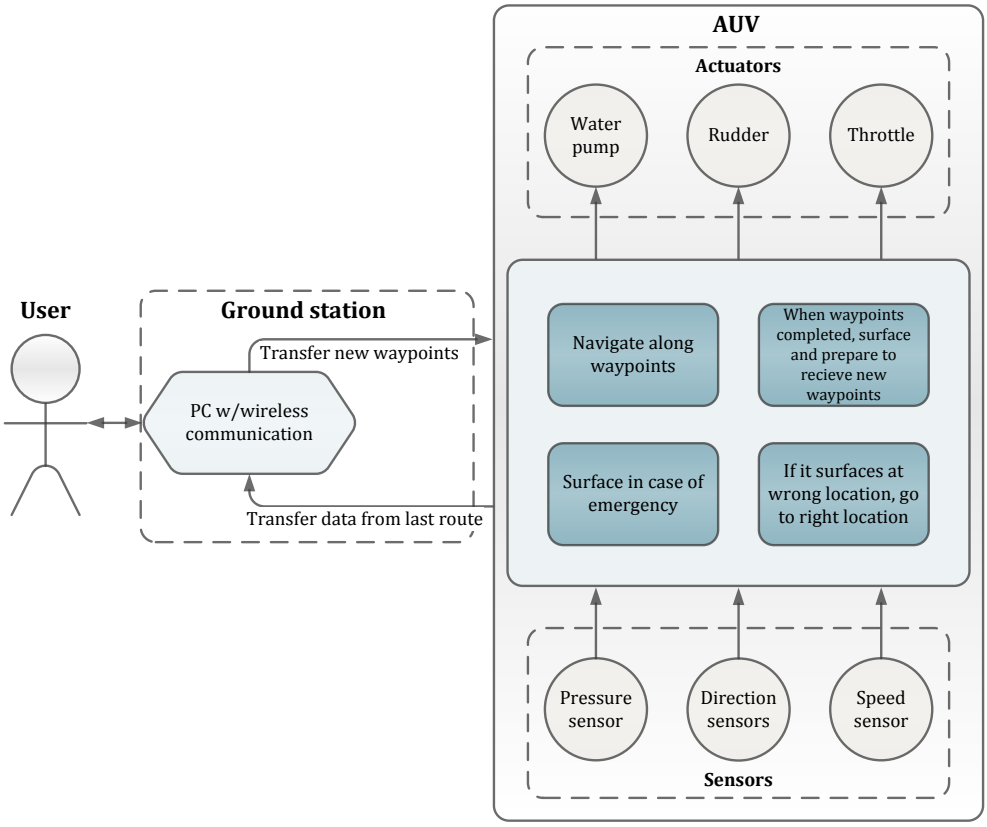


Figure 4.1: Overview over use cases

Use case 1: Transfer new way points to AUV

The purpose of this use case is to instruct the AUV with a new route. The route is setup as a series of waypoints using a user interface on a computer. The typical scenario is as follows:

1. User opens AUV software on a computer

2. User makes a three-dimensional route
3. User selects to transfer new way points
4. User saves the route so it will be transferred next time AUV is at the surface
5. When AUV surfaces, it sends an acknowledgement of receiving new way points

Use case 2: View recorded data from AUV

The purpose of this use case is to read out data from the AUVs last dive. The typical scenario is as follows:

1. User opens AUV software on a computer
2. User selects to read out data
3. User interface shows data from last dive and lets the user save them to a file

Use case 3: Unexpected error

The purpose of this use case is to send an error signal to the user right before the AUV experiences any kind of failure to the system. The typical scenario is as follows:

1. The AUV experiences some kind of error, such as the battery level falls to a critical level or the system gets a failure.
2. The AUV surfaces.
3. The AUV sends an error indicator to the PC so the user can read the error. By reading this error, the position of the AUV be stated.
4. User handels the error and can go fetch the AUV at the given position.

These three usecases shows the interaction between user and the AUV. The use cases can be used to specify how the software of the system should be running. The only thing the user can do is to request data from the AUV and send new routes to the AUV. Therefore it is necessary to make a Graphical User Interface (GUI) for the user, so the person can interact with the AUV when it is merged.

4.1 Interface between AUV and ground station

Throughout this section the interface between the AUV and a ground station will be described and it will be concluded what will be required for a communication link.

Requirements for data transmission

As mentioned in section 4 on page 19 it has been determined that the user should be able to do two-way communication with the AUV. It is well documented that wireless communication though water is extremely energy inefficient, all wireless communication will therefore take place at surface level. Because all communication takes place while the AUV is at surface level, the aim is to eliminate the need for continuous transmissions, and it has been determined that the transmissions will consist of:

- A new route for the AUV to travel
- Current location of the AUV
- A log of the already travelled distance
- Collection of data from the sensors

A new route for the AUV to travel consists of a series of way points sent from the ground station to the AUV. These way points can be as simple as a set of GPS coordinates, from the which the AUV can deduce how to reach the new destination depending on its current location. It is assumed that the amount of data from this data transmission would be limited to an amount that can be handled in burst transmissions rather than transmitting a continuous datastream.

Current location of the AUV is a special case of the previous case, where there will be transmitted only one set of three-dimensional coordinates from the AUV to the ground station. As before it is not necessary to transmit the current location on a continuous basis

A log of the already travelled distance is necessary to determine how the AUV have manoeuvred.

Collection of data from the sensors is the primary function of the AUV. A continuous stream of data from the AUV to a ground station is one way to go. This is however not desirable as it is only desired to transmit while surfaced. Therefore it is required that the data is sorted and stored.

Communication use cases

This section will describe the various use cases for communication between the AUV and the ground station. To use as little power as possible the AUV will in all use cases initiate communication whenever the AUV is at the surface. Whenever the ground station establishes contact with the AUV there is a data exchange which specifies the type of use cases to be executed. Furthermore all

data transmissions are ended by a End Of Data (EOD) code. On figure 4.2 is an overview of the different data transmissions illustrated.

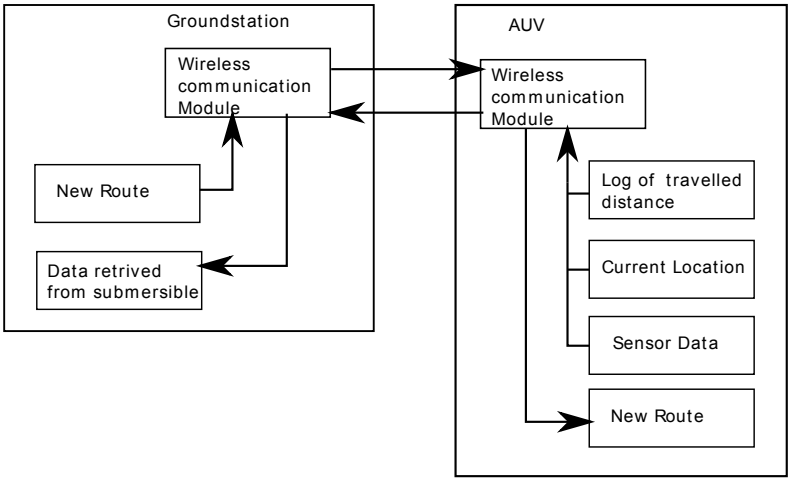


Figure 4.2: Overview of data transmissions.

Use case 1: AUV is out of range

This use case describes how the system should react in case the AUV is not in communication range either because it is gathering data, or otherwise unable to establish contact with the ground station. The typical scenario is as follows for ground station:

1. The ground station starts a timer to indicate how long the AUV has been out of radio range
2. As long as the ground station do not receive an operation code, this use case repeats
3. When the ground station and the AUV is within reach of one another the timer is reset, and any pending operations at the ground station continues as one of the following use cases.

Use case 2: Transmitting new route

The purpose with this use case is to update the AUV with a new route. The route consist of a series of way points, which consist of a set of coordinates in three dimensions. The typical scenario is as follows:

1. The AUV and ground station initiates communication
2. The ground station transmits operation code to AUV
3. The ground station transmits an acknowledgement for each packet

4. The AUV receives an acknowledgement within specified timeout
5. The ground station transmits data set followed by an EOD code
6. Communication protocol is terminated, and the AUV begins a new route

Use case 3: Data retrieval from AUV

This usecase describes the protocol for the ground station to acquire sensor- and/or location data from the AUV. The typical scenario is as follows:

1. The AUV and ground station initiates communication
2. The ground station transmits an operation code to AUV
3. The AUV begins to transmit the requested data followed by an EOD
4. The ground station transmits an acknowledgement of each packet

Use case 4: Unexpected error

This use case describes the protocol if the AUV have detected a critical error. Unlike the other cases the AUV will not wait for the ground station to make any acknowledgements before transmitting its location. The typical scenario is as follows:

1. The AUV rises to surface
2. The AUV transmits location data
3. The AUV waits for 10 seconds
4. The 2. and 3. items are repeated until the battery is discharged

Part II

Design

In this part the design choises made throughout the design process is described; includeing the hardware, modelling and software.

5 Hardware platform

This chapter describes the hardware platform used in this project. This includes sensors and the communication module.

To fulfil the needs for the AUV, a selection of different sensors and external communication are needed. As concluded in chapter 3 on page 13, the required sensors and external communication consists of:

- Fluxmeter
- Magnetometer
- Manometer
- VHF module

To collect and analyze the data from all the sensors and communication module, a microcontroller is needed. In the following sensors, a communication module and a microcontroller will be chosen and described.

5.1 ARM7 microcontroller

The most important requirement for the microcontroller is that it shall be able to communicate with the sensors. For this project the AT91SAM7A3 (later referenced as ARM7) is chosen. The benefits from this platform are low power consumption, easy to interface with different kinds of components and a selection of freely available operating systems is able to run on it.

In this project the ARM7 processor is mounted on a circuit board which also includes[1]:

- Two eight-channel 10-bit Analog to Digital Converter (ADC) controllers
- Serial Peripheral Interface (SPI) header
- Two Pulse Width Modulation (PWM) outputs
- Three 16-bit timer/counters
- Universal Asynchronous Receiver/Transmitter (UART) for communicating with a PC
- And other miscellaneous features

The ARM7 processor is running at 48 MHz clock speed and includes 256 kB of flash memory and 32 kB of SRAM onboard, which allows running a light operating system and to do real-time calculations and operations. [Atmel, 2006]

[1] /datasheets/SAM7A3-TNB.pdf

5.2 Sensors

Fluxmeter

Measuring the speed is done by measuring the waterflow around the AUV. This measurement can be accomplished by using several methods where two of the relevant methods are either a small propeller or ultrasonic sound waves.

Propeller fluxmeter

The method using a small propeller can be archieved by letting the propeller drive a device which measures the amount of rotations thus producing an output consisting of one or more pulses per rotation.

Ultrasonic fluxmeter

The method using ultrasonic sound waves will send ultrasonic waves from one transducer located at the front of the AUV while sampling signals received by a transducer located at the back of the AUV. The delay will vary proportional to the speed of the waterflow.

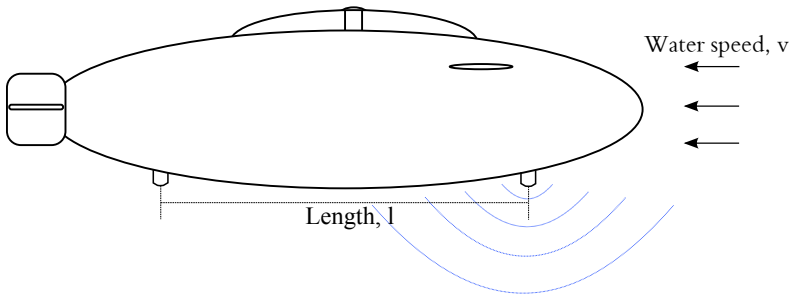


Figure 5.1: Measuring flux using ultrasonic transducers.

An example setup using this method is showed on figure 5.1. To convert the measured delay into a value for the current velocity, it is required to know the speed of sound at the current pressure and temperature of the water and the distance between the two sensors.

A method for calculating the speed of sound in water depending on the temperature and pressure is found in appendix I on page 127.

For velocity measuring the known speed of sound is used to calculate the unknown factor; the flux of the water. This calculation can be done by using equation 5.1.

$$t_s = \frac{l}{c + v} \tag{5.1}$$

Where:

t_s = the delay from the sender to the receiver
l = the distance between sensors
c = the speed of sound
v = the flux of the water

v can obtain both positive and negative values, depending on the direction. If solved for v, it looks like equation 5.2.

$$v = \frac{l}{t_s} - c \tag{5.2}$$

For instance, if the speed of sound is 1482.524 m/s, the distance between sensors is 20 cm and the delay of the ultrasonic pulse is 134.887 µs, the following result is calculated:

$$v = \frac{0.2}{134.887\text{E-}6} - 1482.524 = 0.2 \text{ m/s} \tag{5.3}$$

If the delay of the ultrasonic pulse is 134.905 µs, the result is:

$$v = \frac{0.2}{134.905\text{E-}6} - 1482.524 = 0 \text{ m/s} \tag{5.4}$$

So to measure a velocity difference of 0.2 m/s, it is required to measure a time delay of:

$$134.905\text{E-}6 - 134.887\text{E-}6 = 18 \text{ ns} \tag{5.5}$$

The above states that, to archieve an accuracy of 0.2 m/s, a time delay of 18 ns has to be detected precisely. Because of the difficulties in measuring such small time delays, the idea of measuring flux speed using ultrasonic sound will not be used in this project.

Magnetometer, accelerometer and gyrometer

For measuring the absolute direction a magnetometer is used to measure the Earth's magnetic field. The output from a tri-axis magnetometer is a vector pointing towards the Earth's magnetic north pole. As inertial sensors a tri-axis accelerometer and tri-axis gyrometer is needed for inertial navigation.

For this project a suitable sensor is found. The sensor is an Analog Devices ADIS16405 and features: [Analog Devices, 2009]

- Digital tri-axis gyroscope with digital range scaling
- Digital tri-axis accelerometer
- Digital tri-axis magnetometer

The sensor communicates using the SPI protocol and uses a 5 V power supply. This device is later referenced as the IMU. Further information on this is described in section 8.5 on page 70.

Manometer

The manometer is used to measure the pressure outside the AUV to determine the depth.

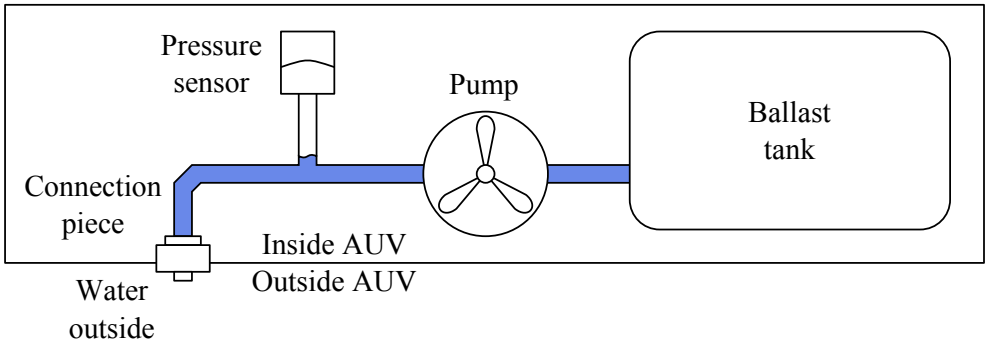


Figure 5.2: Schematic of the buoyancy system. It illustrates the connections of the pump, manometer, ballast tank and connection piece.

To give another input to determine the depth of the AUV, instead of only relying on the IMU, the manometer has an analog output, which is fed into the IMU using the 12-bit ADC that can be read out together with the rest of the IMU data.

5.3 Communication

It has been chosen to use an APC220 radio communication module for the communication between the AUV and the ground station. In this section the APC220 radio communication module will be described in more details. Shown in table 5.1 on the facing page are the technical specifications for an APC220 radio module.

As it can be seen the APC220 operates in the 418–455 MHz band, which is a band used for public wireless applications, meaning that no license is required. The APC220 is equipped with a built-in error correction protocol, which makes the APC220 a good communication platform for this purpose.

Work frequency	418–455 MHz
Modulation	GFSK
Transmitted power	20 mW
Received sensitivity	-113dBm at 9600 bps
Air rate	2400–19200 bps
UART rate	1200–57600 bps
Buffer size	256 bytes
Supply voltage	3.5-5.5V
Receiving current	$\leq$28 mA
Sleeping current	$\leq$5 µA
Transfer distance	1000 m (open space)
Dimension	37.5 mm x 18.3 mm x 7.0 mm

Table 5.1: Technical specifications of the APC220 radio module [Appcon Technologies, 2008]

Kinematics and dynamics 6

In this chapter the dynamic model of the system is described. The dynamic model is a mathematical model of the physical AUV.

To make a model for the AUV, from which to design a control system, the system is divided into smaller subsystems, which is then modelled and put together. In the requirement specification it is specified, that the AUV has to be able to do one two-dimensional manoeuver and two one-dimensional ones:

- Forward and backward motion is 1-dimensional
- Up and downward motion is 1-dimensional
- Turning is 2-dimensional

To represent the AUV different nomenclature is used as two reference frames exist. As seen on 6.1 the two systems are represented, these are used to map the different forces acting on the AUV and to develop a control system.

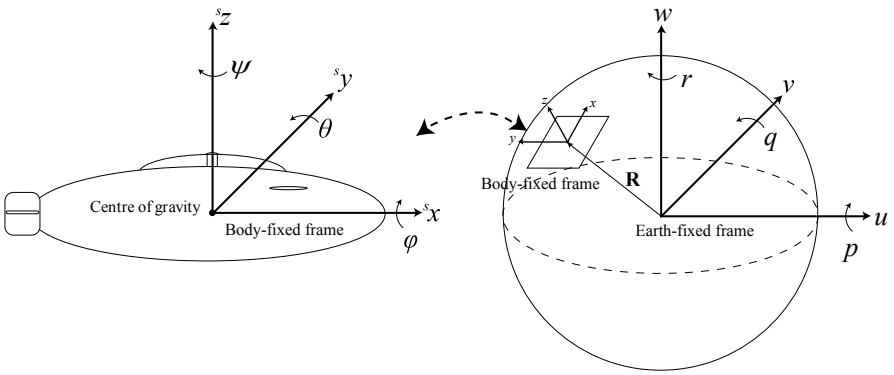


Figure 6.1: Reference frame used to describe the position of the AUV.

The earth-fixed frame can be represented by two vectors: $\mathbf{v}_1$, $\mathbf{v}_2$ that describes the vector as a point, and as angles around the axes. A notation for the Earth-fixed frame is denoted with a superscribed "e" (Earth) in front of the vector, and the vectors used in the body-fixed frame is superscribed with an "s" (submersible).

In table 6.1 on the following page, $\mathbf{v}_1$ and $\mathbf{v}_2$ represents the position in the Earth-fixed frame as a position and angles respectively. $\mathbf{n}_1$ and $\mathbf{n}_2$ represents the position in the body-fixed frame, again as a position and angles respectively.

On figure 6.1 the two systems are represented next to each other to give an overview of the frames. The body-fixed frame is always placed relative to the Earth-fixed frame, as it is the Earth-fixed frame that tells where on the Earth the AUV is, and what bearings the AUV is having. The ${}^{e}\mathbf{O}_{0,0,0}$ is placed in

Description	Name	$\mathbf{v}_1$, $\mathbf{v}_2$	$\mathbf{n}_1$, $\mathbf{n}_2$
Motion in the x-direction	surge	u	x
Motion in the y-direction	sway	v	y
Motion in the z-direction	heave	w	z
Rotation about the x-axis	roll	p	ϕ
Rotation about the y-axis	pitch	q	θ
Rotation about the z-axis	yaw	r	ψ

Table 6.1: The nomenclature used for the reference frames.

the centre of the Earth, and the v-axis cuts through Greenwich. This makes way for the longitude/latitude system used in modern navigation.

The body-fixed frame is placed on the Earth-fixed frame. This is used to calculate the forces acting on the AUV and later used to develop a model. Mapping between the two frames are important, as the sensors read locally, but the position is transferred to the Earth-fixed system, to for instance measure how far the AUV has moved. A mapping matrix has to be set up, as this will be used to see how movement in one frame corresponds to movement in the other. For this purpose a rotational matrix can be computed. This is used to map points from one coordinate system to the other. The rotation matrix must fulfil:

$$n_1 = \mathbf{R}_\mathrm{I} \cdot \dot{\mathbf{v}}_1 \tag{6.1}$$

Where $\mathbf{R}_\mathrm{I}$ is a rotation matrix used to describe the rotation from the inertial Earth-fixed frame to the body fixed. The other unknown in the equation is $\dot{\mathbf{v}}_1$, which is the time derivative of the Earth-fixed frame used to describe the change in time of the Earth-fixed frame. As the AUV only turns about the z-axis, a rotation matrix that only describes the turning about this can be used. Such one is taken from [Diebel], and states that the rotation about the z-axis is given by:

$$\mathbf{R}_\mathrm{I}(\psi) = \begin{bmatrix} \cos(\psi) & \sin(\psi) & 0 \\ -\sin(\psi) & \cos(\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{6.2}$$

The matrix $\mathbf{R}_\mathrm{I}(\psi)$ represents the rotation about the z-axis, and is used to map from the body-fixed frame to the Earth-fixed frame.

6.1 Modelling limitations

When designing the different controllers for the AUV there are several things that are taken into consideration. To simplify the development of the controllers some physical constants are left out as less variables have to be taken into account.

When the AUV is sailing forward, several things are influencing it. One of them is the Coriolis effect, but as the prototype AUV is only used for sailing in pools, and therefore not navigating the seas by it self—this can be omitted.

Another thing to be omitted is the salinity and the density of the water at which the AUV sails. This is also due to the fact that the AUV is sailing in a confined pool where these variables are known.

A lot of other characteristics are also omitted, for instance cavitation generated by the AUV's propeller when it moves through the water, and the amount of pressure generated by the propeller on the rudder, as it only rotates in one direction, thus increasing the pressure on one side and decreasing it on the other side, which in principle makes the AUV turn slightly to one side whilst sailing forward. This can of course be controlled by the rudder controller—but if taken care of beforehand, the computing time can be used elsewhere.

A more important fact, other than variables that are kept constant, is the linearizations and the way the drag is changing as the AUV is turning. The linearizations are used to make the non-linear systems linear. The controller is then designed at these points, and they are used for further calculations. The linearizations are used when the revolutions and the velocity is described and given as $V_0 = 0.5$ m/s and $n_0 = 50$ rps

The last thing left to describe is the way the AUV turns, which influences the drag that is added to this equation. This is throughout this chapter, described as a linear function—changing as the angle of the AUV changes. To make this linear approximation the frontal area of the AUV is measured when pointing straight ahead, and when turned 35° and from this, a formula for the surface area can be calculated and used to determine the drag.

6.2 For- and backward motion

To make a model for the forward and backward motion of the AUV the forces acting on it are taken into account. The only forward force acting on the AUV is the propeller, which is generating an amount of thrust and the only force acting in the opposite direction, is the drag generated by the AUV moving through the water. This model can also be used to measure the speed of the

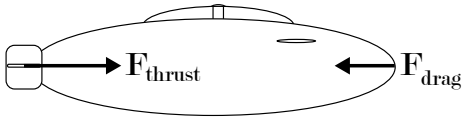


Figure 6.2: The forces acting on the AUV.

AUV as well as the distance covered by it. This is possible because of Newtons second law, stating that the force acting in any direction, is equal to the mass

times the acceleration, direction wise.

$$ {}^{s}\mathbf{F}_{\text{forward}} = \begin{bmatrix} \mathrm{F_x} \\ \mathrm{F_y} \\ \mathrm{F_z} \end{bmatrix} = \begin{bmatrix} m_{\text{auv}} \cdot a_{\mathrm{x}} \\ m_{\text{auv}} \cdot a_{\mathrm{y}} \\ m_{\text{auv}} \cdot a_{\mathrm{z}} \end{bmatrix} \tag{6.3} $$

Where:

${}^{s}\mathbf{F}$ = the total force acting on the AUV in the body-fixed frame.
$\mathrm{F_{x,y,z}}$ = the force acting on the object in a given direction.
m_{auv} = the total mass of the AUV.
$a_{\mathrm{x,y,z}}$ = the acceleration of the object in a given direction.

To simplify the forward and backward motion it is considered to be one dimensional, so the y- and z-composant are eliminated leaving a vector for forward and backward motion as follows:

$$ {}^{s}\mathbf{F}_{\text{forward}} = \begin{bmatrix} \mathrm{F_x} \\ \mathrm{F_y} \\ \mathrm{F_z} \end{bmatrix} = \begin{bmatrix} m_{\text{auv}} \cdot a_{\mathrm{x}} \\ 0 \\ 0 \end{bmatrix} \tag{6.4} $$

Moving through water is not the same as moving through air, and even at low velocities, the force acting against the AUV is to be taken into account, as the shape of the AUV is not very hydrodynamic. Therefore the total force along the x-axis is a sum of the thrust provided by the propeller and the drag created by the AUV as it moves through the water from where the following can be stated:

$$ {}^{s}\mathbf{F}_{\text{forward}} = \begin{bmatrix} \sum \mathrm{F_x} \\ \sum \mathrm{F_y} \\ \sum \mathrm{F_z} \end{bmatrix} = \begin{bmatrix} \mathrm{F_{thrust\text{-}x}} - \mathrm{F_{drag\text{-}x}} \\ 0 \\ 0 \end{bmatrix} \tag{6.5} $$

The formula for the thrust of the propeller can be given as in [Soerensen, 2005], that states the following:

$$ \mathrm{F_{thrust\text{-}x}} = \rho_{\mathrm{w}} \cdot D^4 \cdot \mathrm{K_t} \cdot |n| \cdot \ n \tag{6.6} $$

Where:

ρ_{w} = the temperature dependent density constant of water
D = the diameter of the propeller
$\mathrm{K_t}$ = a propeller constant
n = the number of revolutions per second

The constant $\mathrm{K_t}$ is given as a summation of several variables and constants as stated by [Oosterveld and van Oossanen, 1975]. The formula holds for the

general Wagenigen Type B propellers, and it is assumed that this holds for the stock propeller as well.

$$\mathrm{K_t} = \sum_{k=1}^{39} C_\mathrm{k} \cdot (J)^{S_\mathrm{k}} \cdot \left(\frac{P}{D}\right)^{t_\mathrm{k}} \cdot \left(\frac{A_\mathrm{E}}{A_\mathrm{O}}\right)^{u_\mathrm{k}} \cdot Z^{v_\mathrm{k}} \tag{6.7}$$

Where:

C_k = a constant found in the table in appendix H.2 on page 125
J = the advance ratio of the propeller given as: $\frac{V_a}{n \cdot D}$
S_k = a constant found in the table in appendix H.2 on page 125
$\frac{P}{D}$ = the pitch ratio of the propeller
t_k = a constant found in the table in appendix H.2 on page 125
$\frac{A_\mathrm{E}}{A_\mathrm{O}}$ = the expanded area ratio, calculations can be found in appendix
u_k = a constant found in the table in appendix H.2 on page 125
Z = the number of blades on the propeller
v_k = a constant found in the table in appendix H.2 on page 125

When the constants from appendix H.2 on page 125 is used, the expression reduces to:

$$\mathrm{K_t}(V_\mathrm{x}, n) = -\frac{10.5027 \cdot V_\mathrm{x}}{n} - \frac{141.4065 \cdot V_\mathrm{x}^2}{n^2} - \frac{1297.6154 \cdot V_\mathrm{x}^3}{n^3} + 5.3176 \tag{6.8}$$

From 6.8 it is seen that the thrust coefficient is dependent on the velocity (V_x) and the revolutions of the propeller (n). An approximation of this equation would state that the number of revolutions is quite higher than the velocity, thus eliminating the terms, and giving a constant for the thrust. For revolutions higher than 1 the function is estimated to give a constant, why the $\mathrm{K_t}(V_\mathrm{x}, n)$ only goes for $n < 1$, otherwise the approximation $\mathrm{K_t}(V_\mathrm{x}, n) \approx 5.3176$ holds. The approximated formula for the forward thrust can therefore be written as:

$${}^{s}\mathrm{F}_{\text{thrust-x}}(n) \approx \rho_\mathrm{w} \cdot D^4 \cdot 5.3176 \cdot |n| \cdot \ n \tag{6.9}$$

The force acting in the opposite direction of the AUV is the drag. This is given as:

$${}^{s}\mathrm{F}_{\text{drag-x}} = \frac{\mathrm{C_d} \cdot \rho_\mathrm{w} \cdot \mathrm{V_x^2} \cdot A}{2} \tag{6.10}$$

Where:

$\mathrm{C_d} = 0.82$ = is a drag coefficient found on [Engineering Toolbox, 2011]
ρ_w = the temperature dependent density of water
$\mathrm{V_x}$ = the velocity of the AUV along the x-axis
A = the cross sectional diameter of the AUV

If the formula for the thrust and the formula for the drag can be used to express the motion along the x-axis, these can be put into the vector notation used earlier:

$$ {}^{s}\mathbf{F}_{\text{forward}}(V_{\text{x}}, n) = \begin{bmatrix} \sum {}^{s}\text{F}_{\text{x}}(V_{\text{x}}, n) \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} {}^{s}\text{F}_{\text{thrust-x}}(V_{\text{x}}, n) - {}^{s}\text{F}_{\text{drag-x}} \\ 0 \\ 0 \end{bmatrix} \tag{6.11} $$

And when Newtons law state that:

$$ m_{\text{auv}} \cdot {}^{s}a_{\text{x}} = \sum {}^{s}\mathbf{F}_{\text{forward}}(V_{\text{x}}, n) \tag{6.12} $$

The acceleration can be given as (where acceleration is a derivative of the velocity, and the velocity is a derivative of the position) the formula for the movement approximates:

$$ \ddot{P}_{\text{x}} = \dot{V}_{\text{x}} = a_{\text{x}} = \frac{\sum {}^{s}\mathbf{F}_{\text{forward}}(V_{\text{x}}, n)}{m_{\text{auv}}} \tag{6.13} $$

Which can then be put into a state space equation:

$$ \begin{bmatrix} {}^{s}\dot{V}_{\text{x}} \\ {}^{s}P_{\text{x}} \end{bmatrix} = \begin{bmatrix} -\frac{\partial}{\partial \cdot V} {}^{s}V_{\text{x}} & 0 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} V \\ P \end{bmatrix} + \begin{bmatrix} \frac{\sum \text{F}_{\text{x}}}{m_{\text{auv}}} \\ 0 \end{bmatrix} \tag{6.14} $$

The calculations on the forces can be used in a model of the AUV, which can later be used as a model for the modelling of the control systems used to regulate the AUV. If inserted into Simulink, it can be modeled as seen on figure 6.13 on page 50. Where the input of the system is the thrust generated by the AUV. This gives the graphs as seen on 6.3 on the facing page.

6.3 Up- and downward motion

> *"Any object, wholly or partially immersed in a fluid, is buoyed up by a force equal to the weight of the fluid displaced by the object."*
> [Archimedes of Syracuse, 212 B.C.]

From this the previous statement must be fulfilled for the AUV to be in a state of equilibrium (i.e. not going up or down), the following figure depicts the forces on the AUV, when seen in an equilibrium:
From Archimedes and the model, a formula for the forces on the AUV can be set up, and when the two forces are in equilibrium (the AUV does not move on the z-axis) the following holds:

$$ {}^{s}\mathbf{F}_{\text{down}} = {}^{s}\mathbf{F}_{\text{up}} \tag{6.15} $$

Where:

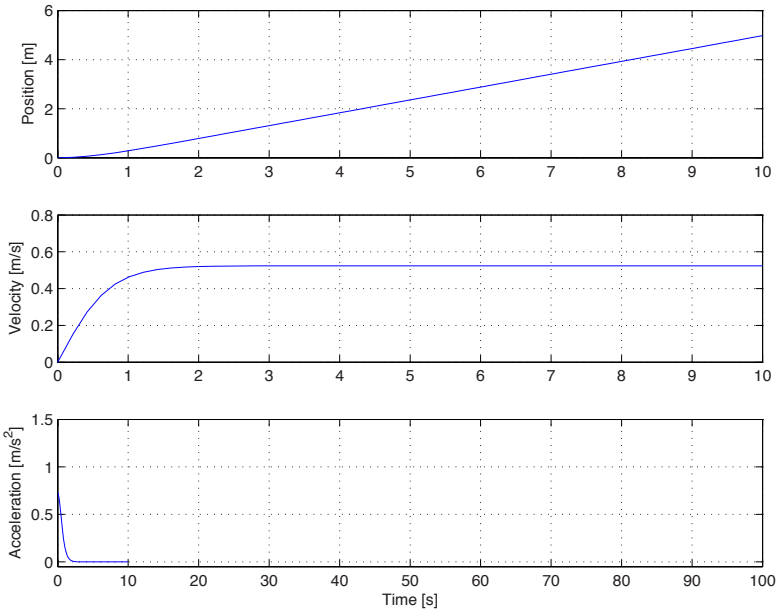


Figure 6.3: The simulation of the AUV moving through the water.

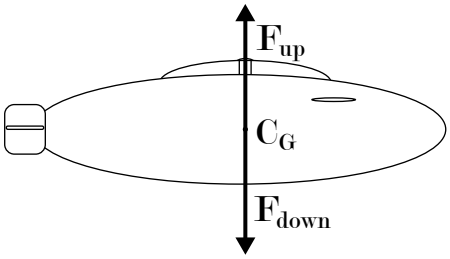


Figure 6.4: The forces acting on the AUV.

${}^{s}\mathbf{F}_{\text{down}}$ = the force acting down on the AUV
${}^{s}\mathbf{F}_{\text{up}}$ = the force acting up on the AUV

The two forces are equal to one another, if the AUV is still in the water, if written as a vector, the force acting down is written as:

$$ {}^{s}\mathbf{F}_{\text{down}} = \begin{bmatrix} \sum \mathrm{F_x} \\ \sum \mathrm{F_y} \\ \sum \mathrm{F_z} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ m_{\text{auv}} \cdot g - \mathrm{F}_{\text{drag-z}} \end{bmatrix} \tag{6.16} $$

The forces bouyancing the AUV upwards, can be written as:

$$ {}^{s}\mathbf{F}_{\text{up}} = \begin{bmatrix} \sum \mathrm{F_x} \\ \sum \mathrm{F_y} \\ \sum \mathrm{F_z} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \rho_w \cdot \mathrm{Vol}_{\text{auv}} \cdot g - \mathrm{F}_{\text{drag-z}} \end{bmatrix} \tag{6.17} $$

The drag still acts in both directions, when the AUV is going up or down, and in this case the only thing changed from 6.10 on page 37 is the axis it is used on (in this case the z-axis) and the drag coefficient, which is alot larger for the AUV when seen from the top or bottom.

$$\mathrm{F}_{\text{drag-z}}(V_\mathrm{z}) = \frac{\mathrm{C_d} \cdot \rho_\mathrm{w} \cdot V_\mathrm{z}^2 \cdot A}{2} \tag{6.18}$$

Where:

$\mathrm{C_d}$ = the drag coefficient
ρ_w = the density of the water
V_z = the velocity of the AUV
A = the cross-sectional area of the AUV

As the drag force is dependent on the velocity of the AUV the total force acting, is therefore also dependent on the velocity. As Newtons law states the total force (dependent on the velocity) can be given by the mass times the acceleration.

$$m_\text{auv} \cdot {}^s a_\mathrm{x} = \sum {}^\mathbf{s}\mathbf{F}_\mathrm{z}(V_\mathrm{z}) \tag{6.19}$$

As acceleration is a derivative of the speed, which is a derivative of position, the following can be stated:

$$a_z = \frac{\sum \mathbf{F}_\mathrm{z}(V_\mathrm{z})}{m_\text{auv}} = \dot{V_\mathrm{z}} = \ddot{P_\mathrm{z}} \tag{6.20}$$

Where:

a_z = the acceleration along the z-axis
V_z = the velocity on the z-axis
P_z = the position on the z-axis

As ${}^s\mathbf{F}_\text{up}$ is constant and can not be dealt with, which determines the AUV's bouyancy is the mass that is controlled by the onboard ballast tank, which can be filled/emptied with water according to the wanted depth. The total force acting on the AUV (being either positive or negative, e.g. the AUV is moving up or down) can be given as seen in 6.21.

$$\sum {}^\mathbf{s}\mathbf{F}_\mathrm{z} = \mathbf{F}_\text{down} - \mathbf{F}_\text{up} \tag{6.21}$$

As Newtons law states, the acceleration can be expressed as a derivative of the velocity, which in turn is a derivative of the position, from where the following state space equation can be set up.

$$\begin{bmatrix} {}^s\dot{V}_\mathrm{z} \\ {}^sP_\mathrm{z} \end{bmatrix} = \begin{bmatrix} -K & 0 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} V_\mathrm{z} \\ P_\mathrm{z} \end{bmatrix} + \begin{bmatrix} \frac{\sum \mathrm{F_z}}{m_\text{auv}} \\ 0 \end{bmatrix} \tag{6.22}$$

6.4 Turning motion

To make a sufficient model for turning the AUV, several things must be taken into account. For example the drag coefficient which changes with the rate of turn, as it changes shape from a sphere (seen from the front) to an ellipsoid (seen from the side). This influences on the turn rate as a larger resistance and decreases the rate of turn for the AUV.

The initial condition is one where the AUV is sailing forward with a constant velocity. In this case it is assumed that the rudder is at an angle of zero degrees. If the rudder begins turning to one side, forces all of the sudden begins to act on the AUV. To depict the forces acting on the AUV in case of a turn. The figure 6.5 illustrates these forces and is used as a reference throughout this section.

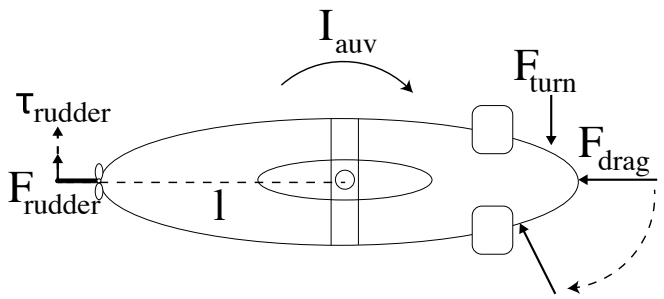


Figure 6.5: The forces acting on the AUV while turning.

Several forces come into play and the corresponding formula for turning are two-dimensional rather than the one-dimensional ones used in the previous models. This is due to turning being a two-dimensional movement, as the forces are acting in two directions (x and y). To simplify it, the matrix used in this case, is given as angles, as the rotation will be a motion about the z-axis (also denoted yaw). The following list describes the different forces used in figure 6.5.

- I_{auv} = the inertial moment generated in the AUV.
- $\mathbf{F}_{\text{rudder}}$ = the force generated by the rudder.
- τ_{rudder} = the torque generated by the rudder.
- $\mathbf{F}_{\text{drag}}$ = the force generated by the AUV whilst moving through the water.

To determine the rate of turn of the AUV, an expression for the angular acceleration is used, as this can be integrated which gives the angular velocity, as well as the angle referenced to its initial reference. If the system is converted to a spherical coordinate system, the process of turning is simplified to a rotation about the z-axis. If this is done, the following statement is true for

the AUV whilst turning:

$$ {}^{s}\mathbf{F}_{\text{turn}} = \begin{bmatrix} {}^{s}\mathrm{F}_{\phi} \\ {}^{s}\mathrm{F}_{\theta} \\ {}^{s}\mathrm{F}_{\psi} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \sum {}^{s}\mathrm{F}_{\psi} \end{bmatrix} \tag{6.23} $$

This expresses the process of turning as a sum of all forces acting on the AUV in the process of turning, however the drag is not taken into account. More about this can be read in the next subsection, further calculations will also be present in this subsection. However, the process of turning is best described as a change in angle according to the position of the rudder. Therefore, a force of the rudder needs to be calculated, which can be done as described by Baker and Bottomly in [Rawson and Tupper, 2001], and gives an expression for the force generated by the rudder according to its angle:

$$ {}^{s}\mathbf{F}_{\text{rudder-y}} = \begin{bmatrix} \mathrm{F}_{\mathrm{x}} \\ \mathrm{F}_{\mathrm{y}} \\ \mathrm{F}_{\mathrm{z}} \end{bmatrix} = \begin{bmatrix} 0 \\ \mathrm{C} \cdot A_{\mathrm{R}} \cdot V^2 \cdot \delta_{\mathrm{R}} \\ 0 \end{bmatrix} \tag{6.24} $$

Where:

C = a rudder constant, [Rawson and Tupper, 2001] uses 18.
A_{R} = the area of the rudder.
V = the velocity of the water passing the rudder
δ_{R} = the angle of the rudder

As all the constants can be measured or calculated, then the only unknown remaining is the angle of the rudder δ_{R}. Baker and Bottomly also proposed to multiply the velocity by 1.3 as the rudder is placed behind the propeller, thus increasing the velocity of the water passing the rudder. This seems reasonable, and is also taken into account. To get from linear forces to angular ones, conversions between torques and intertial moments are used. The torque can be given as:

$$ \tau_{\text{rudder}} = l \cdot {}^{s}\mathbf{F}_{\text{rudder-y}} \tag{6.25} $$

Where:

τ_{rudder} = the torque generated by the rudder.
l = the distance from the centre of gravity.

From this follows that the torque is dependent on the angle of the rudder. To determine the angular velocity, the equation for the inertia is used:

$$ \tau_{\text{rudder}} = \alpha_{\text{auv}} \cdot I_{\text{auv}} \implies \alpha_{\text{auv}} = \frac{\tau_{\text{rudder}}}{I_{\text{auv}}} \tag{6.26} $$

Where:

α_{auv} = the angular acceleration.
I_{auv} = is the inertia of the AUV.

Through this, it is possible to calculate the angle of the AUV by integrating the angular velocity α_{auv}. The inertia for a rotating body can for the AUV be simplified to a cylinder rotating about a centre point (the centre of gravity). The simplified formula for rotating cylinders is given by:

$$I_{\text{auv}} = \frac{m_{\text{auv}} \cdot l^2}{12} \tag{6.27}$$

Where:

m_{auv} = the mass of the AUV.
l = the distance from the centre of gravity.

When all of these are taken into account, an expression for the force acting on the AUV as it turns can be written as:

$$\psi_{\text{auv}} = \begin{bmatrix} {}^{s}\text{F}_{\phi} \\ {}^{s}\text{F}_{\theta} \\ {}^{s}\text{F}_{\psi} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \iint \frac{18 \cdot A_{\text{R}} \cdot (1.3 \cdot V^2) \cdot \delta_{\text{R}} \cdot l}{\frac{1}{12} \cdot m_{\text{auv}} \cdot l^2} \end{bmatrix} \tag{6.28}$$

The above equation only takes into account the positive force moving the AUV forward, while approximating that the drag changes linearly as described in the following subsection.

Drag of turning bodies

The drag is changing in relation to the angle of the AUV. This can be expressed either as the precise drag coefficient or as an approximation. Due to lack of measuring equipment, an approximation has to be used for the calculations. As the AUV consists of a cylinder with a half-sphere as tip, the shape can be approximated to (also when seen from the side) a sphere. From a table lookup this approximates the drag coefficient to 0.82.

The only thing that is unknown to the AUV is the surface area, upon which this force will act. The surface area of the AUV is measurable, and a maximum angle of turning can be determined. A function for the surface area can be calculated, and this can then be inserted in the formula, and makes the drag dependent on the angle the AUV is turning, with regards to its initial bearing.

The total area measured prior to the turning is approximately measured to be $17 \cdot 22$ cm^2. If the AUV is turned 35 degrees to one side, the area is measured to be $41 \cdot 22$ cm^2. Hence it is only one direction that changes, a function can

be calculated, assuming a linear approximation holds. Through calculations, the function is given as:

$$f(\delta_{\text{rudder}}) = 0.1408 \cdot \delta_{\text{rudder}} + 0.374 \tag{6.29}$$

Which gives an estimate of the area as the AUV is turning. The only thing changing for the AUV whilst turning, is the drag, that is now dependent on the angle of the rudder, the speed of the AUV and the angle of the AUV.

6.5 Modelling of the DC-motor with propeller

In order to determine the velocity of the AUV, the speed of the propeller has to be known, and therefore the surge motor has to be modelled. The DC-motor used in this project is a standard DC-motor with a permanent magnet as stator, described in detail in [Charles M. Close, 2002, p. 347–352]. A DC-motor can be decomposed into an electrical and a mechanical part. The electrical part is a model of how the current is transformed into a magnetic torque driving the mechanical part. The mechanical part is a model describing the transformation from the magnetic torque into a angular velocity on the shaft of the DC-motor and into a forward thrust of the AUV. From this a transfer function is found and a regulator can be designed.

Electrical part

An electric model can be constructed as in figure 6.6. It is assumed that the motor is linear and free of loss. The model is described by a resistor R, inductor L and a generator e_m.

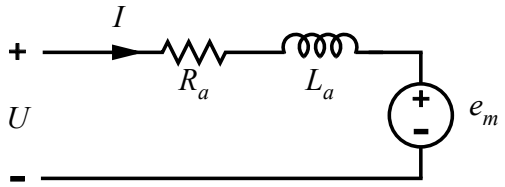


Figure 6.6: A model of the electric part of a DC-motor system.

The resistor R_a is a pair of commutator brushes communicating voltage and current to the armature windings L_a on the rotor. The voltage induced by the inductor is represented by a generator e_m. From the model in figure 6.6 the equation 6.30 is deducted in order to determine the voltage U(t) induced in the armature.

$$U(t) = R_\text{a} \cdot I(t) + L_\text{a}\frac{dI(t)}{dt} + e_\text{m} \tag{6.30}$$

From [Charles M. Close, 2002, p. 348–349] the expression 6.31 is obtained.

$$e_{\mathrm{m}} = K \cdot \omega(t) \tag{6.31}$$

Where:

K = a constant, depending on the physical aspects of the DC-motor
ω = the angular velocity of the rotor

Replacing e_m in equation 6.30 on the preceding page the final expression for the electric part is found as in equation 6.32.

$$U(t) = R_{\mathrm{a}} \cdot I(t) + L_{\mathrm{a}} \cdot \frac{dI(t)}{dt} + K \cdot \omega(t) \tag{6.32}$$

Mechanical modelling

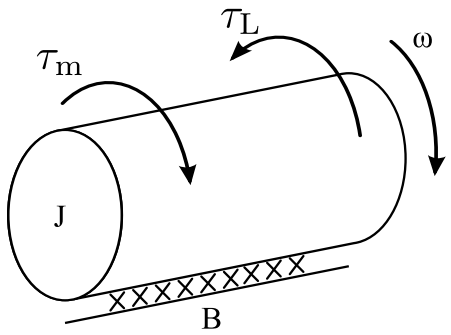


Figure 6.7: A model of mechanical part of a DC-motor system.

The mechanical part of the DC-motor can be described as in figure 6.7. As with the electrical part the DC-motor is considered linear, disregarding Coulomb friction and the stiction friction. Therefore the equation 6.33 is found where the driving torque has to overcome a load torque and a viscosity friction.

$$J\frac{d\omega(t)}{dt} = \tau_{\mathrm{m}} - \tau_{\mathrm{B}} - \tau_{\mathrm{l}} \tag{6.33}$$

Where:

τ_{m}=driving torque
τ_{B}=viscous friction
τ_{l}=propeller load torque

The load imposed by τ_l the propeller can be described by 6.34, see [Oosterveld and van Oossanen, 1975].

$$\tau_l = \rho_{\mathrm{w}} \cdot D^5 \cdot K_{\mathrm{q}} \cdot |n| \cdot n \tag{6.34}$$

Where:

ρ_{w} = the temperature dependent density constant of water
D = the diameter of the propeller
K_{q} = a propeller constant
n = the number of revolutions per second

The number of revolutions n is considered to be positive at all times $|n| \cdot n = n^2$. Replacing the driving torque with $K \cdot I$, the viscous friction with $b \cdot \omega$ and $K_{\mathrm{q}} \approx 0.01284103056$ the equation for inertia is found as seen in 6.35. Details regarding the calculation of K_{q} can be found in appendix H on page 123

$$J\frac{d\omega(t)}{dt} = K \cdot I(t) - b\omega(t) - \rho_{\mathrm{w}} \cdot D^5 \cdot 12.84103056\text{E-}3 \cdot n^2 \tag{6.35}$$

Rewriting 6.35 to 6.36 where $C_{\mathrm{propeller}}$ and ω describes the load τ_l effect on the motor as a torque that is dependent on the number of revolutions n. The number of revolutions is converted to a angular velocity ω by dividing by 2π.

$$J\frac{d\omega(t)}{dt} = K \cdot I(t) - b\omega(t) - C_{\mathrm{propeller}} \cdot \omega(t)^2 \tag{6.36}$$

From the model of the the electric equation 6.37 is known.

$$U(t) = R_{\mathrm{a}} \cdot I(t) + L_{\mathrm{a}} \cdot \frac{dI(t)}{dt} + K \cdot \omega(t) \tag{6.37}$$

From these two equations it can be noticed there is a nonlinearity.

Handling nonlinear parts and modelling the system

By looking at the equations it can be seen that the torque load are non linear and dependent on the angular velocity. This nonlinearity is resolved by splitting up ω^2 into a constant operating point ω_0 multiplied by ω, where the operating is a known constant.

$$\omega^2 \approx \omega_0 \cdot \omega \tag{6.38}$$

With the nonlinearity resolved a model is constructed from equation 6.37 and 6.36 as seen in figure 6.8 on the next page. It should be noted that the angular velocity is converted into a force $\left[\frac{N}{m}\right]$ using formula 6.9 on page 37 described by ω and C_{thrust}.

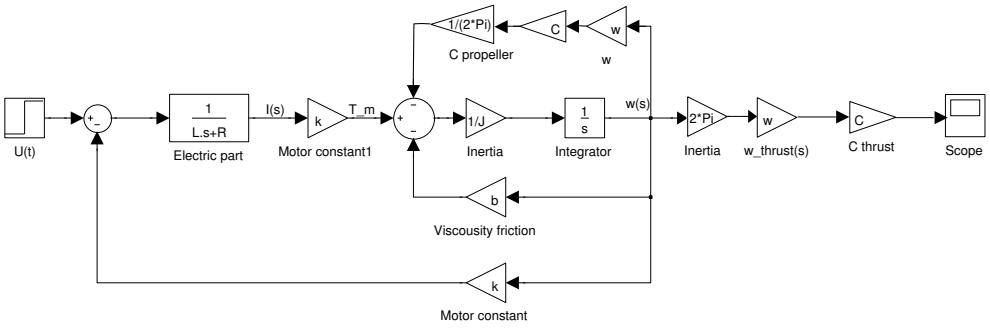


Figure 6.8: A model of a DC-motor system.

Transfer function of DC-motor system

In order to construct a controller it is needed to determine the transfer function for the above model. This is done by reducing figure 6.8 and determine the closed loop function.

Reducing from the inside out the double feedback loop; torque load and the viscosity friction becomes equation 6.39.

$$\frac{1}{2\pi} C_{\text{propeller}} \cdot \omega_0 + B = B + C_{\text{propeller}} \cdot \omega_0 \tag{6.39}$$

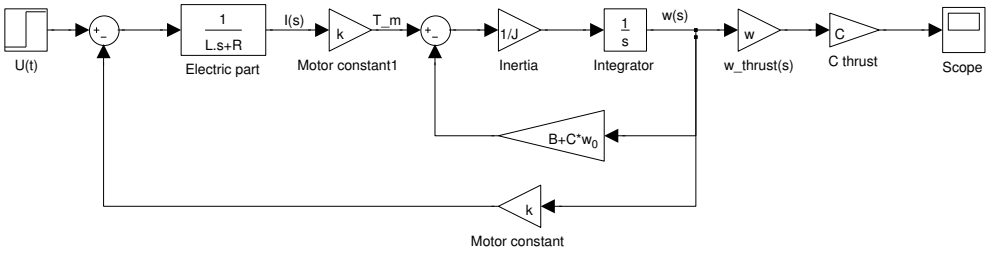


Figure 6.9: Eliminating a feedback loop

This new combined feedback loop is then reduced by determining the closed loop function $\dfrac{H(s)G(s)}{1 + H(s)G(s)D(s)}$ as shown in equation 6.40 and shown on figure 6.10 on the following page.

$$\frac{\frac{1}{J} \cdot \frac{1}{S}}{1 + \frac{1}{J} \cdot \frac{1}{S} \cdot (B + C_{\text{propeller}} \cdot \omega_0)} = \frac{1}{Js + B + C_{\text{propeller}} \cdot \omega_0} \tag{6.40}$$

Calculating the final feedback loop gives equation 6.41 on the next page.

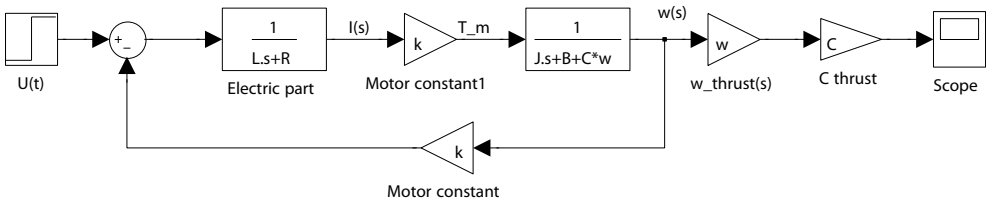


Figure 6.10: Eliminating a feedback loop

$$\frac{\frac{1}{Js+B+C_{\text{propeller}}\cdot\omega_0}\cdot\frac{1}{Ls+R}\cdot K}{1+\frac{1}{Js+B+C_{\text{propeller}}\cdot\omega_0}\cdot\frac{1}{Ls+R}\cdot K\cdot K} = \frac{K}{s^2(JL)+s(JR+LB+LC_{\text{propeller}}\omega_0)+(BR+RC_{\text{propeller}}\omega_0+K^2)} \tag{6.41}$$

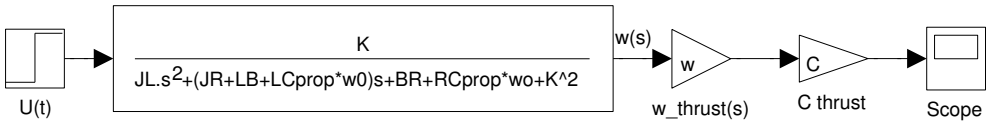


Figure 6.11: The fully reducered system

Completing the transfer function by multiplying with the forward thrust equation gives equation 6.42

$$H(s) = \frac{KC_{\text{thrust}}\omega_0}{JLs^2+s\left(C_{\text{propeller}}L\omega_0+JR+BL\right)+(BR+RC_{\text{propeller}}\omega_0+K^2)} \tag{6.42}$$

This transfer function describes the transformation of voltage into a current into angular velocity and into a force used to push the AUV forward.

By inserting values found as described in appendix C on page 109 the following transfer function is found for the DC-motor:

$$H(s) = \frac{91\text{E}{+}12}{2.47\text{E}{+}6\cdot s^2+1.85\text{E}{+}9\cdot s+8.42\text{E}{+}12} \tag{6.43}$$

Where:

$$\text{C}_{\text{propeller}} = 1025\cdot 0.037^5\cdot 12.84103056\text{E-}3 = 912.71\text{E-}9$$
$$\omega_0 = 50\text{ RPS} = \frac{50}{2\pi}\text{ rad/s} \approx 8\text{ rad/s}$$

By including the forward thrust factor the transfer function becomes:

$$H(s) = \frac{7.4\text{E+}12}{2.47\text{E+}6 \cdot s^2 + 1.85\text{E+}9 \cdot s + 8.42\text{E+}12} \tag{6.44}$$

By locating the poles of the system described in 6.44 it is seen there is a complex conjugated pole in $(-374.5 \pm 1809.1j)$. Because this complex pole is far away from the center it can be ignored and thus the step response for the DC-motor becomes as shown in equation 6.45 figure 6.12.

$$H(s) = \frac{7.4\text{E+}3}{1.85s + 8.42\text{E+}3} \tag{6.45}$$

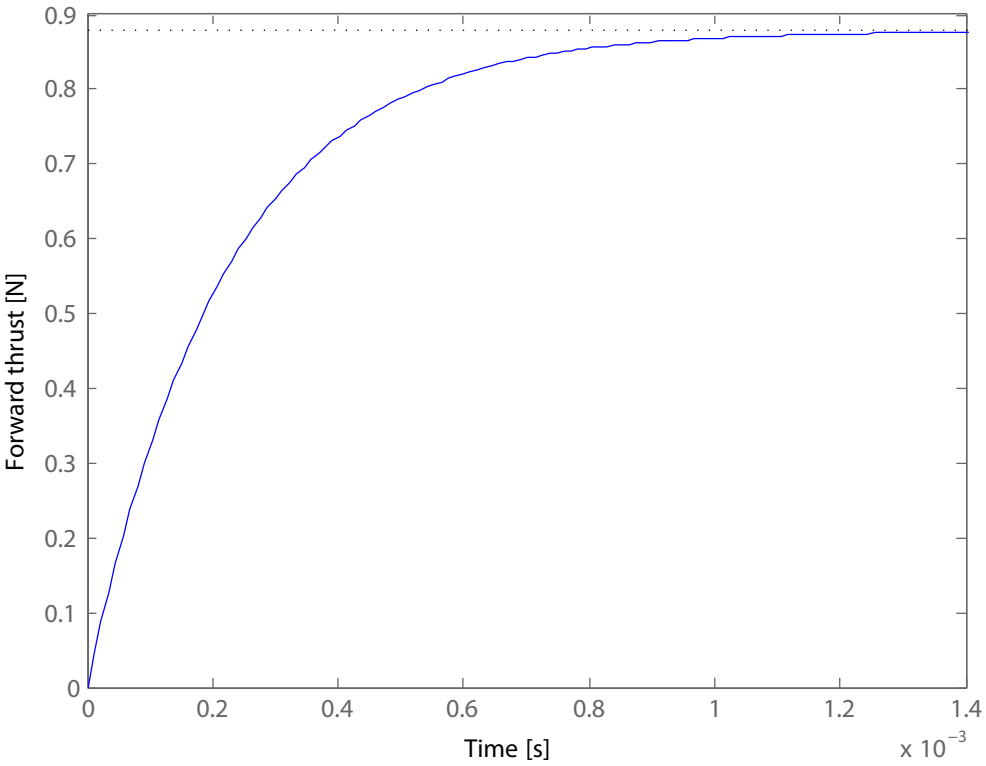


Figure 6.12: Step response for the DC-motor including the linearized propeller drag

The time it takes the DC-motor to reach a steady state is in the range of 1 ms. This is an acceptable time as the full AUV system is far slower to react. However it can been seen that the motor do not reach 1 with a input step of one. This can be corrected with a proportional controller that will improve rise time and the approximation to 1. This is not implemented on the AUV, which means the DC-motor is considered to have a constant transfer function H(s)=1.

6.6 Rapid prototyping using Simulink

For verification purposes and the ability to test various aspects of sailing with the AUV a model has been made in Simulink. The model is based on the functions calculated in the earlier sections, and has three main parts. One part is the forward motion of the AUV. This is illustrated on 6.13 The second

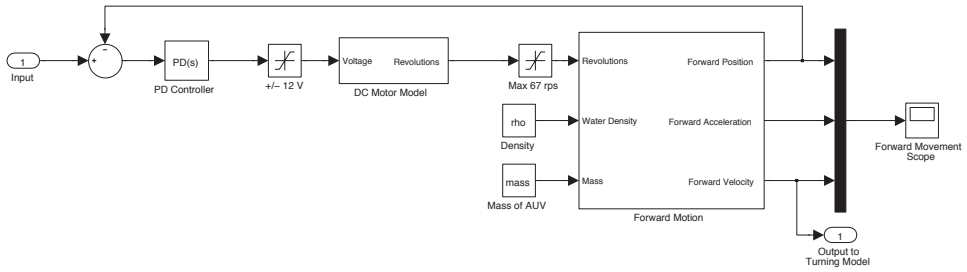


Figure 6.13: Simulink prototype of the forward motion

part of the Simulink model, is the up- and downward motion of the AUV. This is seen on 6.14. This figure takes the input as a desired height, and adjusts the mass accordingly. The last part of the model, is the controller for turning

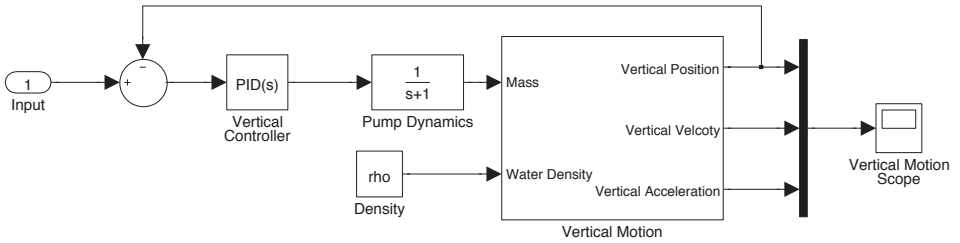


Figure 6.14: Simulink prototype of the upward motion

the AUV. This adjusts the rudder angle, according to the desired input angle (eg. which way the AUV should turn. This model is seen on figure 6.15. For

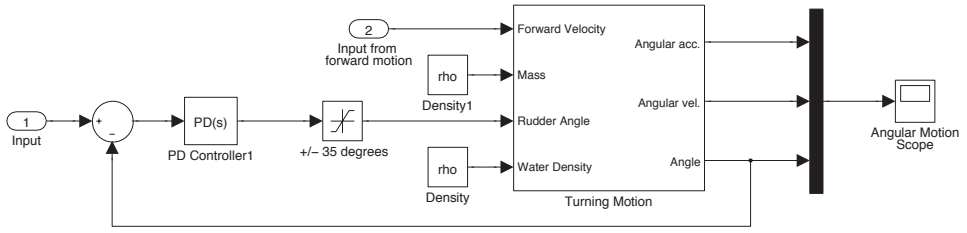


Figure 6.15: Simulink prototype of the turning motion

rapid prototyping purposes Simulink proves a valuable tool. When designing and implementing controllers, these can be tested straight away. This can be implemented as a PD controller, and simulated straight away, to see whether the design is useful or not. The model can never be completely accurate, but will always be an assumption due to external factors that can not be taken into account in the model. To take these into account, a more precise model is to be made. On the CD `/data/simulink` a complete model can be found, which has been used for testing purposes.

Things to note in the model is the saturation blocks used to saturate the inputs and outputs of the functions—these are physical constraints. For instance the motor terminal voltage is limited to ± 12 V as this is the voltage on the battery, and a simulation with a voltage of higher amplitude should be impossible.

7 Control modelling

This chapter describes the development process of a controller used for the AUV. The controller will be developed as a Proportional-Integral-Derivative (PID) controller which works as a single-input-single-output, thus using for example the current input to the DC-motor to control the distance travelled by the AUV.

The chapter is divided into sections for each controller that is to be used. Which in accordance with the requriement specification and the chapter on 6 on page 33 is to be composed of:

- Controller for the for- and backward motion
- Controller for the up- and downward motion
- Controller for the turning motion

A PID controller consists of three parts, namely a proportionate part, an integral part and a derivative part. The algorithm for a PID controller is defined as:

$$\mathrm{u}(t) = K_\mathrm{p} e(t) + K_\mathrm{i} \int_0^t e(\tau) d\tau + K_\mathrm{d} \frac{d}{dt} e(t) \tag{7.1}$$

Where:

$\mathrm{u}(t)$ = the reference signal
K_p = the proportional part
K_i = the integral part
K_d = the derivative part
$e(t)$ = the error on the controller
$e(\tau)$ = the error at the instantaneous time (present)

The time domain representation. When expressed in the Laplace domain the expression becomes:

$$\mathrm{u}(s) = K_\mathrm{p} \left(1 + K_\mathrm{d} s + K_\mathrm{i} \frac{1}{s} \right) \tag{7.2}$$

The different terms in equation 7.1 and 7.2 is used to shape the error signal so it is as close to the reference signal $\mathrm{u}(t)$ as possible. Several ways can be used to accomplish the desired signal, but firstly, the demands to the signal is to be mapped. For instance, how fast the system should converge, how much overshoot is allowed and such. To determine the constants K_p, K_i and K_d for the system, Rouths stability criterion is used, which is described in the following section.

7.1 Routh array

To set the values for K_p, K_i and K_d different approaches can be used, but to kill two birds with one stone, Rouths stability criterion is used for this, as this determines stability as well. To determine the stability of a higher order system the locations of the roots in the system are calculated. This is then converted to a charactersitic equation on the form:

$$a(s) = a_0 s^n + a_1 s^{n-1} + .. + a_{\mathrm{n}-1} s + a_\mathrm{n} \tag{7.3}$$

Which can be placed in the Routh array given by the formula in [Gene F. Franklin, 2009].

$$\begin{bmatrix} s^n: & a_0 & a_2 & a_4 & \dots \\ s^{n-1}: & a_1 & a_3 & a_5 & \dots \\ s^{n-2}: & b_1 & b_2 & b_3 & \dots \\ s^1: & * & & & \\ s^0: & * & & & \end{bmatrix} \tag{7.4}$$

Where:

$$b_1 = -\frac{\det\begin{bmatrix} a_0 & a_2 \\ a_1 & a_3 \end{bmatrix}}{a_1}, \dots, b_n = -\frac{\det\begin{bmatrix} a_0 & a_{2+\mathrm{n}} \\ a_1 & a_{3+\mathrm{n}} \end{bmatrix}}{a_1}$$

This presents a necessary and sufficient condition to determine the stability of a n-th order system:

> *"A system is stable is and only if all the elements of the first column of the Routh array are positive."* [Gene F. Franklin, 2009, p. 150–151]

If Rouths stability criterion is fulfilled, there are only poles in the left half plane. Because of this a range of values can be determined as it is known that all values should be greater than zero. This is used to calculate a range of values the coefficients has to be within, which both takes account for stability, but also computes the values of the system.

7.2 Controlling for- and backward motion

To make a control system for the thrust, the formula derived in the dynamics chapter on 6.2 on page 35 can be used. If set into a free-body diagram, the AUV looks like figure 7.1 on the facing page, (a) shows the translation to a general system, where the drag acts as a damper on the forward motion of the AUV. The formula for describing the forward motion of the AUV can be described by the following formula:

$${}^s\mathbf{F}_\mathrm{thrust} - ({}^s\mathbf{F}_\mathrm{drag}(v_\mathrm{auv}) + {}^s\mathbf{F}_\mathrm{d'Alembert}(\dot{v}_\mathrm{auv})) = 0 \tag{7.5}$$

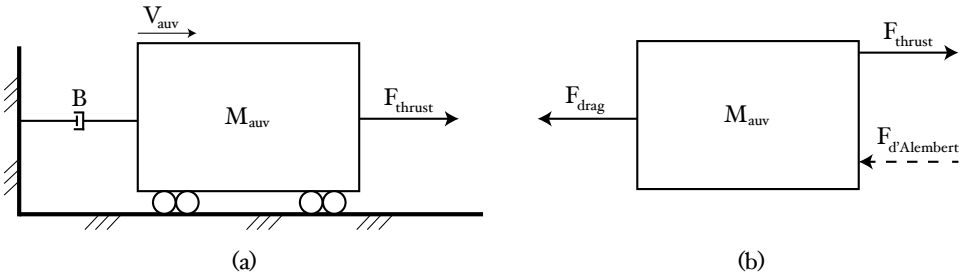


Figure 7.1: (a) Translational system for the AUV. (b) Free body diagram of the AUV.

The ${}^{s}\mathbf{F}_{\text{d'Alembert}}$ is a fictious force on the AUV (shown on (b)) and is only considered a force, as it keeps the AUV from moving. The system can be put into the general equation as written in [Charles M. Close, 2002], which is illustrated on figure 7.2. It shows a model for the system, from where to

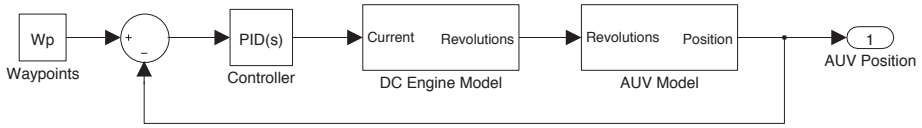


Figure 7.2: Model for the regulation system.

design the controlling-algorithm. As stated in the requirement specification, the only requriements to the regulator is that it should be able to sail in a straight line and reach the way point with a precision of ± 1 m as specified in the requirement specification.

When a controller is added to the system, it makes the input converge toward the reference signal, which is given by way points sent wirelessly to the AUV. The system is then to make the AUV thrust forward by controlling the input to the DC-motor (which is linked to the propeller) thus driving the AUV forward until the destination is reached. To decide how the AUV acts when moving through the water, the characteristics of the DC-motor is also taken into account. This is given in the section 6.5 on page 44, and when fed back, the transfer function becomes:

$$H(s) = \frac{0.1278}{s(s + 17.242)(s + 17.241)(s + 7.792\text{E-}9)} \tag{7.6}$$

This gives the pole-zero plot on figure 7.3 on the next page. To design the controller, the two farthest poles are omitted, and the controller is designed around the two poles located in origo on the pole-zero plot of figure 7.3 on the following page. The transfer function of the system only using the poles

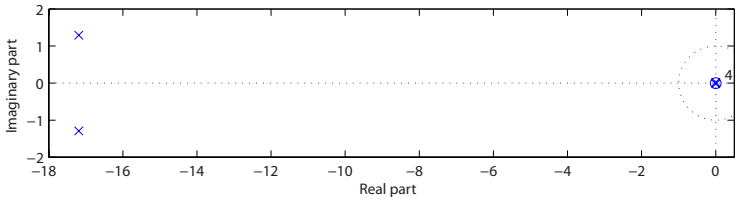


Figure 7.3: Pole-zero plot of the 4th order system

located in zero then becomes:

$$H(s) = \frac{0.1278}{s^2} \tag{7.7}$$

Which is the system to design the controller about. As a system which contains a double pole can not be regulated by a single P-controller it is looked into how a PID-controller can be designed. The PID-algorithm, when Laplace transformed this can be written as:

$$\mathrm{PID}(s) = K_p \left(1 + K_d \cdot s + K_i \cdot \frac{1}{s}\right) \tag{7.8}$$

Which is then multiplied with the system described earlier. Firstly, it is looked if a PD-controller can be designed, and multiplied in the feedback loop, the system response becomes:

$$T(s) = \frac{0.1278 \cdot K_\mathrm{p}(1 + K_\mathrm{d} s)}{s^2 \left(1 + \frac{0.1278 \cdot K_\mathrm{p}(1 + K_\mathrm{d} s}{s^2}\right)} \tag{7.9}$$

To decide the intervals that K_p, K_d and K_i can be within, is determined by Rouths stability criterion, that states that the numbers in the first column of Rouths stability matrix is to be greater than one. Rouths stability criterion, takes it base in the characteristic formula of the poles, which can be factorized and written:

$$\overbrace{5000}^{a_0} \cdot s^2 + \overbrace{639 \cdot K_\mathrm{p} K_\mathrm{d}}^{a_1} s + \overbrace{639 \cdot K_\mathrm{p}}^{a_2} \tag{7.10}$$

The following Routh array can be made:

$$\begin{bmatrix} s^2: & a_0 & a_2 & a_4 \\ s^1: & a_1 & a_3 & a_5 \\ s^0: & b_1 & b_2 & b_3 \end{bmatrix} \tag{7.11}$$

The routh array states that all the numbers in the first column are to be bigger than one, if the system is to be stable. The following equation for a_1 and b_1 is

used, this is found in [Gene F. Franklin, 2009, p. 152].

$$a_1 > 0 \implies 639 \cdot K_p K_d > 0 \tag{7.12}$$

$$b_1 = -\frac{\det \begin{bmatrix} a_0 & a_2 \\ a_1 & a_3 \end{bmatrix}}{a_1} > 0 = 639 \cdot K_p > 0 \tag{7.13}$$

These two equations are used for determining the values of K_p and K_d. As seen, K_p and K_d can assume any positive values, so for further calculations K_p is set to 50 and K_d is set to 5. When inserted, the transfer function becomes:

$$H(s) = \frac{159750 \cdot s + 31950}{5000 \cdot s^2 + 159750 \cdot s + 31950} \tag{7.14}$$

Which, when plotted, gives the step response in figure 7.4. Matlab is used to

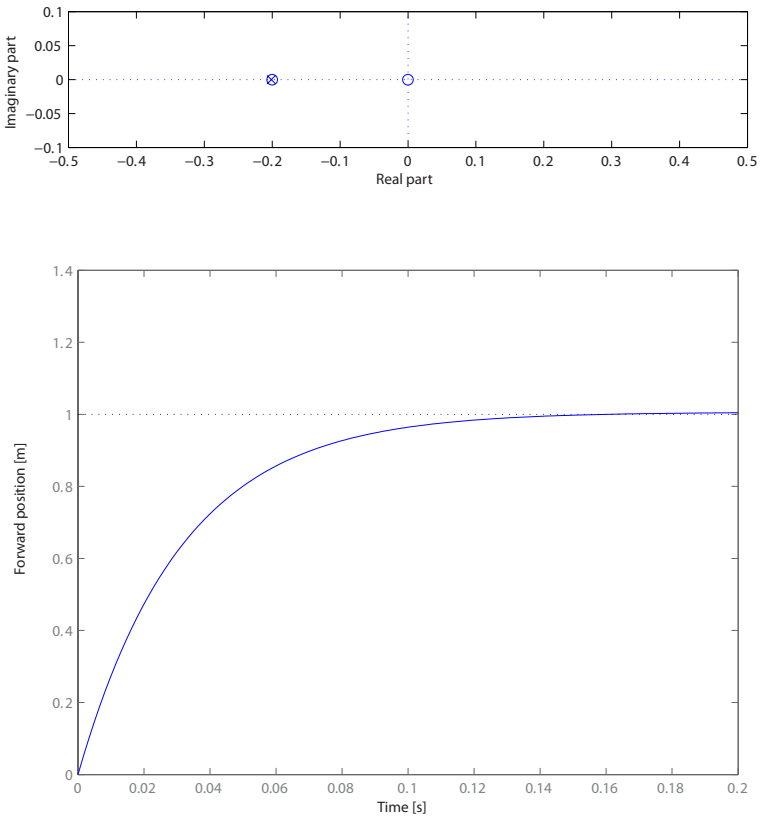


Figure 7.4: Pole-zero plot and step response plot of the forward controller

calculate the constants (such as rise time, overshoot and such) to see if the controller fulfils the criterions set in the requirement specification.

- **Rise time:** 67.5 ms

- **Settling time:** 114.9 ms
- **Overshoot:** 0.56 %

Which gives a tiny amount of overshoot, but as this is half a percent, the AUV has to sail around 200 metres to fail the requirement specified in the requirement specification by missing the way point by more than 1 metre, thus having to reverse back to the way point. This could be resolved by the AUV realizing that the way point has been reached, and move towards the next way point. The next step is to design the algorithm to control the forward motion which is done in section 8.7 on page 73.

7.3 Controlling up- and downward motion

To design the up- and downward controller it is necessary to determine how the system behaves. It is known that the buoyancy of the system is inherently unstable. If it is looked at how the AUV acts under water it cannot maintain the exact same depth, without actively regulating the buoyancy pump, and constantly pumping out water as the AUV is either going up and down. Below is the requirements outlined for the controlled needed.

- Overshoot up to 20 % as to avoid pumping water in and out excessively
- Rise time should be slow but within 15 s
- Settling time should be within 5 % by 60 s

From 6.2 on page 35 the following formula can be written to described the dynamic of the vertical motion:

$$\mathbf{F}_{\text{down}} = \mathbf{F}_{\text{up}} + \mathbf{F}_{\text{drag}} \tag{7.15}$$

This is rewritten to the following using Newtons law $m \cdot a = \sum F$:

$$m \cdot a = \rho_{\text{w}} \cdot \text{Vol} \cdot g + \frac{C_{\text{d}} \cdot \rho_{\text{w}} \cdot V(t)^2 \cdot A}{2} - m \cdot g \tag{7.16}$$

Where:

Vol = the volume of the AUV.
g = the gravitational constant.
ρ_{w} = the density of water.

$$a_{\text{z}} = \dot{V}_{\text{z}} = \ddot{P}_{\text{z}} = \frac{\sum \mathbf{F}_{\text{z}}(V_{\text{z}})}{m_{\text{auv}}} \tag{7.17}$$

It can be seen that $\mathbf{F}_{\text{drag}}$ is dependent of the velocity and which will create a non-linearity. The non-linearity is resolved by choosing a known operating point V_0 and it is assumed that $V^2 = V_0 \cdot V(t)$.

Based on this a preliminary sketch of the system, which is used for further calculations is constructed as seen in figure 7.5. Pump dynamics included in the figure is considered to be similar to the DC-motor modelled in section 6.5 on page 44 and controlled in a manner that allows for a transfer function of $T(s) = 1$.

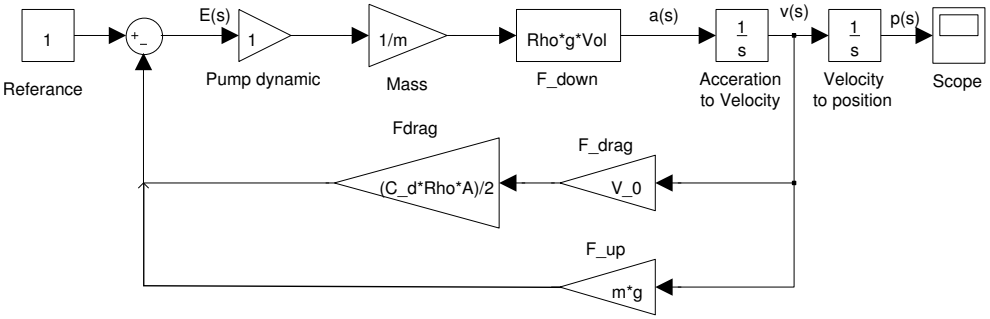


Figure 7.5: Model of the system that should be controlled

For the ease of the calculations the constants have been reduced to:

$$K_1 = F_{\text{up}} = \rho \cdot g \cdot \text{Vol} = 1025 \cdot 9.82 \cdot 7.3\text{E-}3 = 73.47815$$
$$K_2 = F_{\text{drag}} = \frac{A \cdot C_{\text{D}} \cdot \rho \cdot V_0}{2} = \frac{0.82 \cdot 0.165 \cdot 1025 \cdot V_0}{2} = 69.34 \cdot V_0$$
$$K_3 = F_{\text{down}} = m \cdot g = 7.3 \cdot 9.82 = 71.686$$

From figure 7.5 a transfer function is calculated which results in the following equation:

$$H(s) = \frac{k_1}{s(m \cdot s + k_1 k_2 + k_1 k_3)} \tag{7.18}$$

By inserting values for the constants, assuming a operating point V_0 at 1 m/s, it is reduced to:

$$H(s) = \frac{2.01\text{E+}6}{s(200s\text{E+}3 + 283.9\text{E+}6)} \tag{7.19}$$

Due to Routh stability criterion [Gene F. Franklin, 2009] this system is unstable because some of the coefficients are missing (zero). Furthermore it is seen that there are two poles in (0,-1419.509785), and the step response shown in figure 7.6 on the next page. Because the system is highly unstable a PID controller of the form $K_{\text{p}}(1 + K_{\text{d}}s + K_{\text{i}}\frac{1}{s})$ is added resulting in the transfer function shown in 7.20 on the following page. It is assumed that the pressure sensor mounted on the AUV in conjunction with software has a unity feedback of 1 such that the water pressure is mapped from pressure to a voltage. Figure 7.7 on the next page shows how the PID-controller is connected into the system.

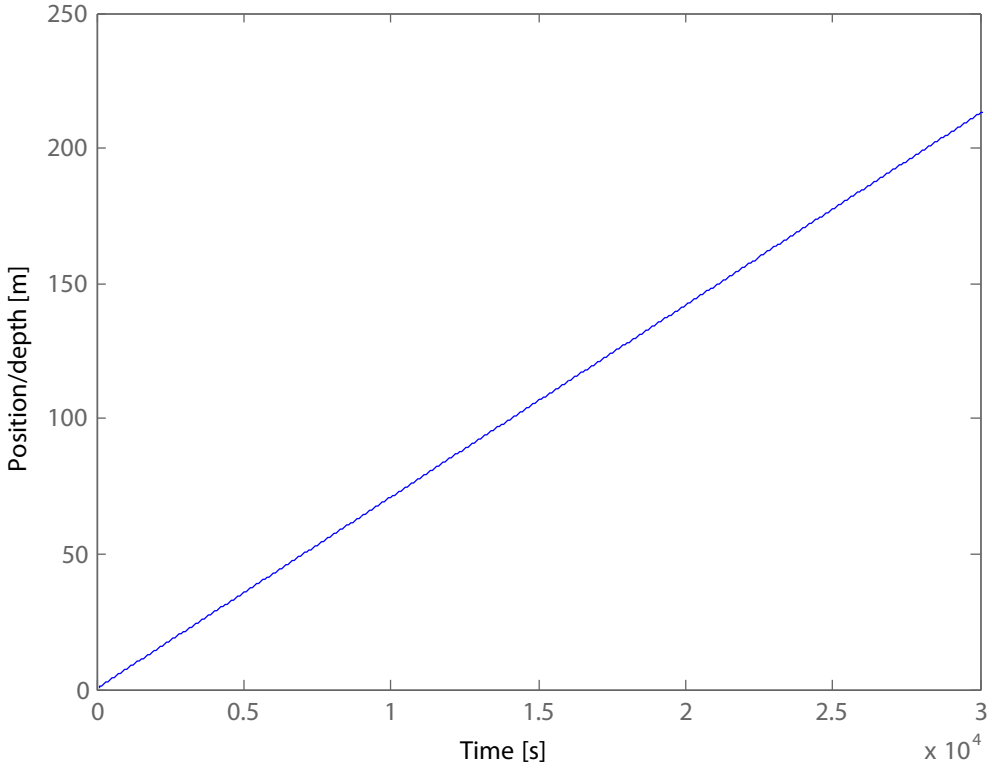


Figure 7.6: Step response of the uncontrolled up and down model

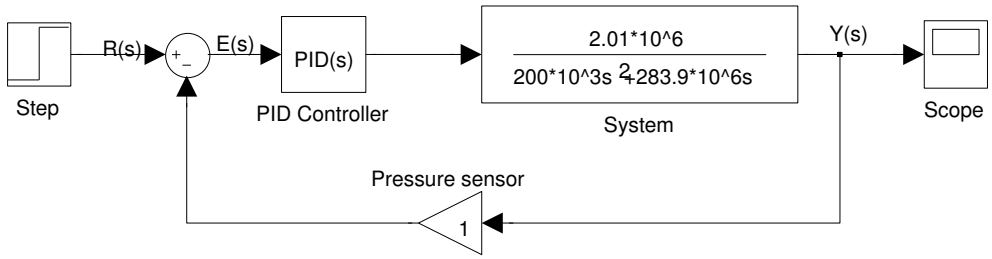


Figure 7.7: Model of the system including a PID controller with a unity feedback

$$H(s) = \frac{2.01\text{E}6(s + k_\text{d}s^2 + k_\text{i})k_\text{p}}{200\text{E}3 \cdot s^3 + (2.01\text{E}6 \cdot k_\text{d}k_\text{p} + 283.9\text{E}6)s^2 + 2.01\text{E}6 \cdot k_\text{p}s + 2.01\text{E}6 \cdot k_\text{p}k_\text{i}} \tag{7.20}$$

Reduced by a factor of 10^4 this gives:

$$H(s) = \frac{201k_\text{p}(s + k_\text{d}s^2 + k_\text{i})}{20s^3 + (201k_\text{p}k_\text{d} + 28390)s^2 + 201k_\text{p}s + 201k_\text{p}k_\text{i}} \tag{7.21}$$

From this the characteristic equation is given by:

$$20s^3 + (201k_\mathrm{p}k_\mathrm{d} + 28390)s^2 + 201k_\mathrm{p}s + 201k_\mathrm{p}k_\mathrm{i} = 0 \tag{7.22}$$

In order to determine the range of K_p,K_d and K_i the Routh array is built as seen in table 7.11 on page 56.

$$\begin{bmatrix} s^3: & 20 & 201K_\mathrm{p} & 0 \\ s^2: & 201K_\mathrm{d}K_\mathrm{p} + 28390 & 201K_\mathrm{p}K_\mathrm{i} & 0 \\ s^1: & \frac{201K_\mathrm{p}(-20K_\mathrm{i}+201K_\mathrm{p}K_\mathrm{d}+28390)}{201K_\mathrm{d}K_\mathrm{p}+28390} & 0 & 0 \\ s^0: & 201K_\mathrm{p}K_\mathrm{i} & 0 & 0 \end{bmatrix} \tag{7.23}$$

Therefore in order to have stability it is necessary be within the following value range:

- $K_\mathrm{i} > 0$
- $if K_\mathrm{d} = 0$ then $K_\mathrm{p} = K_\mathrm{p}$
- if $K_\mathrm{d} > 0$ then $K_\mathrm{p} > -\frac{141.24}{K_\mathrm{d}}$
- $K_\mathrm{p} > 0$
- $10.05 \cdot K_\mathrm{d}K_\mathrm{p} + 1419.5 > K_\mathrm{i}$

In order to meet the stability requirements K_p and K_i should be larger than 0 but still as small as possible to allow for a slow rise time and minimal overshoot. From the stability value ranges it can be seen that K_d has no effect and is thus omitted effectively making the PID-controller into a PI-controller.

By trial and error a solution is found using Matlab as seen i figure 7.8 on the next page. Firstly a realistic K_p-value is found as this gives a approximate rise time. Determining the K_i is done in steps as to determine the range of overshoot and settling time. In the case of the up and down controller a value of approximate 20 is found for K_p. With a K_p of 20 it is determined that the K_i value could be around 0.05 for a small overshoot. For a shorter settling time the K_i value should be increased, however this will in exchange increase the overshoot as well. From figure 7.8 on the following page the following values are found: From 7.8 on the next page and 7.1 transfer3 is chosen as the rise

	Transfer1	**Transfer2**	**Transfer3**
Overshoot [%]	58.91	25.1	16.9
Settling time [s]	58.3	99.3	60.9
Rise time [s]	2.86	7.53	9.39

Table 7.1: Overshoot, settling time and rise time for different constants

time is fairly realistic, while the settling time is acceptable. The overshoot is 17 % which is within the requirements.

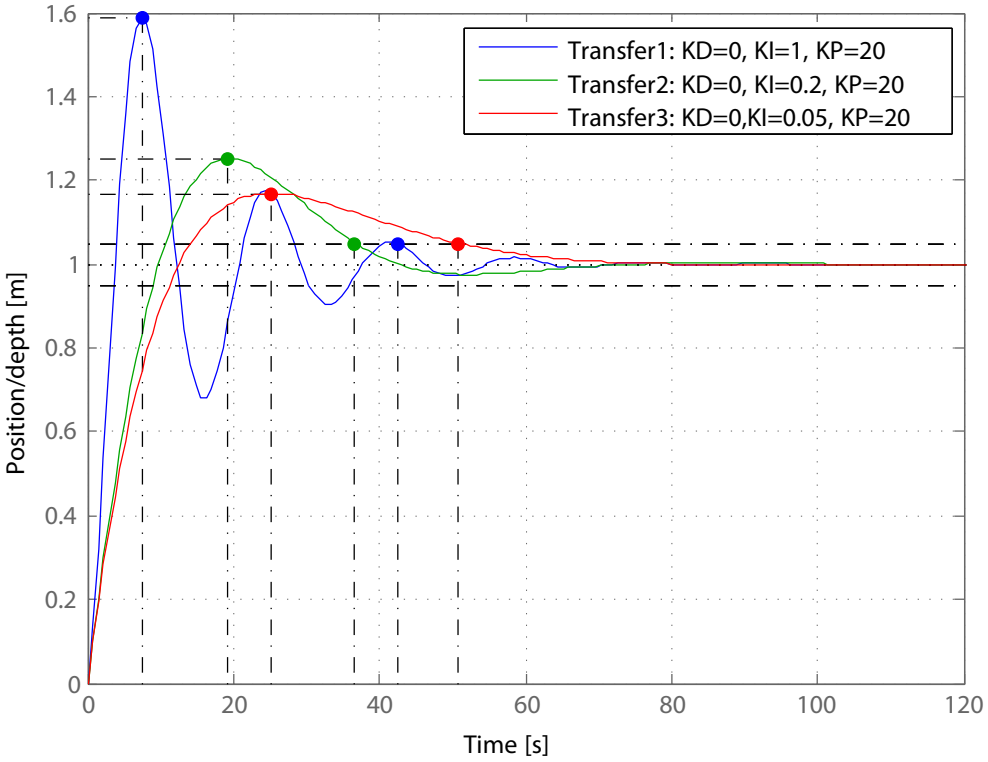


Figure 7.8: Step response of the PID controlled system. Peak times and settling time is marked by dots on the figure.

Summation

It has been determined that this system is highly unstable, but by constructing a PID-controller it is possible to make the system stable by actively controlling the pump. The PID-values found is as following:
$K_\mathrm{p} = 20$
$K_\mathrm{i} = 0.05$
$K_\mathrm{d} = 0$

7.4 Controlling turning motion

The last thing to design a controller for, is the process that makes the AUV turn. This is different from the two other controllers, as this moves between the two frames, as the body-fixed frame cannot describe the action of turning, but the Earth-fixed frame can, as this would be a rotation about the origo of the AUV. As before the system can be depicted as a general feedback system. The transfer functions in the system blocks in the Simulink models in the

section are dummy transfer functions. A more detailed figure is used to depict

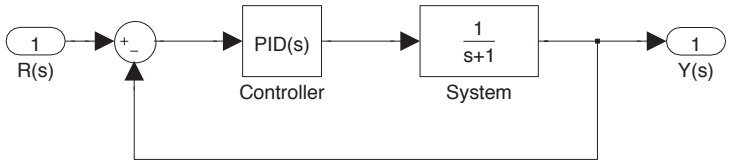


Figure 7.9: System model for the turning controller

the transfer function of the system, that again can be expressed via its transfer function. This is then used to design the controller depicted on figure 7.9. As before, a PD-controller is used to control the system and as before the stability criterion is used to decide at which intervals the different coefficients should be between. For designing a controller, the transfer function of the turning system on figure 7.9 is to be calculated, this is done by a more comprehensive figure of the system that includes the drag. Figure 7.10 depicts the controller

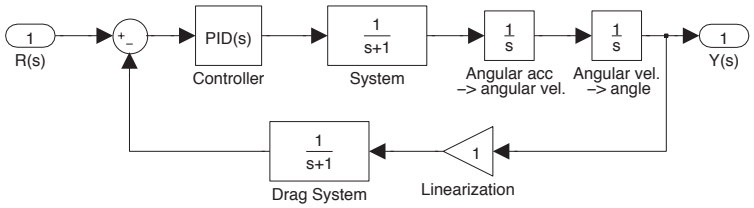


Figure 7.10: Precise system model for the turning controller

and a more comprehensive description of the system, this is used to calculate the transfer function. As many of the variables in the function are static, these are inserted, and an expression for the turning motion of the AUV with respect to the angle of the rudder and the monumental velocity can be given as:

$$\frac{\theta(s)}{\delta V^2(s)} = \frac{20.73}{0.07s^2 + 59.17\omega_0} \tag{7.24}$$

Equation 7.24 describes the relation between the angle of the rudder, and the angle of the AUV. This is the transfer function that uses a reference signal (for example an angle the AUV should turn) and through here steer the rudder to the desired position, and uses a gyrometer measurement, to determine if the desired angle is reached. As the gyrometer gives a readout of the angle turned, this can be subtracted from the reference signal, to see if the AUV has reached the desired angle. From this, the Routh array is to be set up, to determine the values of the controller. This is done by multiplying with a term that describes the PID-controller. This is given as $K_\mathrm{p}\left(1 + K_\mathrm{d}s + K_\mathrm{i}\frac{1}{s}\right)$

which is inserted in the block on the figure noted controller. The Routh array is calculated as described in section 7.1 on page 54. The values of the terms are then calculated to be:

- $K_\mathrm{p} = 20$
- $K_\mathrm{d} = 1$
- $K_\mathrm{i} = 3$

When inserted the transfer function for the PID-controller becomes:

$$H(s) = 20(1 + s + 3\frac{1}{s}) \tag{7.25}$$

The transfer function is then used to design a controller in the following chapter.

7.5 Control conclusion

Having designed all the controllers, the next step is to get these implemented. As written in the episode, each of the controller takes its base in the models designed for the specific controllers, and from these, a controller is designed. The individual transfer functions of the systems are given as:

- Transfer function of the for- and backward controller: $H(s) = 50(1+5s)$
- Transfer function of the up- and downward controller: $H(s) = 20(1 + 0.05\frac{1}{s})$
- Transfer function of the turning controller: $H(s) = 20(1 + s + 3\frac{1}{s})$

In the following chapter an implementation of these are described.

8 Software design

In order to implement a real time software system, the functional needs are divided into subsystems. This is done throughout the following chapter with rich image diagrams, flowcharts and descriptions of the underlining operating system Free Real Time Operating System (FreeRTOS).

8.1 Subsumption

The subsumption is intended to clarify the functional needs in the system. This way of visualizing the software can come in handy in the planning stages of the development process.

The subsumption architecture [Brooks, 1986] is based on a modularized way of organizing software. The architecture divides the different functions in a system into a number of levels, where the lowest level contains the simplest functions closest to the hardware and the higher levels combines the lower functions into more complex functions. It always assumes that the lower levels work as intended. The idea behind the subsumption architecture is that the functions can be designed in modules, where the different modules utilizes one or more modules at a lower level, which then again may utilize modules on an even lower level. Each layer does not need to understand each others operations, but will just get the job done as efficient and correct as possible. Subsumption is a frequently used method of making autonomous systems and a further description for this project will follow.

It is very useful to look at the software functionality instead of looking at the specific functions. In this way the software will be developed with functionality in mind instead of specific functions. This allows adoption of new and better ideas during the development process, opposed to developing specific functions in which case new functions are not easily implemented.

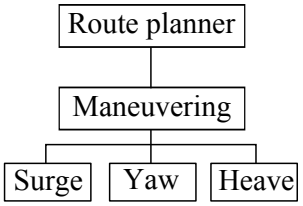


Figure 8.1: Subsumption figure with "Route planner" being the highest level and "Surge", "Yaw" and "Heave" being the lowest.

A subsumption model is made to show the principle of the AUV manoeuvering, this is shown in figure 8.1. In this case the subsumption describes how the AUV travels from way point to way point using the "Route planner". This is

the highest level that makes the AUV sail autonomously. Below this is "Manoeuvering". This is where the system calculates the settings of the actuators. The lowest level contains the blocks used for each actuator, which makes the actuators move according to the specified parameters. These are the three levels used in planning and sailing a route.

8.2 Rich image

To identify functions in the software, a rich image is constructed from the use cases specified in chapter 4 on page 19. From the rich image specific tasks will be derived and deadlines for a system response will be found. The rich image can be seen in figure 8.2. In the centre of the figure is the AUV microcontroller, which is where all data from the sensors and ground station are processed and the actuators are controlled.

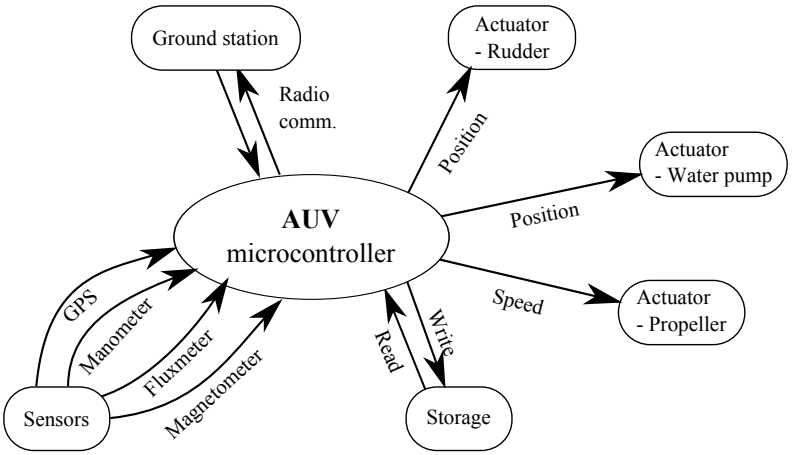


Figure 8.2: Rich image

From this rich image, a more detailed rich image is constructed, figure 8.3 on the facing page, in order to determine internal tasks in the programming. These tasks are then used to create a base structure for the software, see section 8.1 on the previous page for the resulting subsumption model. The detailed rich image is constructed by opening up the AUV's microcontroller, identifying internal tasks resulting in figure 8.3 on the facing page.

The individual tasks are prioritized so the functionality to different peripherals is only accesed by one task (hence the task categorization), hereby making an interface to make the different functionality they represent able to be run with different frequency and priority.

The AUV's microcontroller will execute three tasks consisting of:

Sensor read and storage Is a task that periodically receives sensor data and subsequent manages storing the data on shared memory.

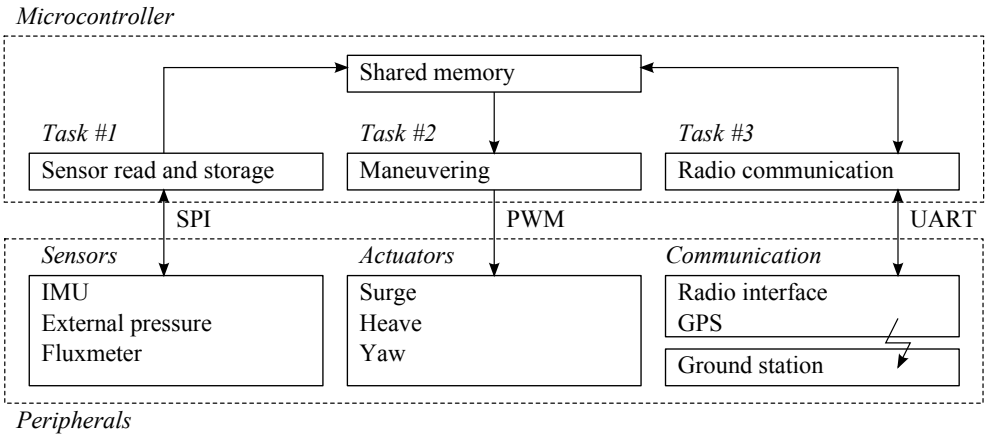


Figure 8.3: Division into tasks

Manoeuvering Is a periodic task that will handle the route planning, control loop and calculate the PWM signal needed to activate the external actuator units correctly.

Radio communication Is a sporadic task that will initiate radio communication, whenever the AUV is at the surface. It will handle the commands from the ground station as well.

From the rich image, calculations on how often each task should be executed, and the time within which the system must respond are computed. See figure 8.1 on the next page for a summery of the following. The sensor and the manoeuvring task are periodic due to the need of constant regulation of the actuators.

The sensors supply measurements to the main system at a frequency of 20 Hz or every 50 ms and the system must respond in less than 10 ms after that. A 20 Hz sample rate is chosen to calculate the mean over several samples. The system must respond before a new measurement from the sensors is requested.

The actuators receive a continuous PWM signal, with a frequency of 50 Hz and the duty cycle is adjusted every 0.5 seconds. The actuators are expected to respond within 0.5 seconds by physically changing angle or speed. Radio communication between the AUV and the ground station is sporadic, however the AUV should surface at least once every 60 minutes, even if it does not reach a way point.

Because it is not critical that the system complete every task before the deadline, the system is a soft real time system.

A single missed deadline on a sensor read or the actuator set functions is at the most going to cause a slight sway in the AUV's path, that will be corrected by later adjustments thereby rendering the deadlines non-critical.

Task	Regularity	Arrival [ms]	Deadline [ms]
Sensor read and storage	Periodic	50	10
Manoeuvring	Periodic	500	500
Radio communication	Sporadic	–	–

Table 8.1: Tasks with deadlines

8.3 Software structure

To get a clear overview of how the software is structured figure 8.4 has been constructed. Figure 8.4 is based on figure 8.1 from subsumption.

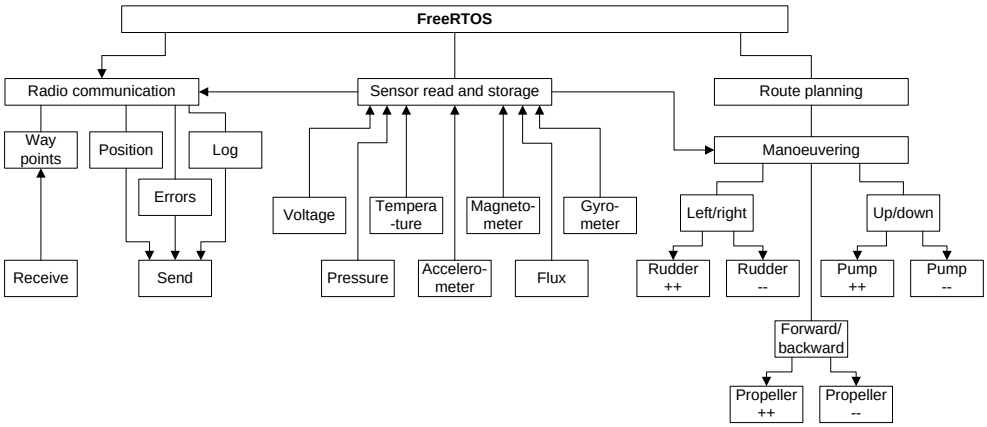


Figure 8.4: This figure illustrates the overall calling sequence, from a higher level all the way down to the hardware.

The "Sensor read and storage" part is built up of series of sensors. Though these sensors are placed beneath the "Sensor read and storage" they can still be accessed by other functions such as the "Manoeuvering" function. The "Radio communication" part handles the data exchange with the ground station, e.g. transfer of logfiles and new way points. The "Manoeuvering" part is the control system and uses the actuator functions to manoeuvre according to the way points.

Data protocol

To ensure the ground station software and the AUV understand each others data, they have to use the same data type. Sensor log data is sent as a 16 bit character instead of "normal" 8 bit characters to represent the numbers. Every set of data is ended with a Line Feed (LF) and Carriage Return (CR) character to move to the next line. This data protocol have been developed to save bandwidth.

There are 12 values (16 bits each) to be sent and the LF and CR characters (8 bits each). Thus the size of a data set will be (assumed a sampling frequency of 20 Hz):

$$12 \cdot 16 \text{ bits} + 2 \cdot 8 \text{ bits} = 208 \text{ bits} = 26 \text{ bytes}$$
$$20 \text{ Hz} \cdot 26 \text{ bytes} = 520 \text{ bytes/s}$$

By this it follows that the communication module have to be able to transmit with a baud rate of at least 520 bytes/s. Sending a log with 10 minutes of data takes 10 minutes at this baud rate. Therefore it is desirable that the baud rate is much higher.

8.4 Task scheduling

The software is organized in three tasks, that run on the FreeRTOS kernel. The tasks group together functions that need to run at the same intervals.

The tasks functions are described in section 8.2 on page 66. But as a brief reminder the sensor reading task is executed with a frequency of 20 Hz, the manoeuvring task is executed at 2 Hz and the communication only executes sporadicly.

Splitting functions into different tasks serves multiple purposes. First of all it assures that the tasks are executed at the appropriate intervals. By separating the different functions into different tasks, the organization of the program becomes more apparent, thus the designer does not have to worry about execution time. The latter could also be achieved by using a hardware timer with an interrupt, but this does not facilitate task handling, which is a practical abstraction to make the system real time.

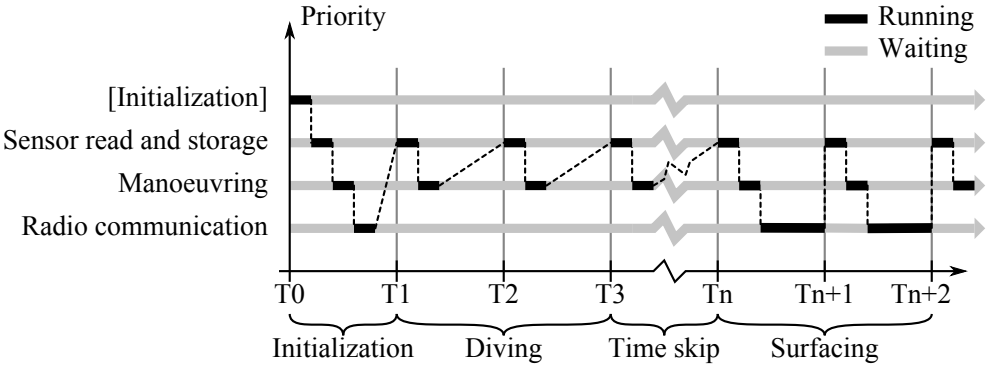


Figure 8.5: An example of scheduling where the system initializes, then dives and surfaces to communicate again. The time skip is to illustrate that it will be diving for much more than three periods, while the two last periods (while surfacing) is illustrating that the radio communication is interrupted by the sample task.

FreeRTOS

In this project FreeRTOS is used to handle the real time execution of tasks on the microcontroller. FreeRTOS is a free operating system designed for embedded systems, and has been ported to a large number of microcontrollers including the AT91SAM7A3 from Atmel used in this project.

FreeRTOS takes care of real time execution of the programs by scheduling the tasks. Thereby FreeRTOS is simplifying the scheduling of tasks for the developer, that can be used for optimizing the tasks instead of worrying about real time issues.

The scheduler in FreeRTOS is run as a preemptive prioritized scheduler in this project.

8.5 Reading sensors

As described in the previous description of the hardware platform, section 5.2 on page 29, it has been chosen to use an IMU which contains different inertial mesurement devices. These sensors are tri-axis gyro-, accelero- and magnetometers. In addition to that it also features a temperature sensor for the internal temperature and an auxiliary ADC. All this data can be read via the SPI interface.

SPI

The SPI standard specifies a serial interface with four logic signals: Serial Clock (SPCK), Master Out Slave In (MOSI), Master In Slave Out (MISO) and Slave Select (NSS) (designations used in the ARM7 datasheet).

SPI is specified to operate with one master device and at least one slave device. The NSS makes it possible to operate more devices on the same bus. To use more devices on the same bus, every device is required to have a tri-state MISO output so their output becomes high impedance when the device is not selected.

To communicate over SPI, the master first sets up a clock at a supported frequency for the slave device. After that, the master selects the desired device, waits if necessary (e.g. for analog to digital conversion) and when ready, the master starts to send clock cycles. During each clock cycle, a full duplex data communication occurs as the master sends a bit on the MOSI line while the slave sends a bit on the MISO line. The clock's phase and polarity is selected using the clock phase and clock polarity options. The data are stored in shift registers of typically 8, 12 or 16 bits length. Timing diagrams for the SPI bus on the ARM7 microcontroller can be found in [Atmel, 2006, p. 236, fig. 27-3 and 27-4].

The protocol to the IMU consists of a 16 bit sequence and the structure is depicted on figure 8.6 on the facing page, where the bit sequence for communi-

cating with the ADIS16405 via SPI is shown. The IMU is capable of returning sampled data when requested by the master. The bits sent to the IMU are a bit to indicate if the master will read or write, a 7 bit address and 8 bit of data. The bits received is a 16 bit sequence of a register readout.

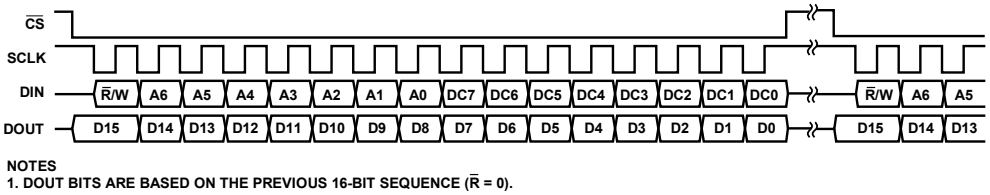


Figure 8.6: Output data register assignments of the IMU. A6-A0 = Register address, D7-D0 = data input, DB14-DB0 = data output. Figure from [Analog Devices, 2009, p. 10].

If write mode is selected, the DC7-DC0 contains eight bits of data to be written while A6-A0 contains the desired register address for writing. If the read bit is set, the address bits indicate which sensor register is to be read, while D7-D0 are ignored. The output data will stay at DB13-DB0 during the next clock cycle. Figure 8.6 illustrates a read cycle. A list of the ADIS16405's address registers is found in [Analog Devices, 2009, page 10].

As described so far, every single sample register is requested seperatly, but the IMU also facilitates a burst read mode, where only one initiating address is sent to request every sample data, this procedure is illustrated on figure 8.7.

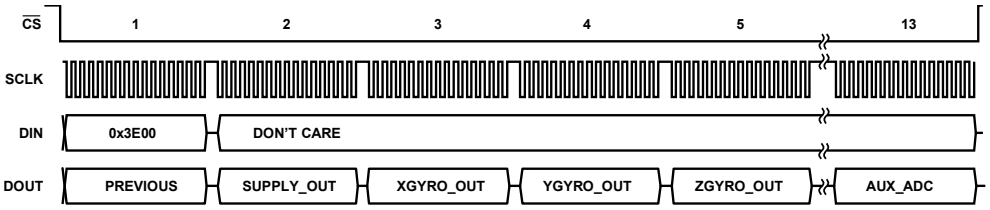


Figure 8.7: Burst read mode of the IMU. Figure from [Analog Devices, 2009, p. 11].

8.6 Actuator control

This section describes how the actuator software control will be designed. First the actuators functionality is described and then the final design is described.

There are four actuators that can control the physical response of the AUV, these are controls for pitch, yaw, surge and heave. The pitch and yaw is controlled by two servos that moves the pitch control surface and rudder angle. The surge direction and speed is controlled with a DC-motor that is driven by a motor driver which is controlled like the servos although there is no feedback.

The pump is controlled by relays configured in an H-bridge, so the pump can act in both directions and be stopped. The pitch control is disregarded in this project and is therefore not of interest.

There have to be designed four functions to easily control these actuators using manageable values. These values has been decided to be relative values as percentages and angles in either positive or negative direction.

Rudder angle

It is desired to make a function that can convert a given rudder angle to a value setting the PWM duty cycle register in the microcontroller, so it is easy to control the rudder angle.

The PWM values and its corresponding rudder angles are listed in table 8.2.

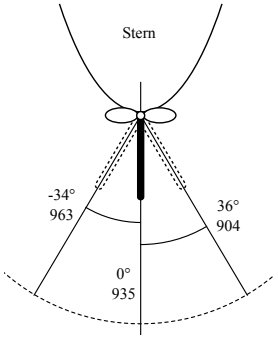


Figure 8.8: Illustration of the rudder angle definition.

Direction	Angle	PWM Value	Duty cycle
Left	-34°	964/1023	5.8 %
Centre	0°	935/1023	8.6 %
Right	36°	904/1023	11.6 %

Table 8.2: Data for the rudders physical servo positions, see figure 8.8.

To make a function converting an input angle to the correct PWM value, it is estimated that the rudder angle is a linear function of the PWM value, and therefore a linear regression is used for the calculation as shown in equations 8.1:

$$f(x) = a \cdot x + b \ , \ a = \frac{y_1 - y_2}{x_1 - y_2} \ , \ b = y_1 - a \cdot x_1 \tag{8.1}$$

Here, x_1 and x_2 represents two of the known input values where y_1 and y_2 are their corresponding output values.

With the values from table 8.2 inserted, the resulting function looks as in equation 8.2.

$$\text{rudder}(x) = \frac{6544 - 6x}{7} \tag{8.2}$$

Motor speed

To control the speed of the DC-motor driving the propeller, a function setting the correct PWM value according to a given percentage is designed. The function will take positive and negative percentages dependent on the desired direction.

The PWM values and its according motor speeds are listed in table 8.3 on the next page.

Direction	Speed	PWM Value	Duty cycle
Forward	100 %	910/1023	11.0 %
Stop	0 %	935/1023	8.6 %
Backward	-100 %	960/1023	6.2 %

Table 8.3: Data for the propellers direction.

As with the rudder function, this function can be determined using linear regression. The resulting function looks as in equation 8.3

$$\text{thrust}(x) = 935 - \frac{x}{4} \tag{8.3}$$

Pump direction

As mentioned earlier the pumps is controlled with an H-bridge, so the motor for the pump can change direction or stop. The hardware is designed to operate two transistors that each control a single pole double throw-relay, so there is two Input Output (IO) pins to control the pump. The function has to control the two IO pins as shown i table 8.4.

	Up	Down	Stop
IO1	1	0	0
IO2	0	1	0

Table 8.4: Logic table for IO to the pump driver.

8.7 Control algorithms

When designing the control algorithms there are things that needs to be considered.

Error sources

To design a stable controller it is important to take sources for errors into account, as certain parts of the system are hardly allowed any room for error. For example an offset in the speed sensor that is not dealt with will make the AUV move when it is supposed to stay at a fixed location or move with a wrong speed and distance. A proportional error in the speed sensor will make the AUV scale it's route with the same factor.

As an IMU is used in this project as speed sensor a large error is to be expected. As the error lays in the sensor it is simple to swap the speed input

from the IMU to a more reliable sensor when such a sensor is acquired, and there by enhancing the AUV's capabilities.

When taking the the IMU's errors into account it is acceptable to make a controller with an amount of errors, as long as the error is due to the lack of an absolute speed measuring device, which are expected to measure the speed more precisely.

Design of control algorithms

In this project the only available speed sensor is the integrated data from the IMU. To get the speed from the IMU the output of the accelerometer in the forward direction is integrated, thereby calculating the speed of the AUV. However the control algorithm developed in 7.2 on page 54 takes the travelled distance as input and not the speed; therefore the integrated accelerometer output has to be integrated once more which yields the distance.

8.8 Ground station software

As of requirement 4, described in the requirement specification in chapter 2.1 on page 9, a ground station is needed to take care of the communication with the AUV. The essential communication consists of exchanging log files from the last dive and new waypoints for the next dive. From this it would be convenient to be able to remote control the AUV manually for testing purposes.

To make the ground station software user friendly, it is chosen to use a GUI to control the functions rather than a command-line based tool. The software itself is programmed in the object-oriented programming language Java, which is cross-platform.

An overview of the basic functions required from the ground station software is shown on figure 8.9 on the facing page and shows the basic blocks with the serial communication and the read and write functions. Furthermore an option to manually control the AUV is shown so the user has the ability to manually control the AUV for testing purposes.

Data format

The data communication will consist of sampled data. After each line, the receiver sends back the data to verify the data is transferred correctly. If an error is detected, the data will be retransmitted. This operation is shown on figure 8.10 on the next page.

If the data size becomes too large, it could be considered to save bandwidth by using a checksum to check the data integrity. Data compression could as well be considered an option to make the transfers faster.

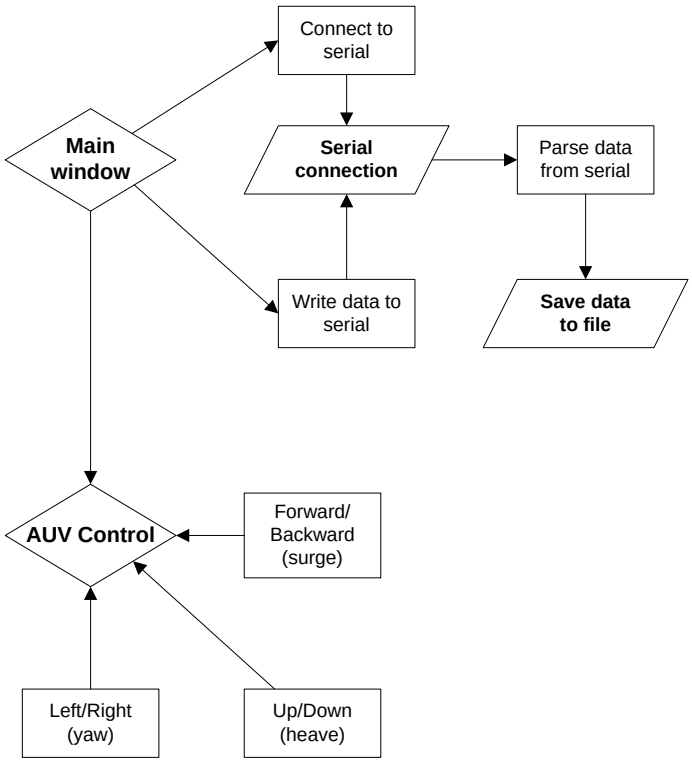


Figure 8.9: Block overview over the ground station software.

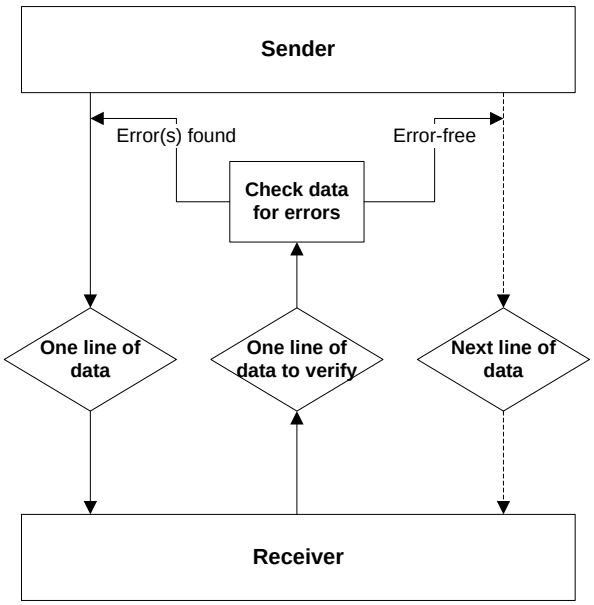


Figure 8.10: Block diagram of data

Part III

Implementation

This part documents the implementation of the design.

9 Software implementation

This chapter describes the implementation of the software used on the AUV. All software used on the AUV is implemented on the ARM7 development board. The chapter is divided into algorithm implementation, software on the AUV and ground station described by flowcharts.

9.1 Algorithm implementation

To go from a transfer function $H(s)$ to a useful algorithm that can be implemented, a conversion is needed. This section is used to describe the transformation, and is based upon the forward motion of the AUV, but the method used is the same for the rest of the controllers. The PID-controller was in the forward case, given as a PD-controller with a P-value of 50, and a D-value of 5. A general expression can be calculated for the controller which is done throughout this section.

$$H(s) = K_\mathrm{p}\left(1 + K_\mathrm{d}s + K_\mathrm{i}\frac{1}{s}\right) \tag{9.1}$$

The transfer function is then z-transformed to make it time discrete. This is done via Tustins approximation, found in [Gene F. Franklin, 2009, p. 587], that replaces every s in the transfer function with:

$$s = \frac{2}{T}\left(\frac{1 - z^{-1}}{1 + z^{-1}}\right) \tag{9.2}$$

Where the T is the sampling time (as the controlling algorithm is executed at 2 Hz, so this is set to $frac12$). When inserted in the above, the z-transform of the system becomes:

$$H(z) = \frac{0.02(50z^2 - 50 + 8K_\mathrm{d}z^2 + 16K_\mathrm{d}z + 8K_\mathrm{d} + 25K_\mathrm{i}z^2 - 50K_\mathrm{i}z + 25K_\mathrm{i})K_\mathrm{p}}{z^2 - 1} \tag{9.3}$$

The signal is then delayed by the highest order of z (in this case z^{-2}) which transforms the signal to:

$$H(z) = \frac{0.02(50z^2 - 50 + 8K_\mathrm{d}z^2 + 16K_\mathrm{d}z + 8K_\mathrm{d} + 25K_\mathrm{i}z^2 - 50K_\mathrm{i}z + 25K_\mathrm{i})K_\mathrm{p}}{z^2 - 1} \tag{9.4}$$

The transfer function is given as the output divided by the input, so rearranging the terms yields:

$$H(z) = \frac{Y(z)}{X(z)} \tag{9.5}$$

Which equates to the following:

$$Y(z)(1-z^{-2}) = \tag{9.6}$$
$$X(z)(0.02(50-50z^{-2}+8K_{\mathrm{d}}+16K_{\mathrm{d}}z^{-1}+8K_{\mathrm{d}}z^{-2}+25K_{\mathrm{i}}- \tag{9.7}$$
$$50K_{\mathrm{i}}z^{-1}+25K_{\mathrm{i}}z^{-2})K_{\mathrm{p}}) \tag{9.8}$$

This is then inverse z-transformed and isolated for Y(z) as this expresses the output as a function of the input.

$$y[n] = K_{\mathrm{p}}x[n] - K_{\mathrm{p}}x[n-2] + 0.16K_{\mathrm{p}}K_{\mathrm{d}}x[n]+ \tag{9.9}$$
$$0.32K_{\mathrm{d}}K_{\mathrm{p}}x[n-1] + 0.16K_{\mathrm{p}}K_{\mathrm{d}}x[n-2]+ \tag{9.10}$$
$$0.5K_{\mathrm{i}}K_{\mathrm{p}}x[n] - K_{\mathrm{i}}K_{\mathrm{p}}x[n-1] + 25K_{\mathrm{i}}K_{\mathrm{p}}x[n-2] + y[n-2] \tag{9.11}$$
$$\tag{9.12}$$

When the values calculated for the PID-controllers calculated earlier is inserted, the algorithm becomes:

$$y[n] = 90x[n] + 10x[n-1] - y[n-1] \tag{9.13}$$

The equation 9.13 is then used to design an algorithm, that controls the forward motion of the AUV. The same method is used for the other two algorithms. The $y[n]$ describes the output (i.e. the input to the DC-motor) and $x[n]$ describes the input to the system, which is the reference minus the measured data. This ensures that the PID-controller only acts on the signal that is left over from the reference and therefore tells whether the DC-motor should thrust for- or backward.

The exact same methods are used to calculate the algorithms for the two other controllers used aboard the AUV. For the up and down case the transfer function is given as:

$$H(z) = 20\left(1 + 0.05\frac{1}{\frac{2}{T}\left(\frac{1-z^{-1}}{1+z^{-1}}\right)}\right) \tag{9.14}$$

This is then inserted into equation 9.9 and the following is obtained:

$$y[n] = 26.25x[n] - 13.75x[n-1] + y[n-1] \tag{9.15}$$

The last algorithm to design is the algorithm that controls the rudder, and thereby the angle of the AUV. Again, the exact same method is used as before. The transfer function of the controller that turns the AUV is given as:

$$H(z) = 20(1 + \frac{2}{T}\left(\frac{1-z^{-1}}{1+z^{-1}}\right) + 3\frac{1}{\frac{2}{T}\left(\frac{1-z^{-1}}{1+z^{-1}}\right)}) \tag{9.16}$$

And when converted, the algorithm becomes:

$$y[n] = 83.2x[n] + 113.6x[n-1] + 43.2x[n-2] + y[n-2] \tag{9.17}$$

This concludes the algorithm design which can be implemented as C-code.

Actuator and sensor subroutines

In this section, the software blocks derived from the software structure in section 8.3 on page 68 are shown as functions using flowcharts.

The burst read function described in figure 9.1 works by sending the address 0x3e00 to the IMU which sequentially reads out measurements from all sensors in the IMU seen on figure 8.7 on page 71.

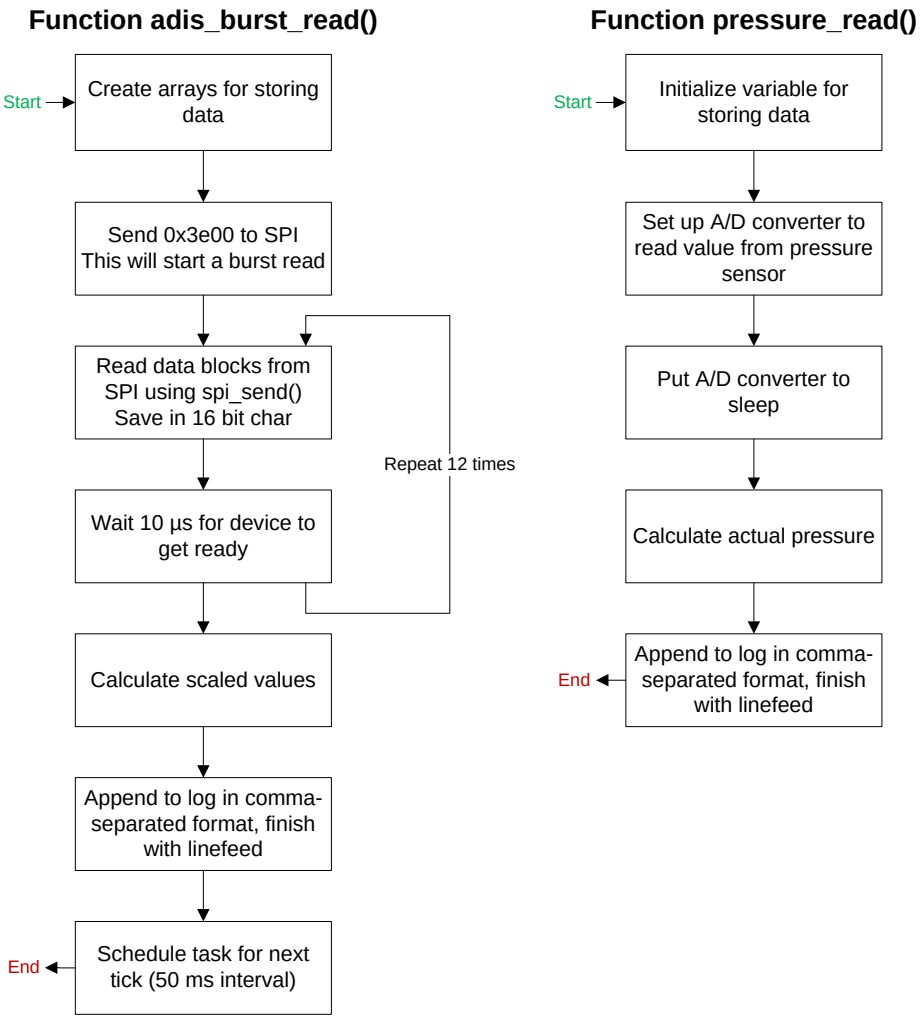


Figure 9.1: Basic sensor readout functions.

The simple actuator and sensor related functions are shown on figure 9.1 and 9.2 on the following page. The actuator related functions take a valid value as input, activates the hardware and returns no value. The sensor related functions need no input and returns an array containing respectively x, y and z-axis values as index 0, 1 and 2.

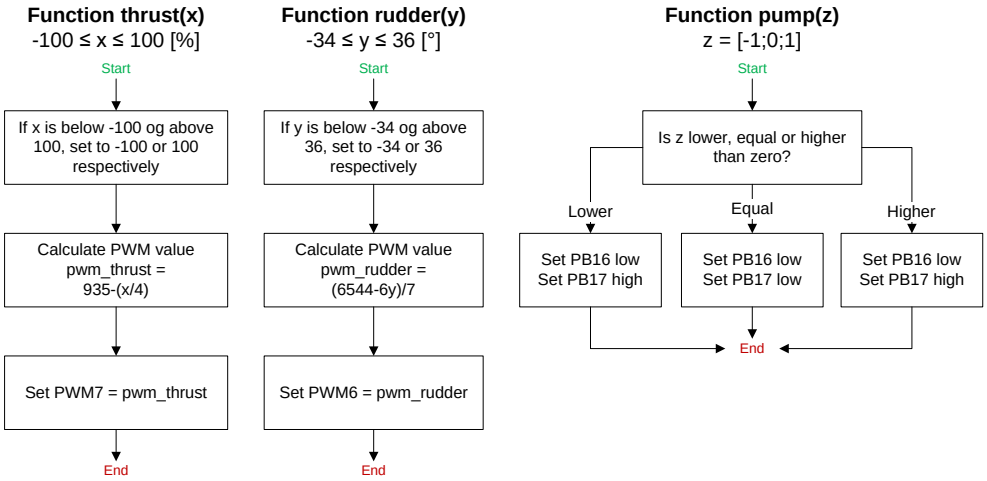


Figure 9.2: Basic actuator functions.

9.2 Route planning and control

This section describes the design of the software to manage way points and set the control algorithms to the correct reference destinations.

The navigation requires a route planner to set the correct destinations and from the given set of waypoints calculate distances and relative coordinates. These data are needed to the controllers algorithms to sail to the correct destinations. Such a route planner is shown on figure 9.3 which shows the principle used in this application. The route planner sequentially runs through each of the way points and for each way point lets the controllers regulate the actuators until the AUV is at the destination, whereafter the next way point is set as destination.

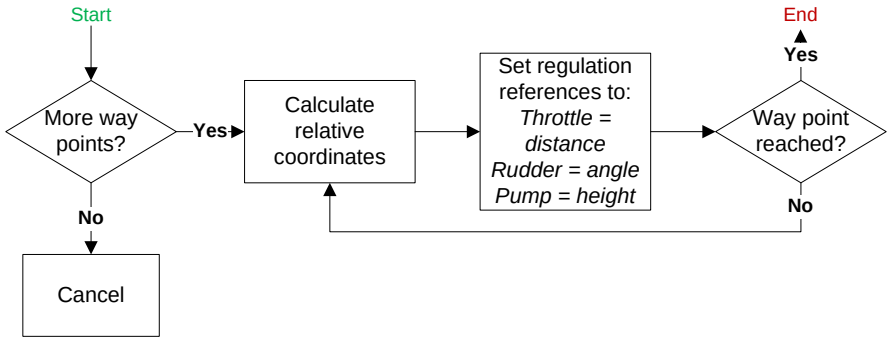


Figure 9.3: Flowchart showing the route planners operation.

The PID-controller's flowchart is shown on figure 9.4 on the facing page. The error is the difference between the reference and the feedback loop. The

feedback is a measurement from the IMU and fetched from a storage variable that is updated by another function. If the error is within a margin of the reference signal the PID-controller function is terminated. The output is then calculated and the signal generator is called with the new value.

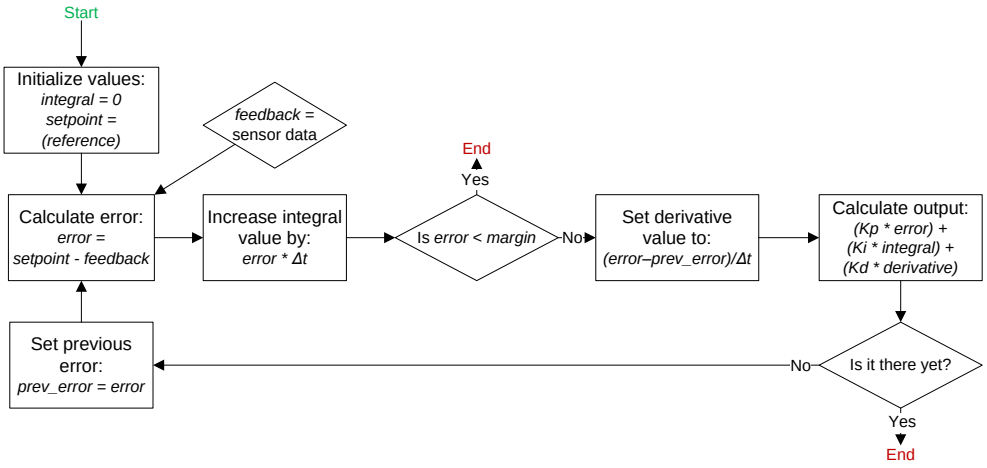


Figure 9.4: Flowchart over a PID-controller.

9.3 Ground station software

The software running on the ground station is programmed in Java. The UML model of the ground station software is seen on figure 9.5 which implies the SerialComm class. This class uses the RXTX Library for Java [Various Authors, 2011]. The classes and their functions will be described below:

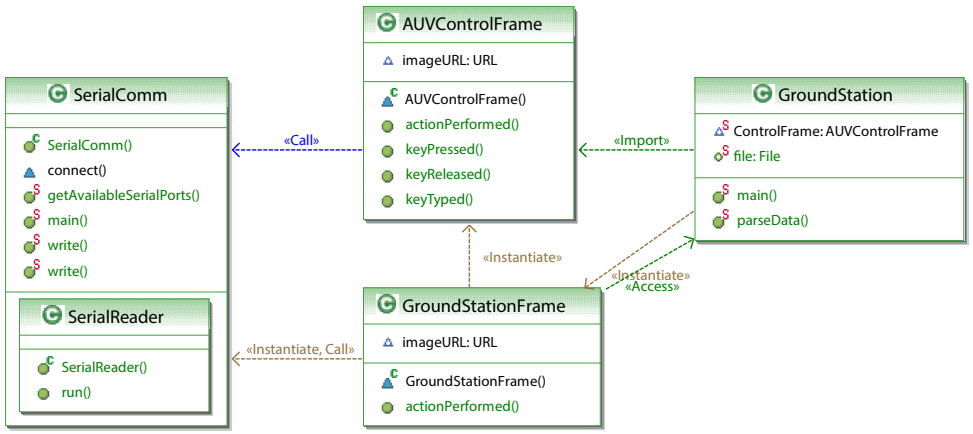


Figure 9.5: UML model of ground station software

SerialComm This class takes care of the serial communication. It contains functions to list all the available ports, then make a connection to the desired port and write data to the connected serial port.
Also, a subclass SerialReader reads all the data received on the connected serial port. This subclass is running in a separate thread parallel with the other functions in the software. The SerialComm class itself is based on the examples in the RXTX Wiki [Various Authors, 2011] which has been modified to suit this project.

GroundStationFrame This class draws the main window on the screen. It has some buttons and an action handler to capture the actions from the buttons.

AUVControlFrame This class draws the manual control panel on the screen. It features an action handler and also event handlers for key inputs. This allows the user to control the AUV by using the keyboard.

GroundStation This class is the main class of the program. It instantiates the GroundStationFrame class and incorporates a function to parse the data read from the serial port.

Hardware implementation 10

This chapter describes how the hardware is implemented, but ignores digital connections that are connected directly to the microcontroller.

10.1 Pump driver

The pump consists of a DC-motor that drives a peristaltic pump. This pump is geared down compared to how much water it pumps and the rotations of motor. The AUV uses two relays configured in an H-bridge to control the pump.

The implemented H-bridge configuration is shown on figure 10.1.

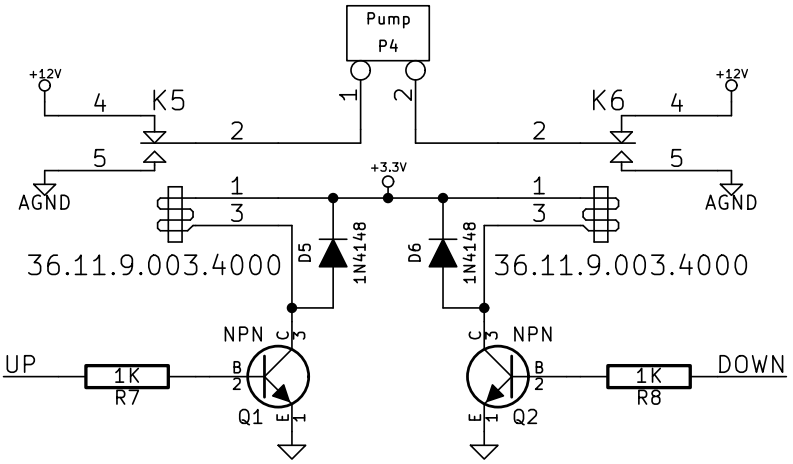


Figure 10.1: Schematic of the pump control.

10.2 Manometer

The manometer used to determine the depth of the AUV is a Honeywell SSCSNBN030PAAA5. As mentioned in chapter 3 on page 13 the manometer needs a resolution of 0.01 bar. Therefore the resolution and precision of the sensor is calculated.

Resolution

The sensor has an operating point between 10 % and 90 % of the supply voltage. This means that the difference between P_{min} and P_{max} is 80 % of the applied voltage. Therefore the operating point spans 4 V. The P_{max} at 4 V in bar equals 2.0684 bar.

Since the sensor operates through the ADC which has a resolution of 12 bits which equals:

$$\frac{2.0684}{4096} = 0.0005\frac{\text{bar}}{\text{bit}} \tag{10.1}$$

Precision

The sensor has a ±2 % precision from the measurement. These 2 % of the 2.0684 bar equals 0.041368 bar. This error corresponds to $\frac{\text{resolution}}{\text{error}}$. This is calculated in equation 10.2.

$$\frac{0.041368}{0.0005} \approx 83 \text{ bit} \tag{10.2}$$

Assuming a sufficient controller for the vertical motion of the AUV is designed, this enables the AUV to maintain a position within ±0.41 m vertically.

Part IV

Summation

This part concludes the report with an acceptance test, conclusion and discussion.

Acceptance testing 11

This chapter gives an overview of whether the requirements are fulfilled. It sums up the requirements, lists if they are failed or passed, while stating the reason.

The acceptance testing is done according to the specified tests in section 2.2 on page 10. As the control algorithms have not been fully implemented in the AUV, the tests described in this chapter are based on the dynamic models derived in chapter 6 on page 33. The tests are performed by verifying that the dynamic models fulfil the requirements. It is only the model of the for- and backward motion which is implemented, and therefore it is only that one the acceptance tests reflects.

Test 1a: Move for- or backward with ±1 m margins √
Test is passed
As seen in `/acceptance_testing/test1a.pdf`, which illustrates the resulting position, speed and acceleration from a input step of 10 m, the AUV reaches the 10 m point within ±1 m, and the test is therefore passed.

Test 1b: Turn left or right with a maximum turning radius of 1 m ╳
Test is failed
The test have not been executed since the AUV did not meet a sufficient functionality level.

Test 1c: Move vertically up or down with ±1 m margins ╳
Test is failed
The test have not been executed since the AUV did not meet a sufficient functionality level.

Test 1d: Move up or down while moving forward with ±1 m margins ╳
Test is failed
The test have not been executed since the AUV did not meet a sufficient functionality level.

Test 1e: Maintain given position within ±1 m margins ╳
Test is failed
The test have not been executed since the AUV did not meet a sufficient functionality level.

Test 1f: Reach way point within a radius of 1 m ╳
Test is failed
The test have not been executed since the AUV did not meet a sufficient functionality level.

Test 2a: Measure velocity with an accuracy of 10 % ╳
Test is failed
The test is failed since the AUV uses the IMU as speed sensor and this does not meet the 10 % accuracy.

Test 2b: Measure depth with an accuracy of ±0.1 m ╳
Test is failed
The test have not been executed since the AUV did not meet a sufficient functionality level.

Test 2c: Measure the direction relative to the Earths magnetic field (compass) with an accuracy of ±2.5° ╳
Test is failed
The test have not been executed since the AUV did not meet a sufficient functionality level.

Test 2d: Measure battery voltage with an accuracy of ±0.01 V ╳
Test is failed
The test have not been executed since the AUV did not meet a sufficient functionality level.

Test 3: Must have an emergency routine so the AUV surfaces if the battery level becomes critical ╳
Test is failed
The test have not been executed since the AUV did not meet a sufficient functionality level.

Test 4: Must be able to wirelessly send logfiles to a ground station and receive a new route within 100 metres line of sight (√)
Test is partly passed
This test can not be executed for logfiles, but as the communication between the AUV and ground station is functional when sending live read outs from the sensors, the test is partly passed.

Test 5: Rise to the water surface to communicate with the ground station wirelessly to recieve new waypoints ╳
Test is failed
The test have not been executed since the AUV did not meet a sufficient functionality level.

Subconclusion

As not all of the models nor the control algorithms have been fully functional at the time of testing, only the for- and backward model could be verified. Also, the communication between the AUV and ground station is verified, but only using current data read outs and not with log files.

Conclusion and future perspective 12

This chapter will summarize the report by discussing what went right and what still needs further development, as well as discussing future applications for the product and technologies developed in this report.

12.1 Conclusion

This report documents the development of an AUV, from the preliminary investigations of the stock submersible used as basis for the AUV, to the final product.

The product is based on a stock submersible with modified electronics. The electronics are based on an ARM7 development board with a shield developed for this project. The most important part of the shield is an IMU that measures acceleration, rate of turn and magnetism in three dimensions. Furthermore the IMU is fitted with a temperature sensor for internal calibration of the sensors, and an ADC to which a pressure sensor has been connected for depth measurements. All these sensors were intended to be utilized by control algorithms to make the AUV follow a predetermined route of way points; however this functionality has not been fully implemented. PID controllers have been developed for for- and backward motion, up- and downward motion and left and right turn. The controller for for- and backward motion has been implemented, though it has not been calibrated to a useful degree; the rest of the controllers have been implemented in the model and not on the actual AUV.

12.2 Future perspective

This section looks at future perspectives and applications for the technology developed in this project.

As this project is only a prototype the practical applications are fairly limited, but a fully functional AUV would be usable in a lot of situations, all kinds of under water surveying, mapping and inspections of e.g. marine life, water composition, mapping of ocean floors, inspection of gas/oil pipe lines and structural integrity of oil rigs and large ships.

To fulfil the previously mentioned functionality the AUV will need upgrades in form of:

A GPS receiver to calibrate the AUV's perception of its location.

A communication system with greater range is needed if the AUV is to communicate with a ground station when at sea, a satellite communication system would be evident to investigate.

A more sturdy frame to enable the AUV to go to greater sea depths to be relevant in a sea research context.

Batteries with greater capacity to allow the AUV to operate for longer periods of time without human interaction.

Alternative propulsion system to operate for longer periods of time a propulsion system like the system used by [Washington State University, 2011] would be very useful to reduce power consumption.

A payload system to mount a mission specific payload, to make the AUV more versatile.

Part V

Appendix

This part includes chapters which are important for the project, but not necessarily interesting to the reader of the report.

Measurements on the AUV using rulers

A

A.1 Purpose

The purpose of this journal is to define limitations caused by the stock submersible as well as measurements for modelling theAUV. For a sufficient the following will be analyzed:

- Turning radius
- Diving angle
- Surging speed
- Rate of turn

A.2 Tools

Type	AAU no.	Manufacturer	Model
2x Plastic rulers	-	-	-
Stopwatch	-	-	-

Table A.1: Tools used for this measurement.

A.3 Measurement procedure

All the tests are performed in a pool, so external disturbances are kept at a minimum. The four specifications that needs to be determined are split into four tests:

1. **Turning radius**
   Place a fixed ruler in the water surface, so theAUV can manouvre in the water under it. Place theAUV ortogonal to the ruler, so theAUV can turn at its maximum to the right. It is noted where theAUV passes the ruler. After it has passed the ruler, the AUV continues to turn as much as possible. When it comes back to the starting point, it is noted where it is. This will a minumum radius of turn.

2. **Diving angle**
   Place a fixed ruler in the warer surface, and place another ruler vertical in the water. TheAUV sails along the surface until it dives into the water with the maximum possible angle. Then it is possible to note the depth at which the AUV is, this will construct a triangle from which the angle of diving can be calculated.

3. **Surging speed**
   To measure the speed of surging. A fixed ruler is placed in the water surface. Place theAUV below the ruler at the starting point. Set theAUV to surge at max speed and measure the time over a predetermined distance. From this the surging speed can be calculated.

4. **Rate of turn**
   Measuring the rate of turn can be done by following test 1, "Radius of turning". TheAUV maintains a fixed radius of turning and is sailed around the ruler 4 times, and the time is noted, from where the average rate can be calculated.

A.4 Theory

In this journal, a few simple formulas are needed. These formulas are been used to calculate the respective limitations from each test.

Radius of turning

To calculate the radius of turning, measure the diameter and divide it by two.

Diving angle

To measure the angle of diving the setup illustrated by A.2 on the next page is used:

$$\tan(A) = \frac{a}{b} \Rightarrow \mathrm{A} = \tan^{-1}\left(\frac{a}{b}\right) \tag{A.1}$$

where

A = the angle at which theAUV can dive.
a = the ruler measurement of the ruler that measures the depth of the AUV.
b = the measurement of the ruler in the horizontal direction.

Surgning speed

To calculate the surgning speed the setup illustrated on figure A.3 on the facing page is used. From this the speed can be calculated.

Rate of turn

To measure the rate of turn, the same as in A.1 on the next page can be used. If a more precise measurement is desired, an average can be calculated.

A.5 Measuring arrangement

1. **Turning radius**

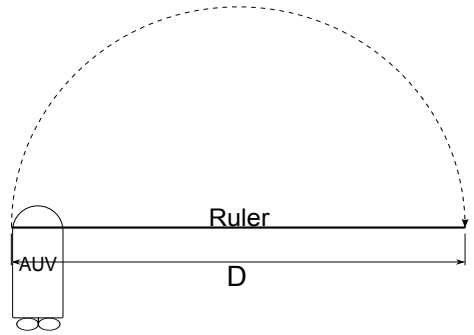


Figure A.1: Setup of test 1

2. **Diving angle**

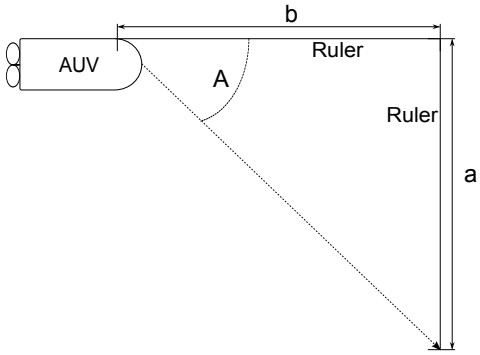


Figure A.2: Setup of test 2

3. **Surging speed**

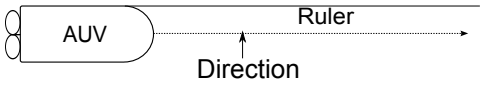


Figure A.3: Setup of test 3

4. **Rate of turn**

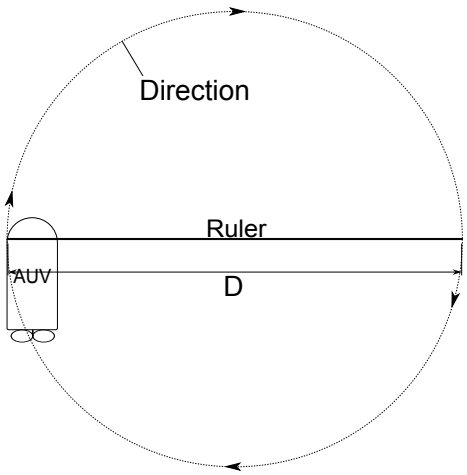


Figure A.4: Setup of test 4

A.6 Results

1. **Turning radius**
   There were made four measurements, which ended up with a average radius of turning at $\frac{1.08}{2} = 0.54$ m.

2. **Diving angle**

   1 b=4.25 m, a=1.15 m, $A = \tan^{-1}\left(\frac{1.15}{4.25}\right) = 15.141°$

   2 b=5.50 m, a=1.17 m, $A = \tan^{-1}\left(\frac{1.17}{5.50}\right) = 12.68°$

   3 b=4.10 m, a=1.20 m, $A = \tan^{-1}\left(\frac{1.20}{4.10}\right) = 16.314°$

   4 b=6.35 m, a=1.16 m, $A = \tan^{-1}\left(\frac{1.16}{6.35}\right) = 10.352°$

3. **Surging speed**
   The measurement of the surgning speed were performed five times, over a fixed distance of 2 m. The test gave the following results:

   1 3.94 s. This gives a speed of surging at: $\frac{2}{3.94} = 0.508\frac{m}{s}$

   2 4.17 s. This gives a speed of surging at: $\frac{2}{4.17} = 0.48\frac{m}{s}$

   3 3.96 s. This gives a speed of surging at: $\frac{2}{3.96} = 0.505\frac{m}{s}$

   4 3.93 s. This gives a speed of surging at: $\frac{2}{3.93} = 0.509\frac{m}{s}$

   5 3.93 s. This gives a speed of surging at: $\frac{2}{3.93} = 0.509\frac{m}{s}$

   This makes an average speed of: $\frac{0.508+0.48+0.505+0.509+0.509}{5} = 0.5022\frac{m}{s}$. From this the standard deviation can be calculated to see if the deviation is too high.

To calculate the standard deviation, first calculate the difference of each measurement from the mean, and square the result:

- $(0.508 - 0.5022)^2 = 0.00003364\frac{m}{s}$
- $(0.48 - 0.5022)^2 = 0.00049284\frac{m}{s}$
- $(0.505 - 0.5022)^2 = 0.00000784\frac{m}{s}$
- $(0.509 - 0.5022)^2 = 0.00004624\frac{m}{s}$
- $(0.509 - 0.5022)^2 = 0.00004624\frac{m}{s}$

$\sqrt{\frac{0.3364e{-}4{+}0.49284e{-}3{+}0.784e{-}5{+}0.4624e{-}4{+}0.4624e{-}4}{5}} = 0.0112\frac{m}{s}$

4. **Rate of turn**
   This test was performed once to the right and once to the left.

   a) The right turn measured to 43.99 s

   b) The left turn measured to 44.17 s

A.7 Errors

There have been a few sources for errors in measuring these specifications. There were other people in the swimmingpool when theAUV was tested. This could induce measurement errors and therefore the tests were performed more than once.

Measurements with the AUV using sensors

B

B.1 Purpose

The purposes of this measurement journal is to verify the model of the AUV, and to see if the sensor meets the expectations.

The parameters to measure are as follows:

- Accelerometer and gyroscope output during:
  - Full forward throttle from full stop
  - Full backward throttle from full stop
  - Half forward throttle from full stop (Approximated because of manual control)
  - Half backward throttle from full stop (Approximated because of manual control)
  - Full backward throttle while at top speed
  - Turning left at maximum speed
  - Turning right at maximum speed
  - Diving down from full stop
  - Diving up from full stop
  - Diving up and down during full speed forward
- Magnetometer output
- Manometer output

B.2 Tools

Type	AAU no.	Manufacturer	Model
Cable	-	-	-
Stopwatch	-	-	-
Large pool	-	-	-

Table B.1: Tools used for this measurement.

B.3 Measurement procedure

All the tests are preformed in a pool so external disturbances kept at a minimum. During all the tests the AUV is supplied with power from an external

powersupply and the data will be transmitted directly to a computer using a cabled RS-232 connection. The pool where the measurements were made, was not deep enough to perform the tests that includes diving. Therefore these tests were omitted. The specifications that needs to be determined are split into these tests:

1. **Full forward throttle from full stop**
   The AUV is placed under water, standing still, and then full forward throttle will be applied without turning . The test lasts until the accelerometer shows forward acceleration below 0.1 G. The test is assumed to show measurements like those shown on figure B.1, where it is assumed to get an acceleration of approximately $0.125\frac{m}{s^2}$ which gives a velocity at approximately $0.5\frac{m}{s}$ due to the fricition generated by the water.

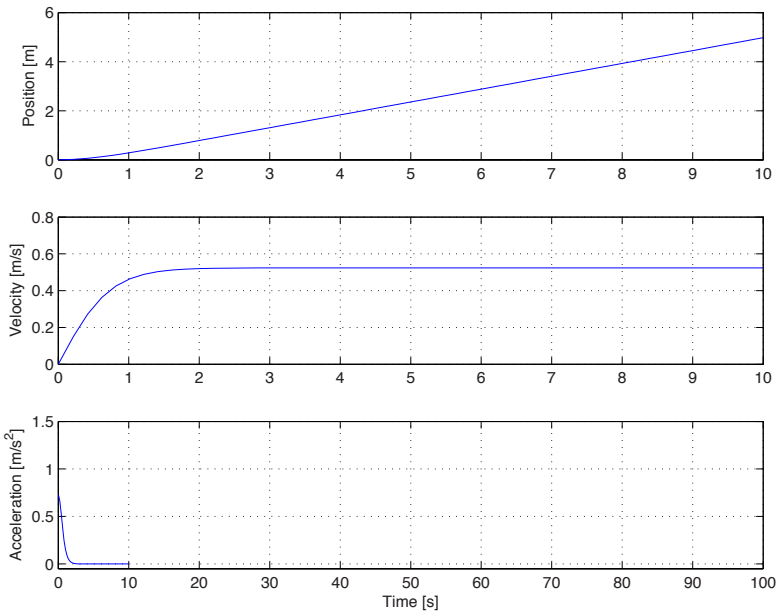


Figure B.1: The simulation of the AUV moving through the water.

2. **Full backward throttle from full stop**
   The AUV is placed under water, standing still, then full backward throttle will be applied without turning . The test lasts until the accelerometer shows backward acceleration below 0.1 G. Like the first measurement it is expected to get acceleration and velocity, though smaller than shown in the first measurement.

3. **Half forward throttle from full stop**
   The AUV is placed under water, standing still, approximately 50 %

forward throttle will be applied without turning . The test lasts until the accelerometer shows forward acceleration below 0.1 G.

4. **Half backward throttle from full stop**
   The AUV is placed under water, standing still, approximately 50 % backward throttle will be applied without turning . The test lasts until the accelerometer shows backward acceleration below 0.1 G.

5. **Full backward throttle while at top speed**
   The AUV is placed under water and accelerated to full forward speed. No turning is allowed during this test. When full speed is reached (the accelerometer shows acceleration below 0.1 G), full backward throttle is applied until the AUV is standing still.

6. **Turning left at maximum speed**
   The AUV is placed under water and accelerated to full forward speed. When full speed is reached (the accelerometer shows acceleration below 0.1 G), the rudder is turned to the leftmost position. When the AUV has turned 90° to the left, the direction is adjusted to a full speed non-turning forward motion.

7. **Turning right at maximum speed**
   The AUV is placed under water and accelerated to full forward speed. When full speed is reached (the accelerometer shows acceleration below 0.1 G), the rudder is turned to the rightmost position. When the AUV has turned 90° to the right, the direction is adjusted to a full speed non-turning forward motion..

B.4 Measurement arrangement

The test setups are shown on B.2

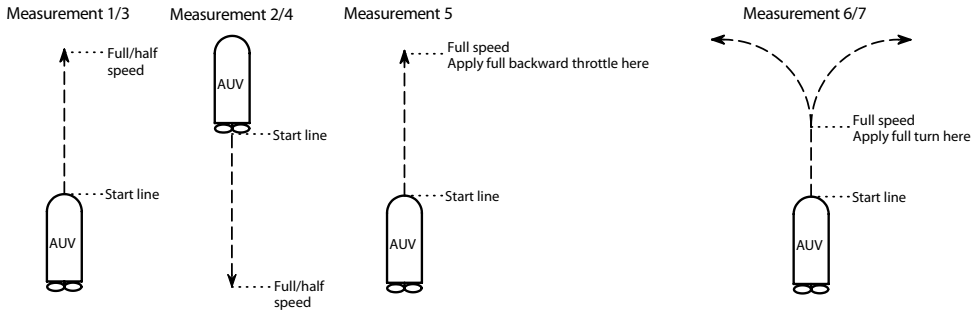


Figure B.2: The arrangement for the first seven measurements.

B.5 Results

The results from the full forward test is shown on figures B.3, B.4 and B.5. The raw data is found in `/measurements/First_IMU_watertest/` The acceleration used for calculations has been filtered with a moving average smoothing using a span of 30 samples and applied with an offset on 0.4 m/s^2. This offset is found by "fiddling" with the numbers. From the results an approximate maximum speed of 0.97 m/s has been reached in 24 seconds. These results are not as expected, as the AUV can not surge faster than 0.75 m/s according to measurements conducted in appendix A.1 on page 95.

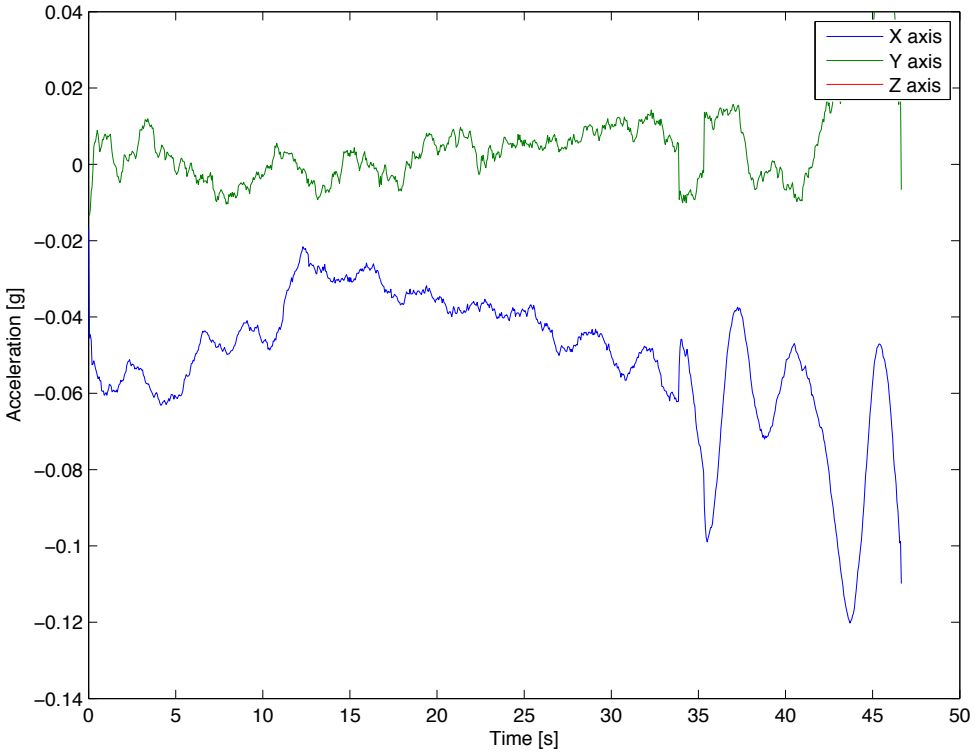


Figure B.3: Accelerometer output during full forward test. Filtered with moving average (span: 30 samples).

Many of the measurements show inconclusive results, as the actual acceleration is very small and hard to distinguish from the movement of the AUV and noise. Another measurement from the full forward test is a bit more useful, as it shows an acceleration offset of -0.58 on the x-axis when standing still in the water. With this offset adjusted, the data will look as in figure B.6 and B.7. The integrated acceleration shows a top speed on 6.9 m/s, which is too fast according to the AUV's specifications.

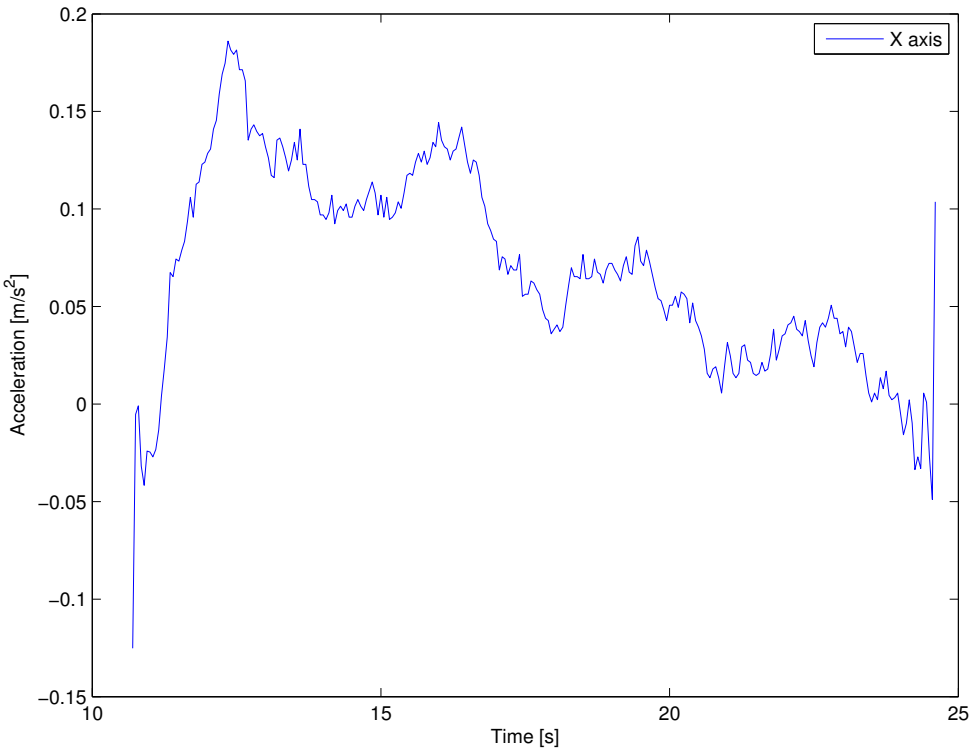


Figure B.4: Accelerometer output during full forward test, zoomed to the accelerating part of the test. Filtered with moving average (span: 30 samples) and applied with a Y-axis offset of 0.4.

B.6 Errors

The accelerometer has to measure small changes in the acceleration of the AUV. Therefore any small disturbance has been measured by the IMU.

- A person was needed to control the AUV.
- The connection to the AUV was made through a RS232 cabel through the top of the AUV. When the AUV surges through the water this cable caused friction in the water.
- If the AUV was not 100 % horizontal the gravitational constant influences the results.
- The pitch can change in relation to the speed.

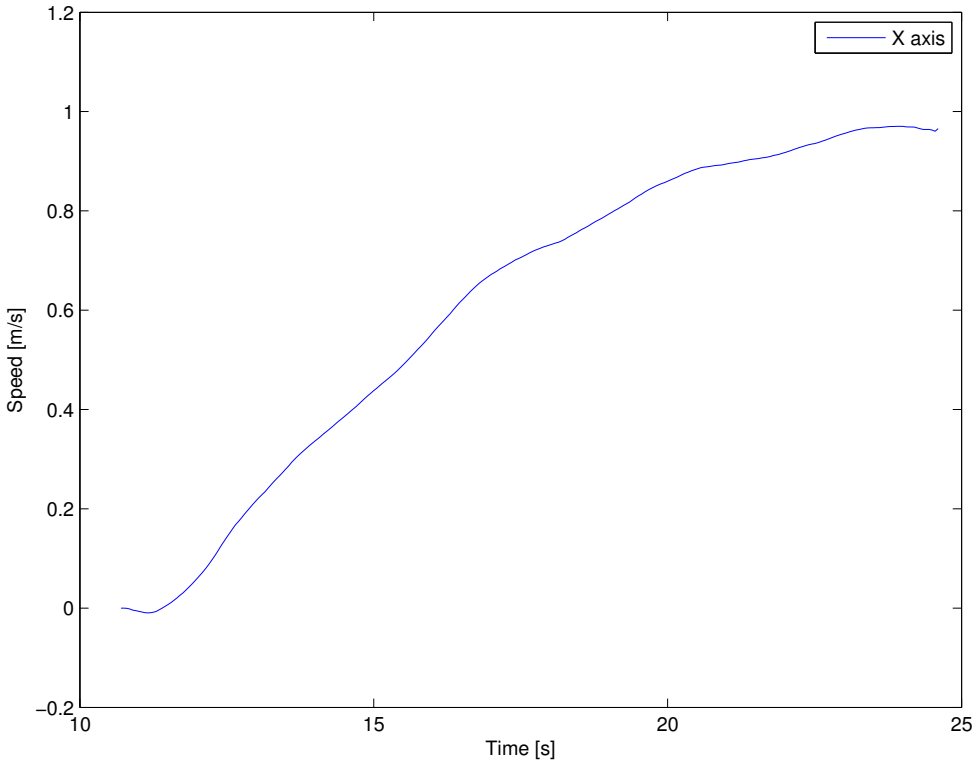


Figure B.5: Integrated accelerometer output showing the speed.

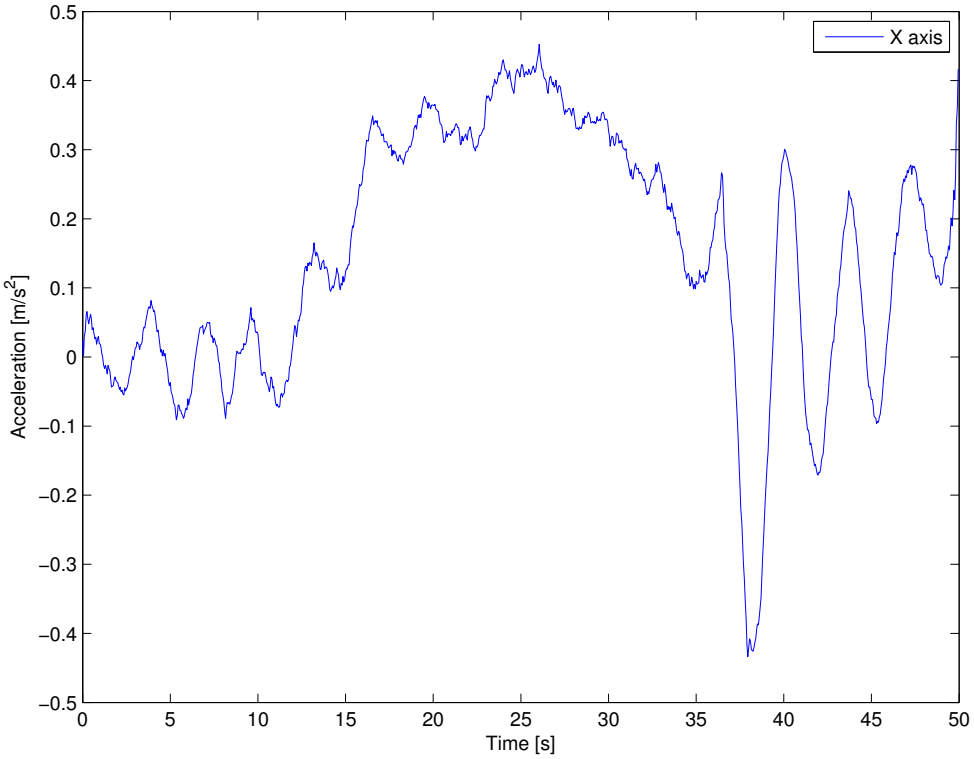


Figure B.6: Accelerometer output during full forward test. Filtered with moving average (span: 30 samples) and applied with a Y-axis offset of 0.58.

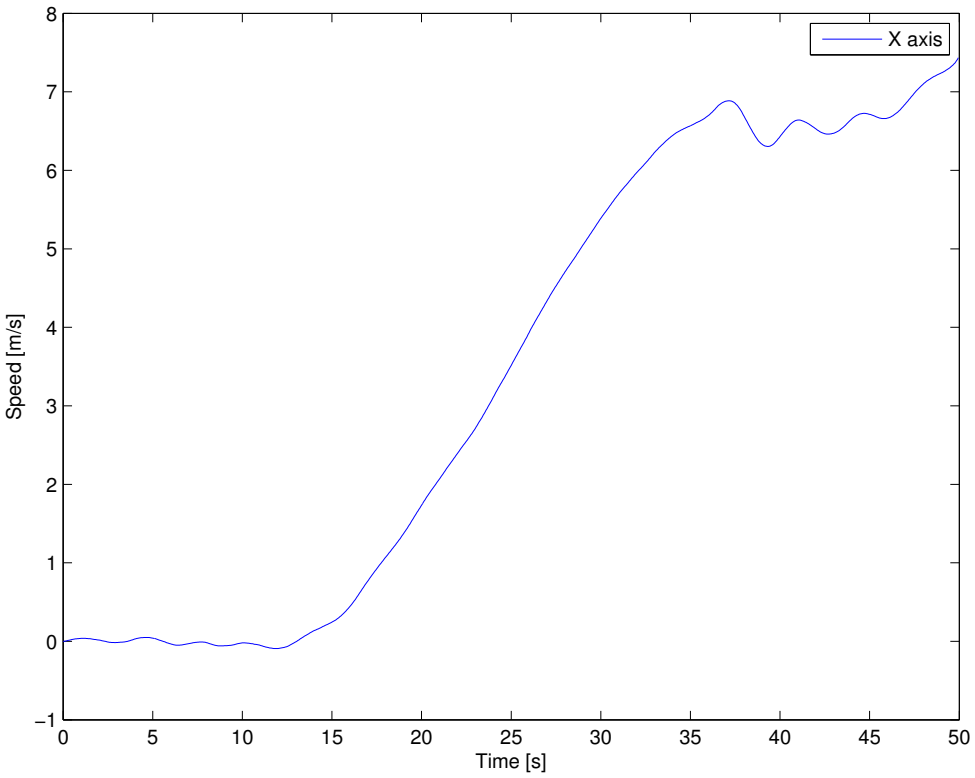


Figure B.7: Integrated accelerometer output showing the speed.

Measurements on the DC-motor

C

C.1 Purpose

The purpose of this journal is to measure the constants for the DC-motor model. It is needed to measure these constants so the model can be implemented in the model for the complete system.

C.2 Tools

Type	AAU no.	Manufacturer	Model
Power supply	52756	Hameg	HM7042-3
Tachometer	-	Compact	A2103/LSR
Ammeter	60760	Fluke	189
Voltmeter	60764	Fluke	189
Oscilloscope	33941	Agilent	54621D

Table C.1: Tools used for this measurement.

C.3 Procedure

The DC-motor is measured directly on the AUV. Several measurements have been made to measure all constants on the DC-motor model needed for the motor model:

- K = the motorconstant
- τ = the time constant
- L_a = motor inductance
- R_a = motor resistance
- b = the viscosity friction
- J = the inertia of the motor

1. **Measuring R_a**
   To measure R_a the motor is prevented from turning. When this is done the circuit measured looks like C.1.

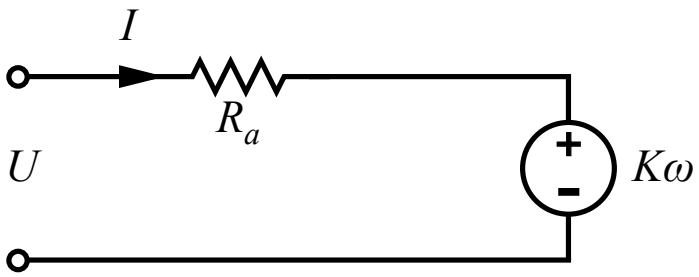


Figure C.1: The circuit while preventing the motor from turning.

Different levels of voltage are applied to the motor and the current is measured. From these measurements a graph can be drawn where the R_a can be found. The graph is made from the relation: $R_a = \frac{U}{I}$ where

R_a = the resistor inside the motor.
U = the voltage applied at the motor.
I = the measured current at the resistor.

2. **Measuring the motor constant**
   By making a similar grapf the motor constant can be found. But in this case the motor is not prevented from turning. Again different levels of voltage will be applied to the motor and the current and revolutions are measured. The revolutions is measured as seen on figure C.2.

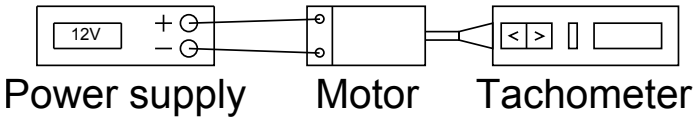


Figure C.2: The way the RPS is measured.

The The motor constant can be expressed as: $K = \frac{U_a - R_a \cdot I}{\omega}$ where

R_a = the resistor inside the motor.
U_a = the voltage applied to the motor.
I = the measured ampere at the resistor.
ω = the rotational acceleration.

3. **Measuring τ**
   To measure τ the charging curve of the inductance is needed. To measure this an additional resistor of a given value is used. This makes it possible to measure the charging curve with an oscilloscope. At 63% τ is read.

4. **L_a**
   To measure L_a; R_a and τ is needed. A transfer function of the motor can

be expressed as a calculation of L_a. This is given as a transfer function where $\tau = \frac{L_a}{R_a}$. From this L_a can be derived.

5. **Calculating the viscosity friction**
   The viscosity friction can be determined by making a graph depending on $K \cdot I$ and ω but evaluated in $\frac{rad}{s}$.

6. **Calculating the moment of inertia**
   All the previous constants are needed to calculate the moment of inertia. The inertia can be derived from the following formula: $\tau = \frac{R_a \cdot J}{R_a \cdot b + K^2}$. where

   b = the viscosity friction.
   J = the inertia of the motor.

Results

1. **R_a**
   An input voltage and the current are measured and expressed as R_a as seen in figur C.3.

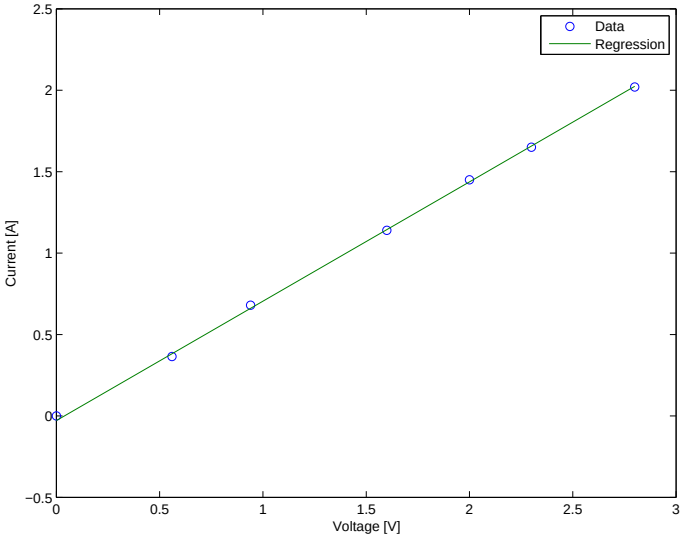


Figure C.3: This figure shows the value of R_a

This gives a value for R_a at 0.6936 Ω.

2. **Motor constant**
   The voltage, current and angular acceleration is measured at different levels. The results of this is:

U [V]	I [A]	ω [RPS]
1.93	0.49	8
4	0.57	23
6.01	0.64	42
8.04	0.67	59.8
10	0.76	77.5
12	0.85	95

The values of ω is converted to $\frac{rad}{s}$ by: $\frac{\text{RPS}}{2 \cdot \Pi}$. Using the measured values in $\text{K} = \frac{\text{U}_\text{a} - \text{R}_\text{a} \cdot \text{I}}{\omega}$ gives six motor constant values. The average value of K is then calculated to be 0.091.

3. τ
   The charging curve of the inductor is measured and can be seen in figur C.4.

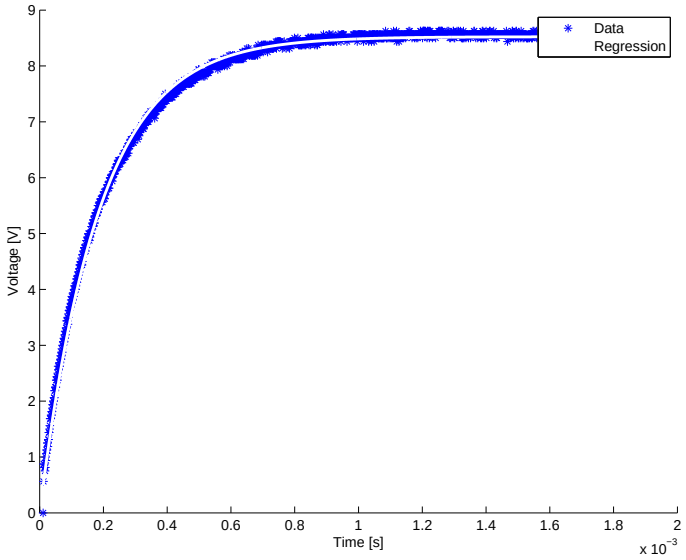


Figure C.4: Charging curve to estimate τ

Three measurements of this charging curve were made. Each of these have been plotted in MATLAB and a curve fitting tool have been utilized to estimate the point at the graph where it has reached steady state. At 63% of the rising curve the τ can be evaluated. The average of these three measurements gave a τ of 0.000195 s.

4. **L_a**
   After R_a and τ are determined L_a can be calculated from $\text{L}_\text{a} = \tau \cdot \text{R}_\text{a}$. This gives a value of L_a at 0.00105 H.

5. **Viscosity friction**
   When evaluating K · I and the ω, a graph seen in C.5 is drawn. The charging curve of the inductor is measured and can be seen on figure C.4.

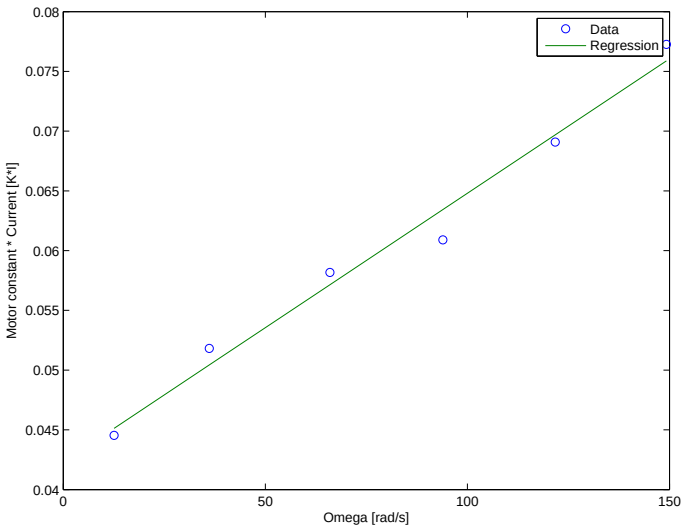


Figure C.5: Friction curve where the slope is the viscosity friction

   The slope of the graph is the value of the viscosity friction. This is evaluated as 0.0002 $\frac{kg}{s \cdot m}$.

6. **Moment of inertia**
   After all the other constants have been determined the moment of inertia can be calculated using $\tau = \frac{\mathrm{R_a \cdot J}}{\mathrm{R_a \cdot b + K^2}}$. This gives an moment of inertia at 0.000002357 $kg \cdot m^2$.

C.4 Summary

- $\mathrm{R_a} = 0.6936\Omega$.
- $\mathrm{K} = 0.091$.
- $\tau = 0.000195$ s.
- $\mathrm{L_a} = 0.00105$ H.
- $b = 0.0002$ $\frac{kg}{s \cdot m}$.
- $J = 0.000002357$ $kg \cdot m^2$.

C.5 Errors

While measuring ω there have been some minor errors due to friction of the tachometer when applied to the motor.

Measurements of force on AUV

D

D.1 Purpose

The purpose of this journal is to measure the maximum force the AUV can deliver. This is needed to verify the model made of the AUV.

D.2 Tools

Type	AAU no.	Manufacturer	Model
Dynamometer	02054	Leybold Didactic	314-181
Scale	-	Kern	CH 15 K 20

Table D.1: Tools used for this measurement.

D.3 Procedure

Two tests have been performed, one with each of the measurement tools. Both tests were performed in a water canal. In both tests the AUV is set to full speed. When using the dynamometer an amount of newton pulled can be measured. Similarly using the scale, an amount in kg can be measured. When measuring the amount in kg a multiplication of 9.82 is needed to get the pulled force in newton.

D.4 Measurement arrangement

1. The measurement using the dynamometer can be seen in figure D.1 where the AUV pulls on the dynamometer. From this an amount of newtons pulled can be noted.

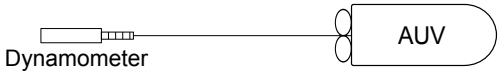


Figure D.1: Overview of test with dynamometer

2. The measurement using the scale can be seen in figure D.2 where the AUV pulls the scale. From this an, amount of kg pulled can be noted.

Figure D.2: Overview of test with scale

D.5 Results

- Using the dynamometer
  When the AUV is set to full speed the dynamometer showed a pulled force at 1.3 N.
- Using the scale
  When the AUV is set to full speed the scale shows an amount of kg of 0.11 kg. This needs to be transformed into newton, this is done in D.1.

$$\text{kg} \cdot 9.8 = 1.1 \text{ N} \tag{D.1}$$

From the two measurements, performed several times each, an average force pulled is calculated: $\frac{1.1 \text{ N}+1.3 \text{ N}}{2} = 1.2 \text{ N}$

D.6 Errors

The string used to connect the AUV and the measurement tool was somewhat elastic. This gave an unstable measurement but the values were calculated from average measurements.

Measurements of propeller speed under water

E

E.1 Purpose

The purpose of this journal is to measure the rotational speed of the propeller when the AUV is submerged. This speed will be used for modelling.

E.2 Tools

Type	AAU no.	Manufacturer	Model
Function generator	07897	Philips	PM 5131
Power supply/battery	-	Jin Power	FM1222
LEDs	-	Seoul Semiconductor	LW-520A
Observer	-	-	-
Water bassin	-	-	-
Multimeter	-	-	-

Table E.1: Tools used for this measurement.

E.3 Procedure

A white marker is put on a propeller wing. The Light Emitting Diode (LED)s are connected to a function generator, which is set up to generate a 3.5 V pulses so the light blinks at the rotational speed of the propeller. The pulse width of the LEDs input signal should be maximum $\frac{1}{6f}$ to make the propeller frequency visible. The light frequency is adjusted to a suitable frequency, to start with 50 Hz.

The test will be carried out in a dark place. While the propeller is set to rotate at maximum speed, the AUV's propeller is put entirely under water while the LEDs light up the propeller. The light frequency is adjusted until the mark on the propeller appears to be static. At this point, the light frequency is read and noted. This will be the rotational speed in RPS. After the test is complete, the frequency is measured using a frequency measurement device to verify the number.

E.4 Measurement arrangement

The measurement is arranged as seen on figure E.1 on the following page.

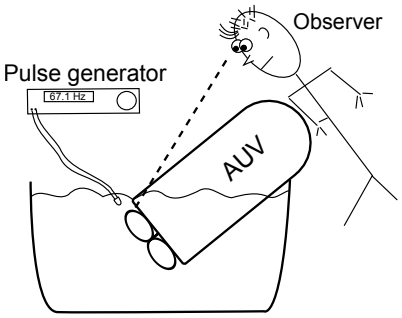


Figure E.1: Measurement arrangement for measuring the propellers speed under water.

E.5 Results

The measurement of the propeller speed under water resulted in a measured speed of 67.1 RPS = 4026 RPM.

E.6 Errors

There are several error sources in this measurement. Firstly, as the readout is done by an observer, that makes for a small error in the readout. This error is estimated to be relatively small.

Another error source is the charge level of the battery, which affects the voltage output. This is a small error source as the battery voltage varies only with a few hundreds of millivolts from fully charged to almost discharged.

The last error source is the size of the water bassin used, in this case a wash, which can affect the waters flow around the propeller.

Measurements on the IMU when standing still

F

The purpose of this journal is to verify the IMU's output data when standing still. In this situation, the accelerometer should output 1 g on the z-axis and the gyrometer should output zero on alle axes. The magnetometers output should be stationary on values depending on the magnetic fields around the IMU and the orientation relative to the Earth. The results can be found in `/measurements/First_IMU_watertest/stillstanding_test.log` and the resulting measurements are plotted on figure F.1, F.2 on the following page and F.3 on the next page.

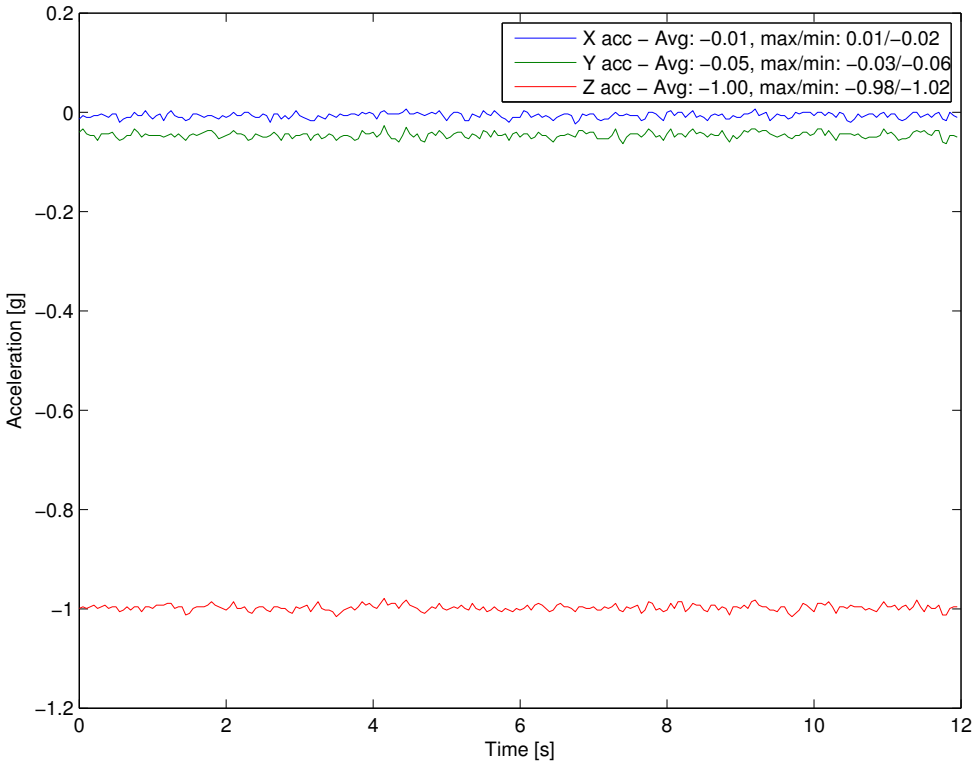


Figure F.1: Raw output from accelerometer while standing still

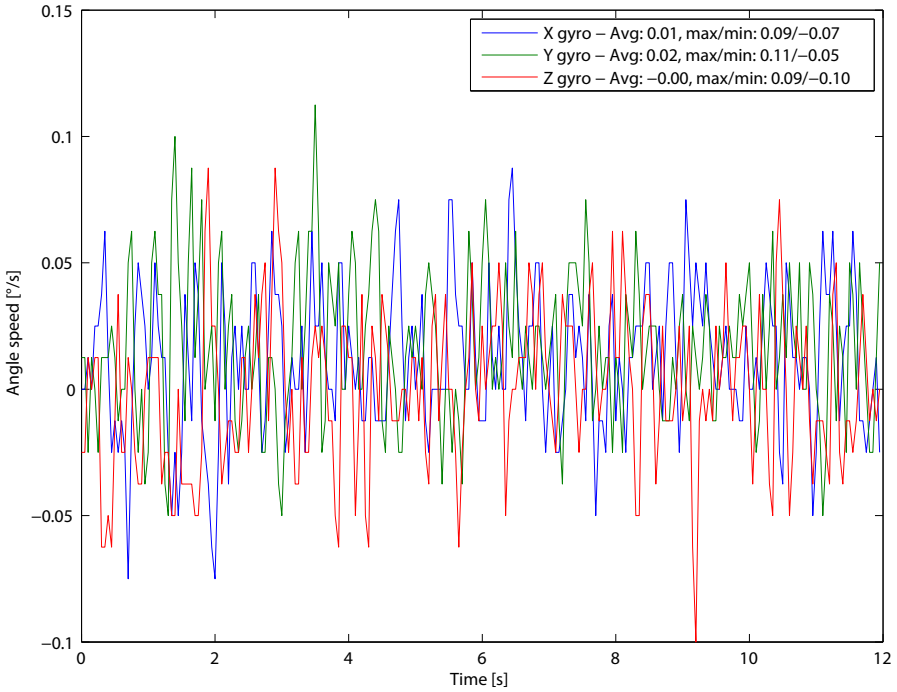


Figure F.2: Raw output from gyrometer while standing still

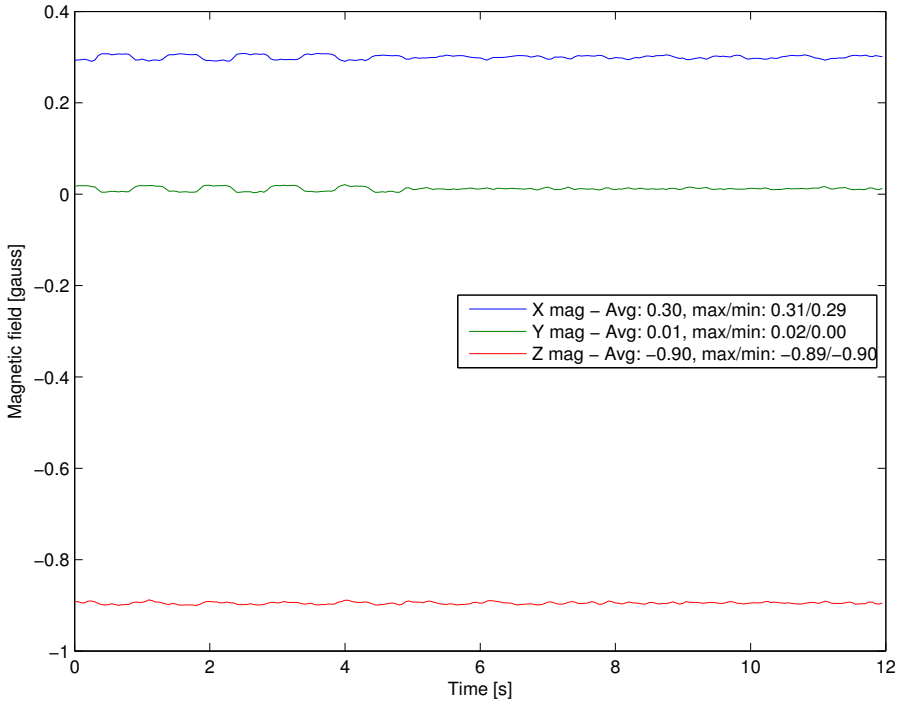


Figure F.3: Raw output from magnetometer while standing still

Calculations on propeller G

The dynamics of a propeller is closely related to two constants. Namely K_t and K_q. These constants express a thrust and torqe coefficient respectively, and the different terms is calculated to end up with an expression for the propeller. The formula for K_t and K_q is given by:

$$\mathrm{K}_t = \sum_{k=1}^{39} C_k \cdot (J)^{S_k} \cdot \left(\frac{P}{D}\right)^{t_k} \cdot \left(\frac{A_E}{A_O}\right)^{u_k} \cdot Z^{v_k} \tag{G.1}$$

$$\mathrm{K}_q = \sum_{k=1}^{47} C_k \cdot (J)^{S_k} \cdot \left(\frac{P}{D}\right)^{t_k} \cdot \left(\frac{A_E}{A_O}\right)^{u_k} \cdot Z^{v_k} \tag{G.2}$$

The C_k values for K_t is found in chapter H on page 123 and is a constant, that changes on each iteration of the summation. The other terms, J, $\frac{P}{D}$, $\frac{A_E}{A_O}$, Z expresses the advance ratio, the pitch ratio, the expanded area ratio and the number of blades. The advance ratio is given by:

$$\frac{V_a}{n \cdot D} \tag{G.3}$$

Where:

V_a = the inflow of water to the propeller
n = the number of revolutions per second
D = the diameter of the propeller

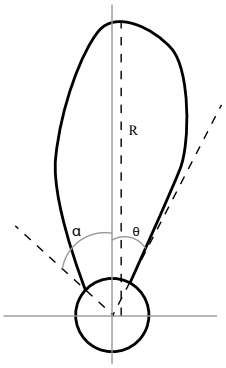


Figure G.1: Single blade illustration

The inflow of water to the propeller is assumed to be the same as the speed at which the AUV is advancing, and is therefore not kept constant and as the number of revolutions changes the thrust delivered by the propeller, this is also a variable. The only constant is the diameter of the propeller, which is 0.037 m.

The next term to be calculated is the pitch ratio, which is defined as the pitch of the blades on the propeller, divided by the diameter. The pitch is measured to 15.66°= 0.273 rad, and the diameter of the propeller is measured earlier to be 0.037 m. So the pitch ratio is calculated by:

$$\text{Pitch Ratio} = \frac{\text{Blade Pitch}}{\text{D}} \tag{G.4}$$

And gives a pitch ratio of 73.31. The $\frac{A_E}{A_O}$ is a ratio that describes the surface area of the blades divided by the surface area the propeller covers. To calculate the area of the blades, a formula from [Carlton, 2007]. is used, that suggests the area of a single blade is calculated by:

$$A_{\text{single-blade}} = \int_{D}^{\text{Pitch}} (\alpha - \theta) \cdot r \cdot dr \tag{G.5}$$

If the formula G.5 on the previous page is used for size calculation per blade, and is then multiplied with 3 the expanded area ratio becomes slightly larger than the overall surface area projected by the propeller (when the propeller is seen as a disc). The last constant is Z - which is the number of blades, which in the csae of the stock submersible is 3. The above equations are used for calculating the thrust and torque coefficients, and are used to calculate how the propeller acts in the water.

Formula used to calculate propeller torque and thrust

H

All values in table H.1 on the following page and H.2 on page 125 are derieved from [Oosterveld and van Oossanen, 1975] and used in the formula to calulate K_{q}:

$$\mathrm{K}_{q,t} = \sum_{k=1}^{N} C_k \cdot (J)^{S_k} \cdot \left(\frac{P}{D}\right)^{t_k} \cdot \left(\frac{A_E}{A_O}\right)^{u_k} \cdot Z^{v_k} \tag{H.1}$$

Where:

C_k = a constant found in table H.1 on the following page
J = the advance ratio of the propeller given as: $\frac{V_a}{n \cdot D}$
S_k = a constant found in table H.1 on the next page
$\frac{P}{D}$ = the pitch ratio of the propeller
t_k = a constant found in table H.1 on the following page
$\frac{A_E}{A_O}$ = the expanded area ratio, calculations can be found in appendix G on page 121
u_k = a constant found in table H.1 on the following page
Z = the number of blades on the propeller
v_k = a constant found in table H.1 on the next page

Inserting values and and reducing the above equation yields equation H.2.

$$\tau(V_x, n) = 12.84103056\text{E-}3 - \frac{83.16005\text{E-}6 \cdot V}{n} - \frac{24.21428828\text{E-}6 \cdot V^2}{n^2} - \frac{9.674514347 \cdot V^3}{n^3} \tag{H.2}$$

And for $\mathrm{K_t}$ as well:

$$\mathrm{K_t}(V_{\mathrm{x}}, n) = -\frac{10.5027 \cdot V_{\mathrm{x}}}{n} - \frac{141.4065 \cdot V_{\mathrm{x}}^2}{n^2} - \frac{1297.6154 \cdot V_{\mathrm{x}}^3}{n^3} + 5.3176 \tag{H.3}$$

Noting that the number of revolutions n is much larger than the velocity V these terms are considered negligible and thus a constant for the load expressed by the propeller is found in equation H.4.

$$\tau_1 \approx 12.84103056\text{E-}3 \tag{H.4}$$

N	C_k	S_k	t_k	u_k	v_k	N	C_k	S_k	t_k	u_k	v_k
1	3.79368E-3	0	0	0	0	25	-39.7722E-3	0	3	2	0
2	8.86523E-3	2	0	0	0	26	-3.50024E-3	0	6	2	0
3	-32.241E-3	1	1	0	0	27	-10.6854E-3	3	0	0	1
4	3.44778E-3	0	2	0	0	28	1.10903E-3	3	3	0	1
5	-40.8811E-3	0	1	1	0	29	-313.912E-6	0	6	0	1
6	-108.009E-3	1	1	1	0	30	3.5985E-3	3	0	1	1
7	-88.5381E-3	2	1	1	0	31	-1.42121E-3	0	6	1	1
8	188.561E-3	0	2	1	0	32	-3.83637E-3	1	0	2	1
9	-3.70871E-3	1	0	0	1	33	12.6803E-3	0	2	2	1
10	5.13696E-3	0	1	0	1	34	-3.18278E-3	2	3	2	1
11	20.9449E-3	1	1	0	1	35	3.342680E-3	0	6	2	1
12	4.74319E-3	2	1	0	1	36	-1.83491E-3	1	1	0	2
13	-7.23408E-3	2	0	1	1	37	112.451E-6	3	2	0	2
14	4.38388E-3	1	1	1	1	38	-29.7228E-6	3	6	0	2
15	-26.9403E-3	0	2	1	1	39	269.551E-6	1	0	1	2
16	55.8082E-3	3	0	1	0	40	832.65E-6	2	0	1	2
17	16.1886E-3	0	3	1	0	41	1.55334E-3	0	2	1	2
18	3.18086E-3	1	3	1	0	42	302.683E-3	0	6	1	2
19	15.896E-3	0	0	2	0	43	-184.3E-6	0	0	2	2
20	47.1729E-3	1	0	2	0	44	-425.399E-6	0	3	2	2
21	19.6283E-3	3	0	2	0	45	86.9243E-6	3	3	2	2
22	-50.2782E-3	0	1	2	0	46	-465.9E-6	0	6	2	2
23	-30.055E-3	3	1	2	0	47	55.4194E-6	1	6	2	2
24	41.7122E-3	2	2	2	0						

Table H.1: Table of constants used for calculating propeller torque in H.1 on page 123.

N	C_k	S_k	t_k	u_k	v_k	N	C_k	S_k	t_k	u_k	v_k
1	8.80496E-3	0	0	0	0	21	10.465E-3	1	6	2	0
2	-204.554E-3	1	0	0	0	22	-6.48272E-3	2	6	2	0
3	166.351E-3	0	1	0	0	23	-8.41728E-3	0	3	0	1
4	158.114E-3	0	2	0	0	24	16.8424E-3	1	3	0	1
5	-147.581E-3	2	0	1	0	25	-1.02296E-3	3	3	0	1
6	-481.497E-3	1	1	1	0	26	-31.7791E-3	0	3	1	1
7	415.437E-3	0	2	1	0	27	18.604E-3	1	0	2	1
8	-14.4043E-3	0	0	0	1	28	-4.10798E-3	0	2	2	1
9	-53.0054E-3	2	0	0	1	29	-606.848E-6	0	0	0	2
10	14.3481E-3	0	1	0	1	30	-4.9819E-3	1	0	0	2
11	60.6826E-3	1	1	0	1	31	2.5983E-3	2	0	0	2
12	-12.5894E-3	0	0	1	1	32	-560.528E-6	3	0	0	2
13	10.9689E-3	1	0	1	1	33	-1.63652E-3	1	2	0	2
14	-133.698E-3	0	3	0	0	34	-328.787E-6	1	6	0	2
15	6.38407E-3	0	6	0	0	35	116.502E-6	2	6	0	2
16	-1.32718E-3	2	6	0	0	36	690.904E-6	0	0	1	2
17	168.496E-3	3	0	1	0	37	4.21749E-3	0	3	1	2
18	-50.7214E-3	0	0	2	0	38	56.5229E-6	3	6	1	2
19	85.4559E-3	2	0	2	0	39	-1.46564E-3	0	3	2	2
20	-50.4475E-3	3	0	2	0						

Table H.2: Table of constants used for calculating propeller thrust in H.1 on page 123.

Calculating the speed of sound in fresh water

I

To calculate the speed of sound in water at a given pressure and temperature, an algorithm from [Chen and Millero, 1977, pages 1129-1135] is used. This algorithm is an international standard algorithm for calculating the speed of sound in water and is also known as the United Nations Educational, Scientific and Cultural Organization (UNESCO) algorithm. The algorithm is given by the formula in equation I.1 where S is the salinity in parts per thousand, T is the temperature in degrees celsius and P is the pressure in bar. The formula calculates the speed c in metres per second.

$$c(S,T,P) = Cw(T,P) + A(T,P)S + B(T,P)S^{3/2} + D(T,P)S^2 \quad \left[\frac{\mathrm{m}}{\mathrm{s}}\right] \tag{I.1}$$

As the limitations for this project, as described in section 1.5 on page 6, limits the testing to non-saline water, the salinity S is set to be zero. Therefore, only the first term, $Cw(T,P)$, is used, and it is defined as in equation I.2.

$$\begin{aligned} Cw(T,P) = {} & (C_{00} + C_{01}\cdot T + C_{02}\cdot T^2 + C_{03}\cdot T^3 + C_{04}\cdot T^4 + C_{05}\cdot T^5) \\ & + (C_{10} + C_{11}\cdot T + C_{12}\cdot T^2 + C_{13}\cdot T^3 + C_{14}\cdot T^4)\cdot P \\ & + (C_{20} + C_{21}\cdot T + C_{22}\cdot T^2 + C_{23}\cdot T^3 + C_{24}\cdot T^4)\cdot P^2 \\ & + (C_{30} + C_{31}\cdot T + C_{32}\cdot T^2)\cdot P^3 \end{aligned} \tag{I.2}$$

The coefficients are defined as follows in table I.1.

Coefficient	**Value**	**Coefficient**	**Value**
C_{00} =	1402.38800	C_{14} =	-6.1260E-10
C_{01} =	5.03830	C_{20} =	3.1260E-5
C_{02} =	-5.81090E-2	C_{21} =	-1.7111E-6
C_{03} =	3.34320E-4	C_{22} =	2.5986E-8
C_{04} =	-1.47797E-6	C_{23} =	-2.5353E-10
C_{05} =	3.14190E-9	C_{24} =	1.0415E-12
C_{10} =	0.153563	C_{30} =	-9.7729E-9
C_{11} =	6.89990E-4	C_{31} =	3.8513E-10
C_{12} =	-8.18290E-6	C_{32} =	-2.3654E-12
C_{13} =	1.36320E-7		.

Table I.1: The coefficients used in the $Cw(T,P)$ function.

Using the UNESCO algorithm, the speed of sound in water at a salinity of 0 parts per thousand, a pressure of 1 bar and a temperature of 20 ℃ is calculated in equation I.3.

$$c(0,20,1) = 1482.524\ \frac{\mathrm{m}}{\mathrm{s}} \tag{I.3}$$

The UNESCO algorithm is valid in a range of temperatures from 0 to 40 ℃, salinity from 0 to 40 parts per thousand and pressure from 0 to 1000 bar [Wong and Zhu, 1995, pages 1732-1736].

Decoding of IMU data J

This appendix describes how the raw bits from the IMU (ADIS16405) can be decoded to get meaningfull data out of it. This is not included in the main report, as it is not considered essential for the project, but yet important to document.

The data from the IMU is given as a output format of twos complement and the units for each output register is defined in table J.1.

Register name	Bits	Scale/LSB
SUPPLY_OUT	14	2.43 mV
XGYRO_OUT	14	0.05 °/s
YGYRO_OUT	14	0.05 °/s
ZGYRO_OUT	14	0.05 °/s
XACCL_OUT	14	3.33 mg
YACCL_OUT	14	3.33 mg
ZACCL_OUT	14	3.33 mg
XMAGN_OUT	14	0.5 mG
YMAGN_OUT	14	0.5 mG
ZMAGN_OUT	14	0.5 mG
TEMP_OUT	12	0.14 ℃
AUX_ADC*	12	0.81 mV

* This register is not coded as twos complement, it is binary

Table J.1: Output data register formats [see Analog Devices, 2009, p. 11]

J.1 A brief reminder on twos complement

Twos complement is a way to represent signed integers in binary patters. It works by defining the integer number zero as all zeros in the bit sting. The Most Significant Bit (MSB) denotes the sign of the integer, when MSB is zero the integer is positive, and while the MSB is one the integer is negative. In the positive case, the other bits represent the binary value directly, and in the negative case the zero bits now represent the ones in the binary encoding, plus one. See table J.2 on the next page for the representation.

J.2 The actual decoding

Decoding is all about handling bits correctly. The source codes J.1 on the following page is a "C" implementaion of a way to decode the data. The

Bits	Integer
01111111	127
01111110	126
00000010	2
00000001	1
00000000	0
11111111	-1
11111110	-2
10000001	-127
10000000	-128

Table J.2: Examples of twos complement.

`singed int` in C is actually coded as a twos complement integer. As the data is not 16 bits as the signed integer is, the irrelevant bits should be handled, so they represent the range the integer is in with respect to the sign. These irrelevant bits are handled in lines 9 and 10. Line 8 is to stitch the two received bytes together to a signed integer.

```
1  /**
2   * This decodes the 14 bit raw data from the ADIS16405 sensor
3   * scale sets the scaling per least significant bit
4   * SENSOR_OUT[2] is the recived raw data
5   */
6  int adis_decode_14bit_raw(unsigned char SENSOR_OUT[2], int scale){
7    int sensor = 0;
8    sensor = (int)((SENSOR_OUT[0] << 8) | SENSOR_OUT[1]) & 0x3fff;
9    if(sensor>=0x2000)
10     sensor = 0xffffc000 | sensor;
11   sensor = sensor * scale;
12   return sensor;
13 }
```

Source code J.1: An exampe of decoding data from the IMU, here the 14 bit data.

Software code examples K

K.1 Actuator control

The actuator control "C" file lists a set of functions that can be utilized by the controlling algorithms. an example for the rudder control is given in source code K.1

```
1 void set_rudder_angle( signed int degrees ) { // Yaw
2   degrees = degrees > 36 ? 36 : (degrees < -34 ? -34 : degrees);
3   pwm_set(6 , (6544-(6*degrees))/7;
4 }
```

Source code K.1: Function for setting the rudder angle.

Line one in source code K.1 defines the function that turns the rudder and a the variable this function takes, i.e. degrees.

In line two degrees are converted into the corresponding duty cycle the servo motor needs, to position the rudder at the right angle see section 8.6 on page 72.

The third line sets the calculated value to the PWM port connected to the rudder servo, the PWM functions are described more thoroughly in section K.1.

The control functions for the motor and the buoyancy control pump is devised in a similar manner. Furthermore there have been written a code for pitch fine tuning, though this is only used for keeping the right angle and not active control.

The PWM functions

The PWM "C" file contains functions for initializing the PWM, setting a specific duty cycle on a PWM port and a function that contains more complex functions for controlling servos.

Code K.2 on the next page shows the function for initializing the PWM ports. Before the function, `AT91C_BASE_PWMC_CH3`, `6` and `7` is defined. In line 1 in the source code K.2 on the following page defines the function, which takes one argument, which determines which channel to initialize.

Line 18 on the next page sets the update time for of the PWM cycle as a multiple of the master clock. In this case the update rate is the same as the master clock.

Line 20 on the following page activates the PWM channel, by annotating the register responsible for that to on the microcontroller.

The switch case started in line 22 on the next page enables the PWM channels acording to the input. First the update rate of the PWM signal is set, secondly the number of steps per cycle is set, last the starting value of the PWM output is set.

```
1  void pwm_init(int ch){
2    /* Pointer to PMC controller */
3    volatile AT91PS_PMC pPMC = AT91C_BASE_PMC;
4    volatile AT91PS_PIO pPIO = AT91C_BASE_PIOA;
5
6    /* Activates the PWM controllers clock */
7    pPMC->PMC_PCER = (1 << AT91C_ID_PWMC);
8
9    /* Activates the PIOC (Periphial Input/Output Controller) */
10   pPIO->PIO_PDR = (AT91C_PIO_PA21);  // Enable PWM3 on port PA21
11   pPIO->PIO_ASR = (AT91C_PIO_PA21);  // Assign PA21 to peripheral A
12   pPIO->PIO_PDR = (AT91C_PIO_PA24);  // Enable PWM6 on port PA24
13   pPIO->PIO_ASR = (AT91C_PIO_PA24);  // Assign PA24 to peripheral A
14   pPIO->PIO_PDR = (AT91C_PIO_PA25);  // Enable PWM7 on port PA25
15   pPIO->PIO_ASR = (AT91C_PIO_PA25);  // Assign PA25 to peripheral A
16
17   /* Configurations of PWM */
18   pPWMC->PWMC_MR = 0; // No prescaler when 0
19
20   pPWMC->PWMC_ENA = (1 << ch); // Activates a PWM channel
21
22   switch (ch ) {
23     case 3:
24       pPWM3->PWMC_CMR = 0x0A; // MCK/1024
25       pPWM3->PWMC_CDTYR = 1024; // outHz = pwm_clock / pwm_CPRD
26       pPWM3->PWMC_CPRDR = 1024; //Period time ts = 46.9kHz/PWMC_CPRDR
27       break;
28     case 6:
29       pPWM6->PWMC_CMR = 0x0A; // MCK/1024
30       pPWM6->PWMC_CDTYR = 1024; // outHz = pwm_clock / pwm_CPRD
31       pPWM6->PWMC_CPRDR = 1024; //Period time ts = 46.9kHz/PWMC_CPRDR
32       break;
33     case 7:
34       pPWM7->PWMC_CMR = 0x0A; // MCK/1024
35       pPWM7->PWMC_CDTYR = 1024; // outHz = pwm_clock / pwm_CPRD
36       pPWM7->PWMC_CPRDR = 1024; //Period time ts = 46.9kHz/PWMC_CPRDR
37       break;
38   }
39 }
```

Source code K.2: Function for initializing the PWM ports.

K.2 Ground station

A code example from the serial reader is shown on figure K.3 on the facing page. Line two in this code example defines a byte-array of the size 256 bytes which is the size of the receive buffer in the wireless radio module. Line five to seven reads new data and sends it to the data parser in the GroundStation class. Line nine prints some useful trace information if anything goes wrong.

A code example from the serial write function is shown on figure K.4. Line

```
1  public void run () {   // Reader running in its own thread
2    byte[] buffer = new byte[256]; // Rcv. buffer 256 bytes
3    int len = -1;         // Initialize length array
4    try {                 // Read new data when available
5      while ( (len = in.read(buffer)) > -1 ) {
6      GroundStation.parseData(buffer,len); // Parse data
7      }
8    } catch ( IOException e ) {
9      e.printStackTrace(); // Exception handling
10   }
11 }
```

Source code K.3: Function for reading data from serial port.

two in this code example checks if there exists a valid connection by checking if the outputstream out equals null. Line four writes the desired data to the serial port. Line five to eight checks if it has been selected to send a new line and if that is the case, it writes LF and CR characters. Line ten prints useful information if anything goes wrong.

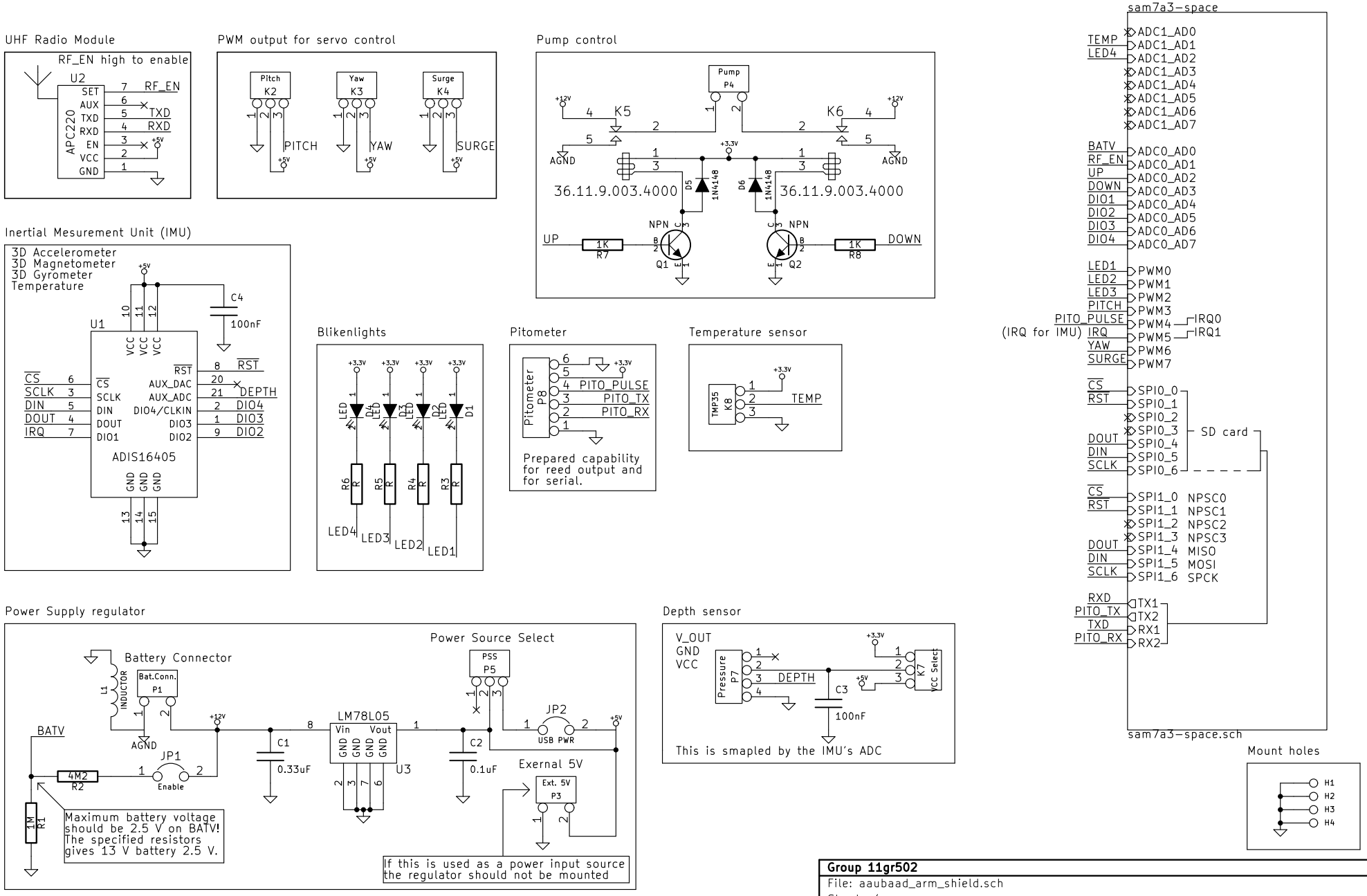
```
1  public static void write(String data, boolean EOL) {
2    if(out != null) {     // Check if there is conection
3      try {               // Write the characters
4        out.write(data.getBytes());
5        if(EOL){          // Send new line?
6          out.write(0xa); // LF character
7          out.write(0xd); // CR character
8        }
9      } catch (IOException e) {
10       e.printStackTrace(); // Exception handling
11     }
12   }
13 }
```

Source code K.4: Function for writing data to serial port.

Schematic of hardware

L

This is the schematic diagram of the developed shield for the ARM7 microcontroller that is used in this project.


UHF Radio Module
RF_EN high to enable
U2
APC220
SET
AUX
TXD
RXD
EN
VCC
GND
RF_EN
TXD
RXD
PWM output for servo control
Pitch
K2
PITCH
Yaw
K3
YAW
Surge
K4
SURGE
Pump control
Pump
P4
K5
K6
AGND
36.11.9.003.4000
D5
D6
1N4148
NPN
Q1
Q2
UP
DOWN
1K
R7
R8
Inertial Mesurement Unit (IMU)
3D Accelerometer
3D Magnetometer
3D Gyrometer
Temperature
U1
ADIS16405
C4
100nF
VCC
GND
CS
SCLK
DIN
DOUT
IRQ
RST
AUX_DAC
AUX_ADC
DIO4/CLKIN
DIO3
DIO2
DIO1
DEPTH
Blikenlights
LED
LED4
LED3
LED2
LED1
R6
R5
R4
R3
Pitometer
Pitometer
P8
PITO_PULSE
PITO_TX
PITO_RX
Prepared capability
for reed output and
for serial.
Temperature sensor
TMP35
K8
TEMP
sam7a3-space
ADC1_AD0
ADC1_AD1
ADC1_AD2
ADC1_AD3
ADC1_AD4
ADC1_AD5
ADC1_AD6
ADC1_AD7
TEMP
LED4
ADC0_AD0
ADC0_AD1
ADC0_AD2
ADC0_AD3
ADC0_AD4
ADC0_AD5
ADC0_AD6
ADC0_AD7
BATV
RF_EN
UP
DOWN
DIO1
DIO2
DIO3
DIO4
PWM0
PWM1
PWM2
PWM3
PWM4
PWM5
PWM6
PWM7
LED1
LED2
LED3
PITCH
PITO_PULSE
(IRQ for IMU) IRQ
YAW
SURGE
IRQ0
IRQ1
SPI0_0
SPI0_1
SPI0_2
SPI0_3
SPI0_4
SPI0_5
SPI0_6
SD card
SPI1_0 NPSC0
SPI1_1 NPSC1
SPI1_2 NPSC2
SPI1_3 NPSC3
SPI1_4 MISO
SPI1_5 MOSI
SPI1_6 SPCK
RXD
PITO_TX
TXD
PITO_RX
TX1
TX2
RX1
RX2
sam7a3-space.sch
Power Supply regulator
Battery Connector
Bat.Conn.
P1
L1
INDUCTOR
BATV
AGND
JP1
Enable
4M2
R2
1M
R1
Maximum battery voltage
should be 2.5 V on BATV!
The specified resistors
gives 13 V battery 2.5 V.
C1
0.33uF
LM78L05
Vin
Vout
GND
U3
Power Source Select
PSS
P5
JP2
USB PWR
C2
0.1uF
Exernal 5V
Ext. 5V
P3
If this is used as a power input source
the regulator should not be mounted
Depth sensor
V_OUT
GND
VCC
Pressure
P7
DEPTH
C3
100nF
K7
VCC Select
This is smapled by the IMU's ADC
Mount holes
H1
H2
H3
H4
Group 11gr502
File: aaubaad_arm_shield.sch
Sheet: /
Title: AAUBAAD ARM Shield
Size: A4
Date: 16 dec 2011
Rev: 1.1
KiCad E.D.A.
Id: 1/2

Bibliography

Analog Devices. *ADIS 16405 Datasheet*. www.analog.com/static/imported-files/data_sheets/ADIS16405.pdf, 2009. Downloaded October 4 2011. /datasheets/ADIS16405.pdf.

Appcon Technologies. *APC Series Transparent Transceiver Module*, 2008. Downloaded October 10 2011. /datasheets/APC220_manual2.pdf.

Atmel. *AT91SAM7A3 Preliminary*. http://atmel.com/dyn/resources/prod_documents/doc6042.pdf, 2006. Downloaded October 5 2011. /datasheets/doc6042.pdf.

Rodney A. Brooks. *A robust layered control system for a mobile robot*. http://mit.dspace.org/bitstream/handle/1721.1/6432/AIM-864.pdf, 1986. Downloaded November 8 2011 /litterature/Brooks_1986.pdf.

John Carlton. *Marine Propellers and Propulsion*. Butterworth-Heinemann, 2007. ISBN 978-07506-8150-6.

Jonathan C. Newell Charles M. Close, Dean K. Frederick. *Modeling and Analysis of Dynamic Systems*. John Wiley & Sons, INC, 2002. ISBN 0471394424.

C-T. Chen and F.J. Millero. *Speed of sound in seawater at high pressures*. J. Acoust. Soc. Am., 1977.

James Diebel. Representing attitude: Euler angles, unit quaternions and rotation vectors.

Engineering Toolbox. *Drag Coefficient*. http://www.engineeringtoolbox.com/drag-coefficient-d_627.html, 2011. Visited October 4 2011. /litterature/DragCoefficient.pdf/.

Abbas Emami-Naeini Gene F. Franklin, J.David Powell. *Feedback Control of Dynamic Systems*. Pearson Prentice Hall, 2009. ISBN 9780135001509.

M.W.C Oosterveld and P. van Oossanen. Further computer-analyzed data of the wageningen b-screw series. *International Shipbuilding Progress*, 22, 1975.

K.J. Rawson and E.C. Tupper. *Basic Ship Theory*. Butterworth Heinemann, 2001. ISBN 0750653973.

Asgeir J. Soerensen. *Marine Cybernetics Modelling and Control*. Department of Marine Technology NTNU, 2005. ISBN UK-05-76.

Various Authors. *RXTX Wiki*. http://rxtx.qbang.org/wiki, 2011. Visited November 28 2011. /litterature/rxtx_examples/.

Washington State University. *Applied Physics laboratory*. http://www.apl.washington.edu/projects/seaglider/summary.html, 2011. Visited September 28 2011. /litterature/Seaglider-Summary.pdf/.

G.S.K. Wong and S Zhu. *Speed of sound in seawater as a function of salinity, temperature and pressure.* J. Acoust. Soc. Am., 1995.