NTNU
Norwegian University of
Science and Technology

Low Energy Buoyancy Actuator for Vertical Underwater Motion

Stian Børseth

Master of Science in Cybernetics and Robotics
Submission date: June 2017
Supervisor: Jo Arve Alfredsen, ITK

Norwegian University of Science and Technology
Department of Engineering Cybernetics

Abstract

This thesis presents the physical and technical premise for implementing and designing an underwater vehicle capable of moving vertically using the unit's own buoyancy force, and how this principle benefit in energy efficiency. The goal is to create a prototype unit that makes experimental testing and documentation of this principle possible.

An electrical linear actuator with a piston was used to manipulate the unit's volume. Depth control was achieved using a PID controller combined with a pressure sensor, and the control parameters tuned by implementing simulations of the unit's dynamical behavior. By combining two power saving methods, it was estimated (using simulations) to reduce the power consumption to 12.6% of maximum power consumption. A 3D model of the unit was made to determine the vertical stability, mass properties, and to create drawings of a prototype.

A functional prototype was successfully implemented, and two physical experiments were carried out. The physical experiments were not sufficient to determine the unit's energy efficiency using buoyancy as a principle of vertical movement underwater. But the work here suggest there is a promising potential for the unit being energy efficient.

Sammendrag

Denne oppgaven presenterer de fysiske og tekniske forutsetningene for implementasjon og design av et undervannsfartøy i stand til vertical bevegelse ved bruk av enhetens egen oppdriftskraft, og hvordan dette prinsippet drar nytte av energieffektivitet. Målet er åskape en prototype enhet som gjør eksperimentell testing og dokumentasjon av dette prinsippet mulig.

En elektrisk lineær aktuator med et stempel ble brukt til åmanipulere enhetens volum. Dybdekontroll ble oppnådd ved hjelp av en PID-regulator kombinert med en trykksensor, og kontrollparameterene innstilt ved åimplementere simuleringer av enhetens dynamiske oppførsel. Ved åkombinere to strømsparingsmetoder ble det estimert (ved bruk av simuleringer) til åredusere strømforbruket til 12,6% av maksimalt strømforbruk. En 3D-modell av enheten ble laget for åbestemme vertikal stabilitet, massegenskaper og ålage tegninger av en prototype.

En funksjonell prototype ble vellykket implementert, og to fysiske forsøk ble utført. De fysiske forsøkene var ikke tilstrekkelig til åbestemme enhetens energieffektivitet ved åbruke oppdrift som et prinsipp for vertikal bevegelse under vann. Men arbeidet her tyder påat det er et lovende potensial for at enheten skal være energieffektiv.

Acknowledgement

I would first like to thank my project supervisor, Jo Arve Alfredsen of the Department of Engineering Cybernetics at NTNU, for providing an intriguing project, and theoretical and practical guidance.

I would also like to thank Terje Haugen and Daniel Bogen at the mechanical workshop for their hard work and insight toward building the unit. I appreciate the time and effort put in by Stefano Brevik Bertelli to arrange and participate in an ideal testing location at SINTEF Marintek Facilities.

Author

Stian Børseth

Table of Contents

List of Tables

List of Figures

Symbols

Symbol	Name	Unit
F	Force	N
ρ	Density	$\frac{kg}{m^3}$
V	Volume	m^3
m	Mass	kg
A	Area	m^2
P	Power	W (Js^{-1})
U	Voltage	V
x	Position	m
v	Velocity	$\frac{m}{s}$
a	Acceleration	$\frac{m}{s^2}$
T	Temperature	$^\circ C$
S	Salinity	psu
ω	Angular Velocity	$\frac{rad}{s}$
τ	Torque	Nm
n	Gear ratio	
p	Pitch	m
d	Depth	m

Abbreviations

BV	**B**uoyancy **V**ehicle
CTD	**C**onductivity **T**emperature **D**epth
PID	**P**roportional **I**ntegral **D**erivative
ADC	**A**nalog to **D**igital **C**onverter
PCB	**P**rinted **C**ircuit **B**oard

Chapter 1

Introduction

1.1 Outline

The thesis is structured in six chapters: Indroduction, Theory, Method, Results, Discussion and Conclution.

Introduction

The introduction aims to get an overview of what this thesis is about, and what problem it is trying answer. Background and some applications are also included here.

Theory

Theory chapter presents the mathematical equations and theoretical background used in this thesis.

Method

Method tries to not only explain, but also show what methods, design choice and implementations used to get the results.

Results

This is were the findings of physical experiments and prototype properties is presented.

Discussion

Argumentation of the results, trying to explain the findings.

Conclution

Summary about the goal and scope of the project. Future work also included.

1.2 Project Description

This project is meant to clarify the physical and technical premise for creating a unit whose purpose is to move vertically underwater. The movement is created by manipulating the unit's own buoyancy force, by a controllable volume change. Based on this principle, a control system will be developed, which is able to control and keep the unit stable at an arbitrary position in the given range of depth (0 - 50 meter). Slow movement ($< 0.2m/s$) is assumed sufficient, but there should be emphasis on low energy consumption and long operating time. The goal is to develop a prototype, which enables the possibility to study and document the properties and performance of the unit through physical experiments. One application of this system can be associated with surveillance of water quality in aquaculture facilities, through an automatic profiling measuring probe, i.e. the unit is combined with a payload of the necessary water quality sensors. The project consists of the following points:

- Study of relevant literature
- Deriving of mathematical models that describes the unit's dynamic
- Simulation of the unit's behavior under various operating conditions, and develop a control strategy for stabilizing the unit at arbitrary depths
- Plan and realize a prototype, including mechanical design, mechanism for electro-magnetic volume change, and embedded computer and sensors for controlling the unit
- Plan and execute physical experiments, and document the prototypes properties and performance
- Discussion and conclusion

1.3 Background

Using buoyancy for energy efficiency can be observed naturally in biology on fishes with a swim bladder. Species such as Salmon utilizes their swim bladder by inhaling or exhaling air[Saunders, 1965]. If the air intake is correctly adjusted, the Salmon will then change its density to match the surrounding water, and thus will require little or no energy to remain at a given depth. Mimicking this feature could provide useful on technical solutions in oceanography, for energy efficiency.

1.3.1 Ocean Sensing

The study of ocean sensing provides with several difficulties. Unlike long range communication in air, electromagnetic waves are absorbed in a large fraction in water. Sensing from remote technologies, such as satellites and airplanes, will only receive data from the surface layer of the ocean, and can be limited to what they are able to measure[Devi et al., 2015]. In situ measurements will increase the dataset for analyzing ocean dynamics. Because of the vast size of the ocean, collecting meaningful data in situ requires a large fleet of sensors that is automated with long operational time for cost efficiency.

One notable implementation is Argo, a large scale ocean profiling network, currently containing a global array of 3800 free-drifting floats that measures temperature and salinity of the upper 2000 m of the ocean.[Roemmich et al., 2009]

1.3.2 Buoyancy Control Mechanisms

Using buoyancy for vertical movement is not an unused concept in oceanography. There are many examples of working and theoretical devices/instruments using buoyancy for depth control. Some examples of different mechanisms to achieve this will be explored here, and try to look at some of their advantages and limitations.

Compressed Gas

An automatic buoyancy control device was patented already in 1946 (US 2968053 A)[Richard et al., 1961], a gas operated device that is able to quickly descend an underwater acoustic device to a pre-set operating depth and maintain in a stable state there with minimum oscillation. Using gas in a pressurized container provides a compact potential energy source, useful for larger volume changes when lifting heavy objects.

Mass Removal

Another patent (US 5379267 A) from 1995 describes controlling a buoyancy system by jettisoning either a heavy liquid or light liquid[Sparks et al., 1995]. These sort of buoyancy control mechanisms where a payload is expended is efficient for single deployment use at a predefined depth. Releasing materials and liquids into the ocean could provide with unwanted environmental hazards.

Oil

The autonomous Lagrangian circulation explorer (ALACE) [Davis et al., 1992] is a subsurface float, designed to cycle vertically from a pre-defined depth and back up to surface, where data is relayed to satellites. It uses a hydraulic pump to move oil from internal reservoir to external bladder, changing its own volume. It concluded, by predicting the measured battery-voltage decay, it was able to carry out 50+ cycles to 1-km depth over a 4+ year lifetime. This demonstrates an ideal method for energy efficiency with the ability to reach high pressure depths by using a battery driven hydraulic oil pump. This method is also used in the Argo sensing floats.

Phase Transition

A rather peculiar buoyancy control mechanism can be observed naturally on sperm whales, where the sperm whales can change its buoyancy by melting and coagulate the sperm oil in its head[Clarke, 1978]. The buoyancy change happen because the phase transition[1] of materials can result in a volume change. Experimental tests utilizing this mechanism was documented using the phase transition of paraffin wax[Yamamoto and Shibuya, 2016]. SOLO-TREC (Sounding Oceanographic Lagrangrian Observer Thermal RECharging)[Scripps] is a similar subsurface float to ALACE, however it utilized phase transition to charge a battery pack to increase its operating time.

Linear Actuator

The Mini-Autonomous Underwater Explorer (M-AUE) [Jaffe et al., 2017] is a compact device, designed to mimic the vertical swimming behaviors of plankton using buoyancy control. It uses a battery-powered electric motor connected to a small piston, to achieve small incremental changes in the vehicle volume. It demonstrated the ability to provide adequate depth control in the upper 50 m of the water column. Using an electrical motor to move a piston as a buoyancy control method is a simplistic and easy implemented solution with much precision and accuracy on the volume of the vehicle.

[1]Phase Transition is the transition between solid, liquid and gaseous state of a material

1.4 Scope and goal of project

In what way can using buoyancy as a principle of vertical movement underwater benefit in energy efficiency, and what is the technical and physical premise for creating a unit using this principle.

The goal of this project is to create a prototype that makes experimental testing and documentation of this principle possible, and document the prototypes properties and performance.

1.5 System Requirements

The unit that will be developed is referred to as Buoyancy Vehicle (BV), which include all physical parts implemented. The requirements of the BV is the following points:

- Robust physical design, capable of withstanding underwater pressures up to 50 m depth.
- Capability of controlling its vertical position underwater at arbitrary depths by using controllable incremental volume changes.
- Autonomous during operation.
- Be energy efficient for long operation times.
- Function both as stand-alone and have an option to attach an addtional buoyancy negative payload to it.

1.6 Applications

1.6.1 Profiling Point Measurement

The BV can be used to continuously profiling vertically in a single horizontal location, see figure 1.1. This can be achieved by connecting the BV to an anchored line leading from the surface to bottom of the wanted location. Attaching a payload, such as a CTD probe, to the BV, it can continuously measure the water profile over a long time without manual/human interaction. This principle can also be extended with either an acoustic transmitter or a GPS transmitter, for respectively short and long range communication, and thus updating the measured result more frequently during operation.

1.6.2 Local Current Measurement

In sea-based aquaculture, ocean currents around the sea cages have a large impact of the growth and survival of the fishes. This is because stronger currents provide more oxygen to be absorbed by the fishes. Measurements of currents have traditionally been taken outside the sea cage to get an estimate of the water quality inside the sea cage. Having an option to insert a device inside the cage, which is gentle to the surrounding fishes, vertically stationary at various depths, and free floating with the currents in a horizontal direction, can be useful for achieving a better approximation of the speed and movement of the currents inside the sea cage. One example of measuring the currents could be by implementing the BV with an underwater acoustic positioning system, and monitor its horizontal movements at the various depths.

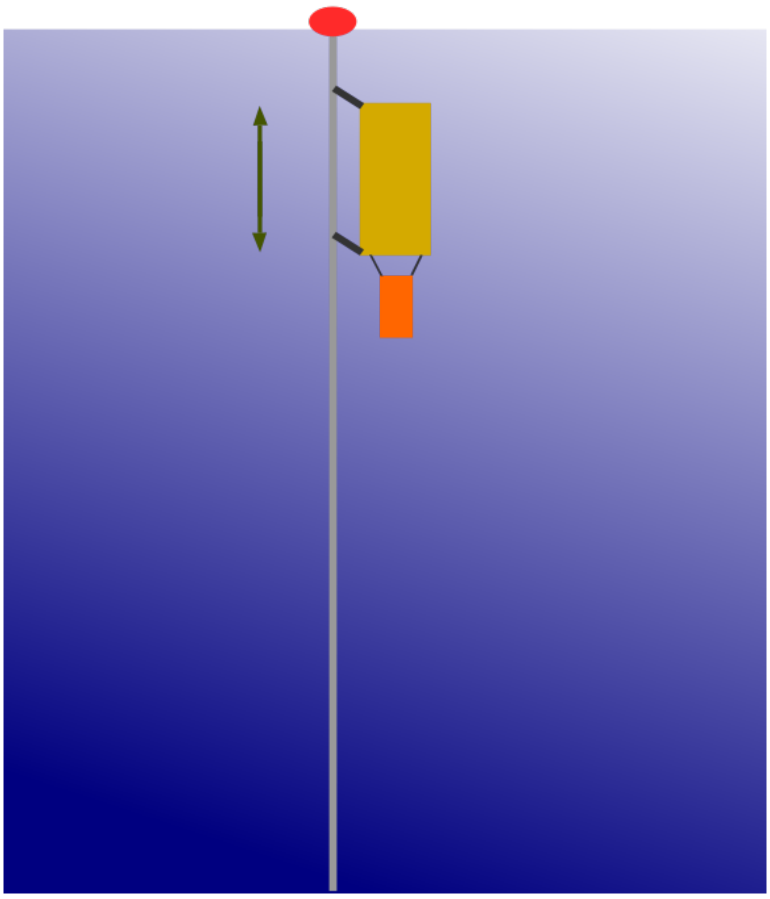


Figure 1.1: Example of an application for the BV being used as a profiling point measurement. The BV can move underwater vertically while remaining horizontally attached to an anchored line.

Chapter 2

Theory

2.1 Fluid Mechanics

This section will focus on the fundamental theory in Fluid Mechanics that is necessary to understand in the process of designing a BV. The theory of Fluid Mechanics is needed primarily for creating a simulated dynamic model, and calculating the properties (mass, volume etc.) of the BV. The theory in this section is based on the book ”Fluid Mechanics Fundamentals and Applications”[Çengel and Cimbala, 2010].

2.1.1 Vertical Forces: Buoyancy and Gravity

When an object is partially or fully submerged in a fluid, an upward force is exerted by the fluid onto the object, called the buoyancy force. This force is caused by an increase of pressure with depth in a fluid, resulting in a different gage pressures on surfaces of an object. The buoyant force acting on the object is equal to the mass of the displaced fluid by the object, e.g.

$$\vec{F}_B(t) = \rho_f \vec{g} V(t) \tag{2.1}$$

where ρ_f is the density of the fluid and V is the volume of the object that is submerged in the fluid. The volume is also varying with time in this application. A gravitational force

$$\vec{F}_G = m\vec{g} \tag{2.2}$$

where m is the mass of the object, works in the opposite direction of the buoyancy force, making those forces the main contributions on the vertical forces of a stationary and undisturbed object. When an object has the properties of:

$$\vec{F}_G = \vec{F}_B(t) => m = \rho_f V(t) \tag{2.3}$$

it is considered neutrally buoyant, meaning the object will neither sink or float in the fluid. If the volume of the object is changed (while maintaining the same mass) from a neutrally buoyant state, will result in the object being able to sink or float in the fluid.

2.1.2 Drag

Drag can be seen as a resistance an object meets when its travelling through a fluid. The drag can be expressed as a force that works in the opposite direction to the velocity of the object. The total drag force is dependent on the density of the fluid, the velocity of the flow over the object, shape and size of the object, and a set variables that is practically set to a drag coefficient:

$$\vec{F}_D(t) = \frac{1}{2} C_D \rho_f \vec{v}(t)^2 A \tag{2.4}$$

A is the frontal area of the object in the direction of the flow. C_D is the drag coefficient, and can to a sufficient degree be approximated based on shape and size of the object.

2.1.3 Density

As seen in the vertical force equations, the fluid density is a crucial factor for the acting forces. Density is defined as mass per unit volume, or $\rho = \frac{m}{V} [kg/m^3]$. Sea water density will vary primarily based on the temperature (T) and salinity (S). Varying pressure is also a factor to the sea water density, but this is negligible for minor depths (0 to 50 m). Sea water density can be estimated using the equation of state for sea water density[Millero and Poisson, 1981]. In short, increased temperature in water results in decreased water density, and increased salinity results in increased water density.

Two examples of a density profile is shown in table 4.3. In weak current waters, the density will constantly increase with the depth, as the heavier density water will sink and leave a low density layer on top of the water. This can cause issues if the densities of the top layer of water is smaller than the BV's minimum density, it can get "stuck" and unable to move up to the surface. This have the opposite effect for increased depth, as the BV might not be able to dive deeper than certian depths if the water density is larger than the maximum density of the BV.

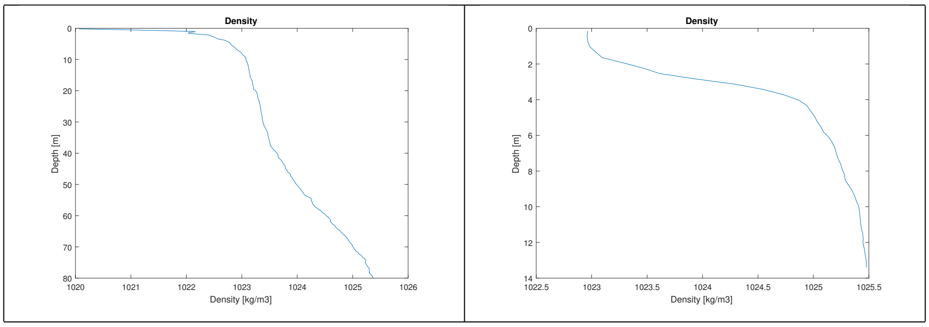


Table 2.1: Two examples of density profiles. The density usually increases with depth, which can cause the BV becoming unable to reach the surface or make it unable to dive deeper than certain depths.

Whenever the BV is stationary vertically in a fluid, means density is equal of the vehicle and its surrounding fluid. This can be exploited by for instance measuring the temperature of the water, to find an estimate of the salinity.

2.1.4 Stability

Looking at the vertical stability of an object immersed in a fluid, the two main acting forces, gravitational and buoyancy, will have their center of gravity and center of buoyancy at relative positions depending on the properties of the object. The center of gravity is determined by the density displacement of the object, in other words, the average center of the mass. The center of buoyancy is determined by the geometric shape of the object, e.g. the average volume center. To achieve vertical stability, the center of gravity should be below the center of buoyancy, thus generating a restoring force that will keep the object in its initial position, figure 2.1 shows an example of an unstable- and stable object. This does not take into account the possibility of rotating from drag forces when moving. But in this project it is assumed slow movement speeds, so this influence can be assumed negligible on the vertical stability.

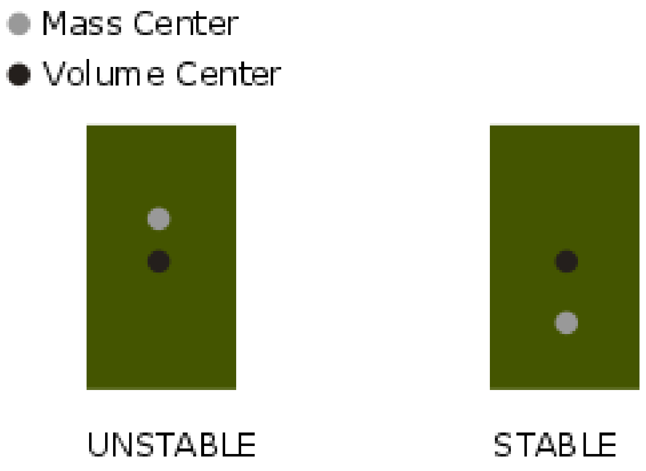


Figure 2.1: If the mass center is above the volume center, then it will result in an vertically unstable system underwater. If the mass center is below the volume center will result in a vertically stable system underwater.

2.2 Mathematical Modelling

2.2.1 Vertical Motion

In this paper the references is based on depth, meaning moving from 3 m depth to 10 m depth requires a net positive velocity. Inserting this convention makes it possible to express the vector based forces as scalar forces. Consequentially, the gravitational force will always work in a positive direction and the buoyancy force in a negative direction. $F_D(t)$ is defined to have a positive value when there is a positive velocity, and negative with a negative velocity. To determine the vertical motion of an object submerged in a fluid, Newton's 2nd law is used:

$$F_{sum}(t) = ma(t) = F_G - F_B(t) - F_D(t) + F_E(t) \tag{2.5}$$

Here is $F_E(t)$ defined as environmental forces, such as ocean currents or waves generating a lift or fall in the vertical direction. These forces will not be measured on site, and therefore be considered as a random noise. By solving the equation for $a(t)$, the velocity and position can be found by integrating:

$$a(t) = \dot{v}(t) = \ddot{x}(t) = \frac{F_G - F_B(t) - F_D(t) + F_E(t)}{m} \tag{2.6}$$

$$x(t) = \iint \frac{mg - \rho_f g V(t) - \frac{1}{2} C_D \rho_f v(t)^2 A + F_E(t)}{m} \, dt \, dt \tag{2.7}$$

2.2.2 Volume Model

The BV's volume can be manipulated by moving a smaller piston, which has the ability to insert or remove water from it. Total volume over time of the BV can be expressed as:

$$V(t) = V_{max} - x_{piston}(t) A_{piston} \tag{2.8}$$

where x_{piston} is the linear displacement of the piston head, and V_{max} is defined as the total volume of the BV when $x_{piston} = 0$, and V_{min} is the minimum volume of the BV when the piston is at its maximum value, $x_{piston} = h_{pistonMax}$. Defining the piston movement in this direction ensures positive relation between the depth and the piston-position, e.g. positive increase in piston-position results in a positive increase in depth. Figure 2.2 displays the piston with this definition.

The minimum amount of the piston volume (V_{piston}) can be found from the total buoyancy mass, and the wanted maximum and minimum densities of the BV. The maximum and minimum densities is defined as:

$$\rho_{max} = \frac{m}{V_{min}}, \; \rho_{min} = \frac{m}{V_{max}} \tag{2.9}$$

The interval between ρ_{max} and ρ_{min} is equivalent to the range the BV must be able to change its density, and can be written as:

$$\rho_{max} - \rho_{min} = \frac{m}{V_{min}} - \frac{m}{V_{max}} = \frac{m(V_{max} - V_{min})}{V_{max} V_{min}} \tag{2.10}$$

Because $V_{piston} = V_{max} - V_{min}$ and $\frac{m}{V_{max} V_{min}} = \frac{\rho_{max}\rho_{min}}{m}$, equation 2.10 can be rewritten to:

$$V_{piston} = m \frac{\rho_{max} - \rho_{min}}{\rho_{max}\rho_{min}} \tag{2.11}$$

If it is assumed the BV is to operate only in sea water, the density "extremes" is approximately $\rho_{min} = 1016 \frac{kg}{m^3}$; $T = 25°C$, $S = 20$ and $\rho_{max} = 1028 \frac{kg}{m^3}$; $T = 4°C$, $S = 35$. V_{piston} can be approximated using these values.

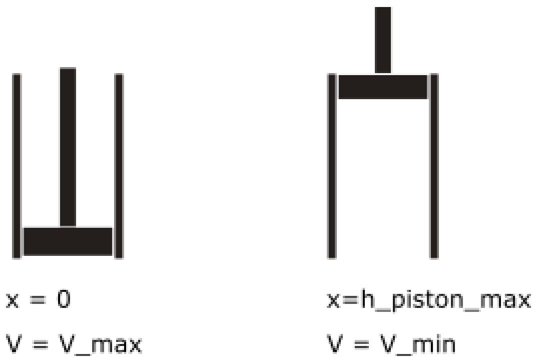


Figure 2.2: The piston position is defined to $x_{piston} = 0$ when the total volume of the BV is at its maximum, and the maximum possible piston position $x_{piston} = h_{pistonMax}$ when the total volume of the BV is at its minimum

2.2.3 Payload

To compensate for additional vertical forces from a payload attached to the BV, the total volume of the BV needs to adjust/increase. This achieved by extending the height of an outer lid from the position the BV is considered neutrally buoyant. The additional buoyancy force from the outer lid requires to be equal to the gravity and buoyancy force of the payload to compensate;

$$\rho_f g V_{add} = m_{load} g - \rho g V_{load} \tag{2.12}$$

This can be solved for additional height of the outer lid;

$$h_{add} = \frac{1}{\pi r_{vehicle}^2}(\frac{m_{load}}{\rho_f} - V_{load}) \tag{2.13}$$

Many sub-surface instruments have their weight in fresh water documented, which makes the volume of the payload unnecessary in the equation:

$$h_{add} = \frac{m_{load_{fw}}}{\rho_f \pi r_{vehicle}^2} \tag{2.14}$$

Using $m_{load_{fw}}$ creates a small error if used in for instance sea water, depending on the volume of the payload.

2.3 PID Controller

A PID controller works as a type of controller in a feedback control system. The PID controller continually takes in the difference between a controllable objects current state value and a desired reference state value. The input difference is known as the error, and the purpose of the PID controller is to minimize this error. Here a state is defined to some sort property belonging to the object that can change over time, for instance, its temperature, position, or volume. The PID controller's output is connected to a variable in the system that is able to alternate the state value of the object.

The computed output value is based on the sum of three components; Proportional, Integral and Derivative, which can be tuned to minimize the error of the system. A typical block diagram of a PID controller can be seen in figure 2.3, and an example of a feedback control system can be seen in figure 3.4.

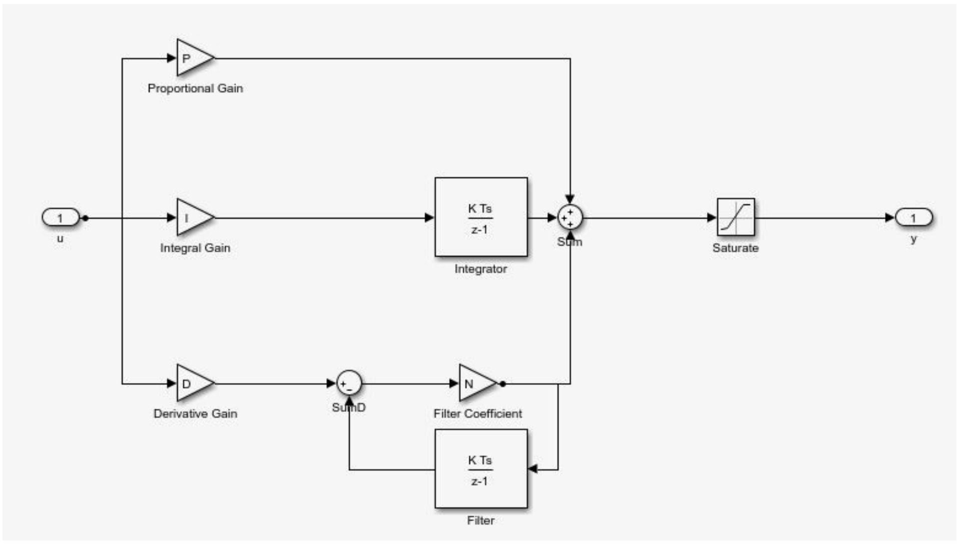


Figure 2.3: Block diagram of a PID controller in parallel form, where the calculated output is a sum of three factors; Proportional, Integral and Derivative gain

2.3.1 Proportional

The proportional component multiplies the error with a constant gain, which adds a proportional value to the output. Too large gain can result in unstable systems, and too small gain can result in slow response time. When tuning, the proportional gain should usually be found to a satisfactory value before proceeding to the other parameters.

2.3.2 Integral

The integral part represents the sum of errors over time. It is meant to remove errors that propagates over time, e.g. it can remove errors when the system is in a steady state.

Because its primary task is to remove the offset in a steady system, it can be advantageous to not include the gain until the error is at a small threshold.

2.3.3 Derivative

The Derivative term considers the rate of change of the input error, and adjusts the output to minimize or remove oscillations on the system. Too large gain here can cause instability on the system when there is much random noise in the error signal, while too small gain might not remove oscillations.

2.4 Actuator

An actuator is a mechanical device for moving or controlling something. This section will focus on electrical DC motors, or stepper motors more specifically, which can be used as an actuator. This section's theory is based on the book ”Modeling and Simulation for Automatic Control”[Egeland and Gravdahl, 2003].

2.4.1 Electrical Motors

Rotating electrical motors consists of a stationary part called the stator and a rotating part called the rotor. In hybrid stepper motors the stator is build up by several stator poles able to create a magnetic field from coils wrapped around it. The rotor is permanently magnetized and will be rotated by alternating the magnetic field of the stator poles.

A shaft connected to the rotor will rotate with the angular velocity $\omega_{mot}[\frac{rad}{s}]$ and a motor torque $\tau_{mot}[Nm]$. Torque is a measurement of rotational force, where the maximum torque for most stepper motors will depend on the rotation speed, often given in a Torque-Speed curve.

Many applications requires more torque than the motor can deliver. A reduction gear increase the torque at the cost of decreasing the velocity. The gear has a gear ratio n, and the resulting output torque and velocity with a motor attached as input on the reduction gear, is given by:

$$\omega_{out} = n\omega_{mot} \tag{2.15}$$

$$\tau_{out} = \frac{1}{n}\tau_{mot} \tag{2.16}$$

2.4.2 Actuator Force

The force a rotating motor is able to push in a linear direction, as described in section 3.3.1, can be estimated by using the output motor torque (τ_{out}), motor efficiency (e) and the pitch of the threaded bolt ($p_{bolt}[m]$)[Nanotec]:

$$F_A = \frac{2\pi\tau_{out}e}{p_{bolt}} \tag{2.17}$$

Efficiency of an electric motor is the mechanical output power divided by the eletrical input power on the motor ($P_{mot} = VI[W]$), making it a number between 0-1:

$$e = \frac{\tau_{out}\omega_{out}}{P_{mot}} \tag{2.18}$$

Equation 2.17 and 2.18 can be put together, and alternatively get:

$$F_A = \frac{2\pi\tau_{out}^2\omega_{out}}{p_{bolt}P_{mot}} \tag{2.19}$$

It is important to notice that doubling p_{bolt} doesn't necessarily halve F_A, as the motor efficiency might change with it. In general, increased loads on the motor increases the motor efficiency. Estimation will therefore rely much on experimental results, and theoretical calculations of the required actuator force should have a large surplus.

The actuator force must be larger than the gauge pressure on the piston area at maximum depth (d_{max}) plus additional friction forces (F_F) to always push the piston head:

$$F_A > F_{gauge} + F_F \tag{2.20}$$

$$\frac{2\pi\tau_{out}^2\omega_{out}}{p_{bolt}P_{mot}} > A_{piston}d_{max}\rho_{max}g + F_F \tag{2.21}$$

2.5 Digital Filter

Digital Filters can be a useful improvement for smoothing out noisy signals, without needing additional physical components[1].

2.5.1 Exponential Filter

An exponential filter is a recursive filter, meaning it uses the last smoothed value (y_{n-1}) and the current measurement (x_n) to calculate a new smoothed value (y_n):

$$y_n = wx_n + (1 - w)y_{n-1} \tag{2.22}$$

The weighting parameter w controls the amount of smoothing on the filter, with a value between 0-1. A weight of for instance 0.1 gives high smoothing, but responds slower to measurement changes , while a weight of 0.9 have lower smoothing on the signal, but responds quicker to measurement changes. This filter is easily implemented and does not require much computing power, ideal for real time microcontrollers.

[1]Physical filters, like RC filter, is often helpful or needed to have in addition to the digital filter

Chapter 3

Method

3.1 Design Concept

The general principles and concepts chosen for constructing the BV is briefly listed here.

Volume Change Mechanism

Volume change will be achieved by an electrical linear actuator changing the volume of a smaller piston, which has the ability to insert or remove ambient water from it. A battery pack is used to power the linear actuator and other electrical components.

Shape and Material

The outer layer of the BV will be constructed in a cylinder shape, which is optimal for withstanding high pressures. Rigid PVC is chosen as the constuction material.

Control Strategy

A pressure sensor is used to read current depth of the BV. A microcontroller uses a PID controller to determine the required volume of the piston to move the BV to a defined set-point. The PID controller will be tuned through simulations of the system.

3.2 Construction of Prototype

3.2.1 Calculating Dimensions

Matlab was used to calculate the required weight that would give the appropriate density of the BV, by adjusting the radius and height of the cylinder shape. The radius is primarily limited by the size of the stepper motor and the battery pack, and the height requires at least two times the height of the piston in addition to the height of the motor and wall thickness.

Required piston volume is dependent as a percentage of the total volume and weight of the BV, which was estimated using equation 2.11. The piston radius was minimized to decrease gage pressure under operation.

There were too many unknown variables to determine the exact final weight of the BV. An estimate of the heaviest and most significant components were made in Matlab, and further increased estimate by adding components in Solidworks. To exactly fit the required total density of the BV, the outer lids placement was used to adjust the volume of the BV.

3.2.2 Solidworks Parts

A 3D model was implemented in Solidworks of the complete BV, for the purpose of visualizing sizes, creating drawings with dimensions for construction, and verifying mass and volume properties. The design revolved primarily around obtaining the required volume and mass, but secondly the BV was built with the ease of reassembly for prototype adjustments.

The BV was constructed in these separable parts, figure 3.1 shows a cross section of the assembeled parts in SolidWorks:

- **Vehicle House** is designed to hold the necessary components while withstanding the water pressure on the outside. It can be considered the mainframe of the construction, and defines all other parts dimensions. At the bottom of the vehicle there is an opening to let water into a smaller cylinder shape, known as the piston. The water is blocked by a piston head, which is connected to a linear actuator. This enables the piston head to move up and down, filling or removing water from the piston, or in other words, changing the volume of the vehicle. The top half on the outside is threaded to connect the Outer Lid.

- The **Outer Lid** is screwed onto the outside of the Vehicle House and seals the water out with an O-ring packing. It is constructed to be placed at various heights. This enables the BV to adjust its volume to compensate for additional payloads that can be attached to the BV, such as water quality sensors or CTDs. The O-ring packing also ensures the outer lid does not rotate easily when put on the vehicle house, but thus will require a special tool to rotate it. A small valve is also placed on top of the outer lid, to let air pressure out that is built up when mounting the outer lid on the vehicle house. The valve needs to close before it is put into water.

- **Inner Lid** functions primarily as a mount for the stepper motor. The stepper motor is attached on the bottom of the Inner Lid with four actuator screw holders. A microcontroller is also attached on the top of the Inner Lid. Three small holes were made to get wires inside the Vehicle house connected to the microcontroller.

- A custom made **Battery Pack** was constructed to fit the batteries in a convenient and space-efficient way around the piston. Alternating positive and negative battery contacts were mounted to create a serial connection of the batteries.

Drawings of the separable parts were created and delivered to the mechanical workshop, ”Kybernetisk Verksted”, where the prototype was made, see Appendix A for SolidWorks drawings.

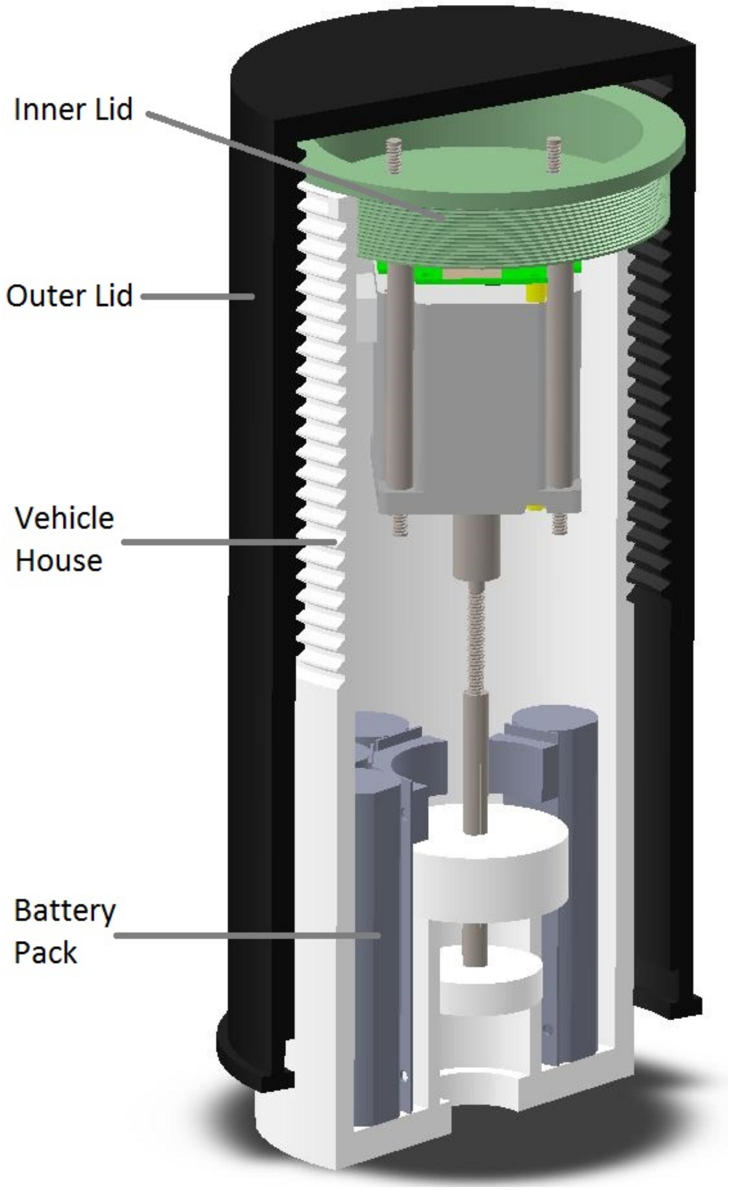
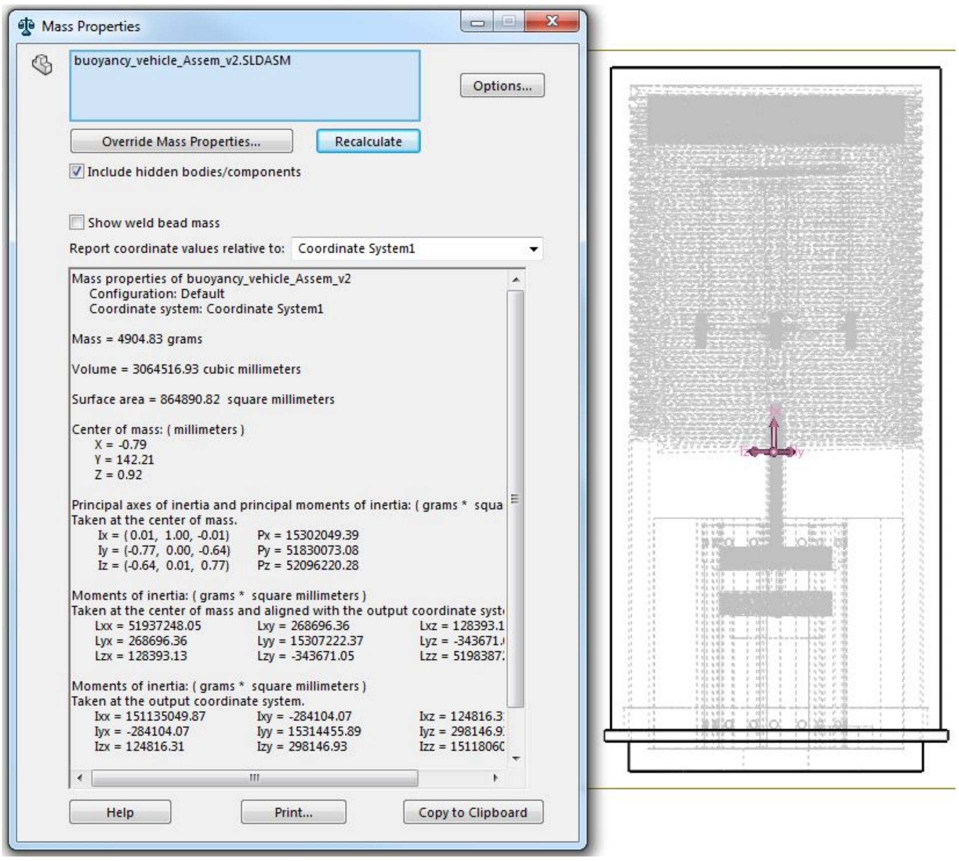


Figure 3.1: Cross section of BV's different parts in Solidworks. Vehicle House is the mainframe of the construction, and contains the piston on the bottom. The Outer Lid seals the water and can be placed at various positions to adjust the volume of the BV. Inner lid functions primarily as a mount for the stepper motor. The battery Pack is custom made to fit 16 C-type batteries around the piston. The gap in the battery pack is to allocate space for the pressure sensor.

3.2.3 Mass estimate/Stability

After the materials or masses of the individual components were defined, the mass properties was found in Solidworks, using the ”Mass Properties” function. As seen in figure 3.2, the center of mass in the vertical axis is located 142.21 mm from the bottom of the vehicle, which is 45.4 % of the total height of the BV. Considering the cylinder shape of the BV, the vertical volume center will be slightly above half of the total height. This means the construction will have a mass center below the volume center, and in theory should maintain vertical stability.



[✓ Include hidden bodies/components]

Show weld bead mass

Report coordinate values relative to: Coordinate System1

```
Mass properties of buoyancy_vehicle_Assem_v2
    Configuration: Default
    Coordinate system: Coordinate System1

Mass = 4904.83 grams

Volume = 3064516.93 cubic millimeters

Surface area = 864890.82  square millimeters

Center of mass: ( millimeters )
     X = -0.79
     Y = 142.21
     Z = 0.92

Principal axes of inertia and principal moments of inertia: ( grams *  squa
Taken at the center of mass.
      Ix = ( 0.01,  1.00, -0.01)      Px = 15302049.39
      Iy = (-0.77,  0.00, -0.64)      Py = 51830073.08
      Iz = (-0.64,  0.01,  0.77)      Pz = 52096220.28

Moments of inertia: ( grams *  square millimeters )
Taken at the center of mass and aligned with the output coordinate syst
     Lxx = 51937248.05          Lxy = 268696.36           Lxz = 128393.1
     Lyx = 268696.36            Lyy = 15307222.37         Lyz = -343671.
     Lzx = 128393.13            Lzy = -343671.05          Lzz = 5198387

Moments of inertia: ( grams *  square millimeters )
Taken at the output coordinate system.
     Ixx = 151135049.87         Ixy = -284104.07          Ixz = 124816.3
     Iyx = -284104.07           Iyy = 15314455.89         Iyz = 298146.9
     Izx = 124816.31            Izy = 298146.93           Izz = 1511806
```

Figure 3.2: ”Mass Properties” function in Solidworks to determine the mass center. It indicates the mass center is placed at 45.4 % of the total height of the BV in a vertical position.

3.3 Linear Actuator

A Linear Actuator is used to create movement in a straight line, and in this application used to move the piston position.

3.3.1 Rotation to Linear Movement

The piston is driven by a stepper motor, which is mechanically configured into a linear actuator, see figure 3.3. This translation is achieved by connecting a lead screw to the rotating shaft of the stepper motor. The lead screw can be screwed into a holster that is threaded inside. On the outside of the holster there are three ingraved tracks running in a straight line parallel to the lead screw. These tracks match inside a fixed housing cap on the top of the piston house, disabling rotation on the holster. At the end of the holster a piston head is placed, which fits inside the piston house.

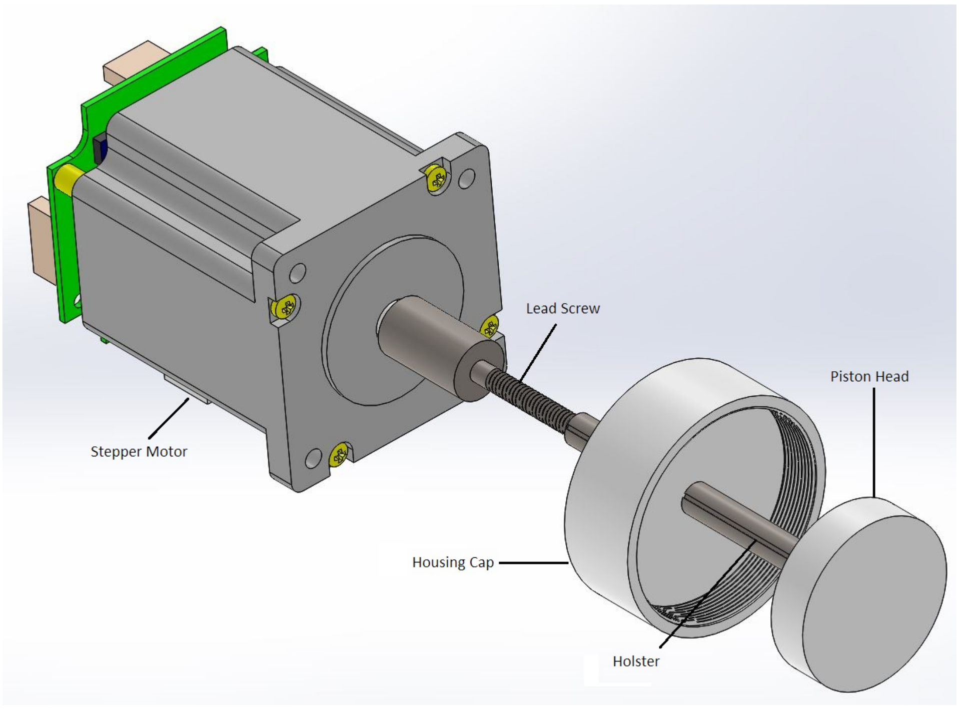


Figure 3.3: Converting rotation from the shaft of a stepper motor into linear movement on the Piston Head. The Lead Screw is able to rotate in a Holster. The Holster is mechanically locked from rotating by the Housing Cap, and will therefore move in a linear direction when the Lead Screw is rotating.

3.4 Simulations

The mathematical models of the system was implemented and simulated using Matlab and Simulink. The purpose of creating a model and simulating the system is primarily to determine the PID control parameters and determine if the system is controllable.

The systems dynamic is divided into two blocks, piston dynamics and vehicle dynamics. The simulation is implemented as a feedback loop, where the measured output value is subtracted from a reference value, to form a resulting error. This error is taken into the PID controller, who determines the optimal piston position. Figure 3.4 shows an overview of the system implemented in Simulink.

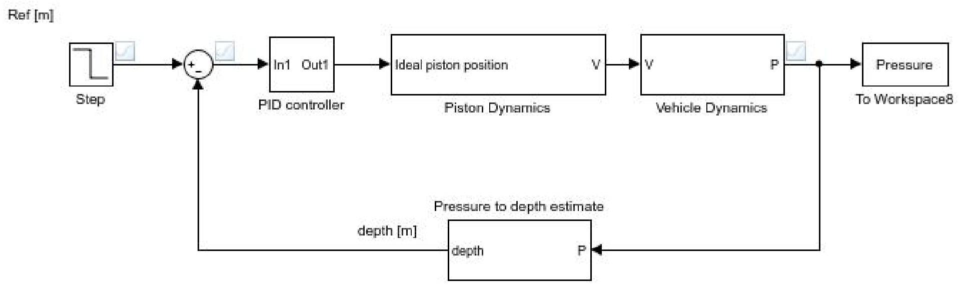


Figure 3.4: Overview of system in Simulink, implemented as a feedback loop. ”Step” block determines the reference depth the BV is supposed to move to. ”Pressure to depth estimate” block is a calculation between the current measured pressure to an estimated depth. The systems dynamics is divided into two blocks, piston dynamics and vehicle dynamics.

3.4.1 Piston Dynamics

The piston dynamics block is aimed to emulate how a linear stepper motor moves a piston head, and thus changing the volume of the vehicle. Figure 3.5 is a block diagram of the implemented piston dynamics in Simulink. Its input is the ideal piston position, which relates to a reference search algorithm in the stepper motor. The velocity is implemented to simulate how the search algorithm on the stepper motor is implemented. This algorithm searches a reference point with maximum velocity in the correct direction, until it hits a set threshold distance from the reference point, where the velocity is set to a quarter of the maximum velocity in the same direction until the reference point is found [Trinamic, 2014].

By integrating the velocity, the new position is found. This position is limited by the height of the piston house and is quantized because of the step size in the motor. The stepper motor has 51,200 steps per full rotation, with a pitch of 1 mm on the lead screw adds up to a resolution of 51,200,000 steps per meter, or 3,072,000 steps in the applicable area.

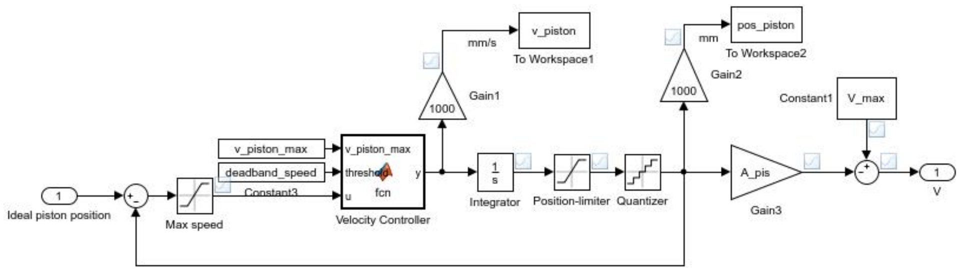


Figure 3.5: Implementation of piston dynamics in Simulink. The piston velocity is defined from a reference seach algorithm, and the piston position found by integrating the velocity.

3.4.2 Vehicle Dynamics

Vehicle Dynamics block is an implementation of the vertical motion and pressure sensing of the BV, figure 3.6 shows the implemented block diagram in Simulink. The water density is calculated by interpolating previously measured temperature and salinity data at given depth, then using equation of state for sea water density[Millero and Poisson, 1981] to simulate a density profile. A surface-check function block ensures that only the volume that is submerged in water creates buoyancy.[1] Equation 2.7 is implemented from the given volume, density and velocity to give the new depth of the BV. A sub block simulates a pressure sensor, and returns a pressure at the given depth. The pressure sensor is quantized based on the ADC resolution and contains a small white-noise disturbance to imitate random noise when measuring.

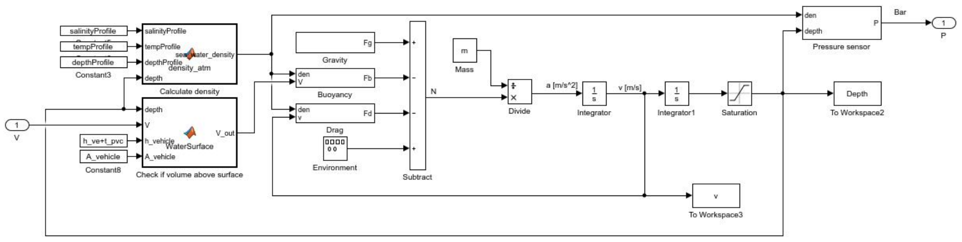


Figure 3.6: Implementation of Vehicle dynamics in Simulink. The vertical motion of the BV is simulated by implementing equation 2.7. "Pressure sensor" block detects the pressure at given depth to simulate a pressure sensor.

3.5 PID tuning

After implementing the simulations in Simulink, the PID controller parameters was tuned by adjusting the Proportional, Integral and Derivative terms, and then observe the simulated response of the BV. The simulated setup contained of releasing the BV from the

[1]Buoyancy from air density is assumed negligible

water surface with the piston position at minimum (minimum vehicle density), then regulate its position to become stationary at a depth of 5 m. After 300 s, the BV is set to move and become stationary at 2 m depth. The graphs in table 3.1 shows the vertical position of the BV as the solid blue line, and the reference depth as the dotted red line.

The process of tuning the PID controller consisted by first finding a stable proportional gain (Kp), while the integral and derivative gains (Ki and Kd) where set to 0. The integral gain was then implemented with a threshold of 2 m, i.e. if the BV is within 2 meters of the reference value the integral term will start to work. Then finally a derivate gain was found and added to remove stationary oscillations.

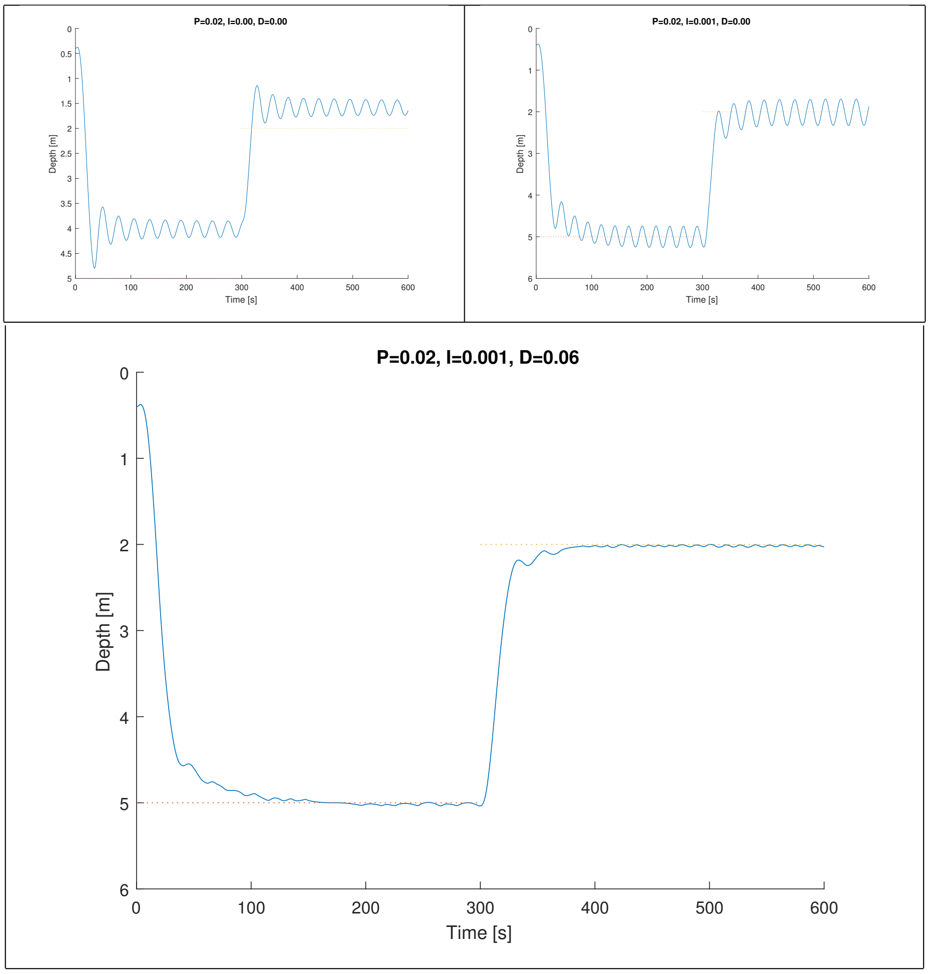


Table 3.1: PID tuning of the BV in simulations, accomplished in three steps; Top left figure is the response of the BV when tuning using only proportional gain, top right figure is tuning with both proportional and integral gain, and bottom figure when tuning with proportional, integral and derivative gain.

3.6 Power Estimate

The power used during operation can be estimated by looking at the duration the stepper motor is activated, which is the largest factor for the power consumption. As long as the stepper motor enters a low power mode when it's inactive, methods can be used to minimize the piston movement to increase the BVs energy efficiency.

Some methods to minimize the piston movement will be gone through here. The methods efficiency will be compared as a percentage of maximum power consumption, which is continuously moving the piston the entire simulation. Simulations will be performed with the same setup as in section 3.5, which uses 44.3% of maximum power consumption without any additional implemented methods.

3.6.1 Piston Position Threshold

To avoid small incremental movements, a threshold on the piston position is implemented, i.e. if the absolute value between the piston position and the new piston position is less than a set threshold, it doesn't move the piston position at all. The result of using this method is implemented in figure 3.7, where the threshold it set to 0.3 mm. Using this method reduced the power estimate to 29.5% of maximum power consumption, but decreases the stability on the setpoint slightly.

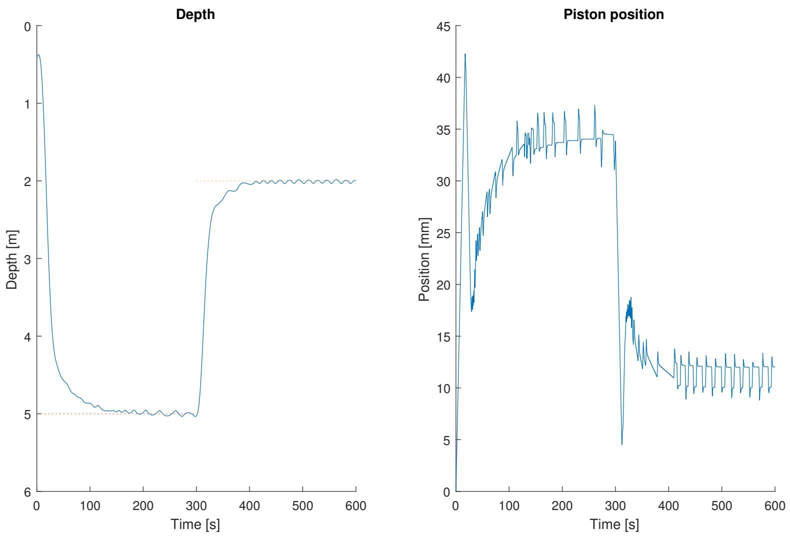


Figure 3.7: BV Simulated Depth and Piston Position with 0.3 mm Threshold on Piston Position

3.6.2 Pressure Sensor Resolution

The pressure sensor used in this project have a resolution of approximately 8.4 cm/step with a 10-bit ADC. By using a 14-bit ADC instead, the power estimate goes to 31.2 % of maximum power consumption, while achieving increased stability on the setpoints, see figure 3.8.

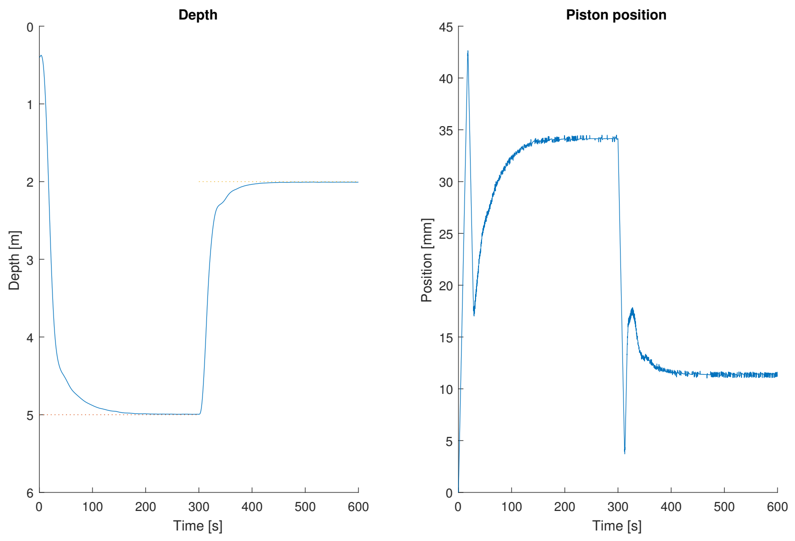


Figure 3.8: BV Simulated Depth and Piston Position with 14-bits ADC

3.6.3 Combined

By combining these two methods, the power estimate goes to only 12.6 % of maximum power consumption while still remaining stable around the setpoint, see figure 3.9.

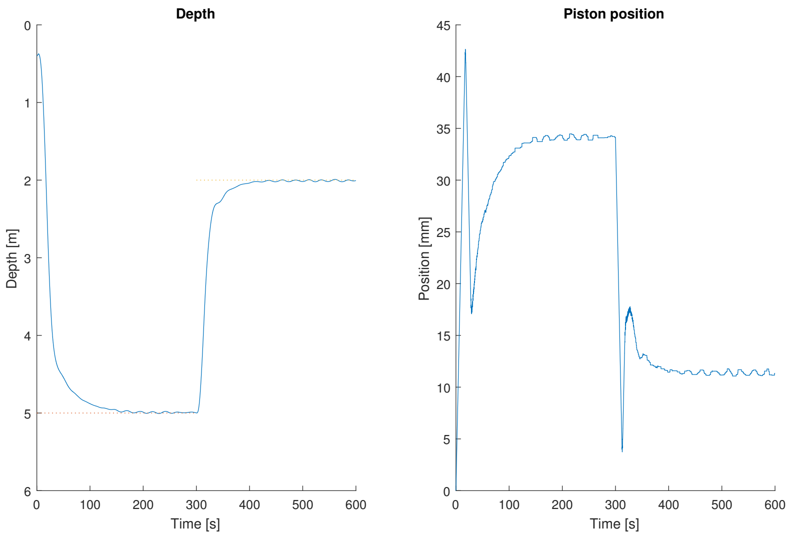


Figure 3.9: BV Simulated Depth and Piston Position with combined energy efficiency methods

3.7 Hardware Design

This section attempts to show what electrical components where used, how they were implemented, and what purpose they serve.

3.7.1 Overview System

Figure 3.10 illustrates the overall hardware setup used in this project. It accomplishes the following functions:

- Providing a regulated and monitored power supply
- Accurate rotational movement with high torque
- Readings of ambient pressure
- Microcontroller capable of running control algorithm
- Data storage

3.7.2 Microcontroller

An Arduino Nano is used as a microcontroller. Arduino is an open source developer for general purpose microcontrollers, often suited as a platform for prototyping. The Arduino Nano is composed by Atmels ATmega328 microcontroller, which contains 12 digital- and 8 analog I/O pins. It also supports necessary communication protocols, such as UART, SPI and I2C. The Arduino Nano can be programmed directly through a USB cable, and programs written in the Arduino IDE. This provides a solution that is easily implemented with other hardware components, suited for prototyping projects.

3.7.3 Power Supply and Voltage Regulator

The BV have its electrical power supplied by a custom built battery pack. The battery pack consist of 16 series coupled C type alkaline batteries (ENERGIZER EN93), 1.5 V each, adding up to 24 V fully charged. As the power is consumed, the battery pack will discharge voltage, where it will drop out (become empty) at about 12.8V. A MC7812CT voltage regulator was implemented to enable a steady voltage supply at 12 V to the stepper motor. A pull-down capacitor was placed after the voltage regulator to avoid voltage spikes that can potentially damage the motor and microcontroller. The microcontroller uses its integrated voltage regulator to pull the 12 V down to 5 V.

3.7.4 Voltage Divider

This block is used to read the raw battery voltage, before regulating it down to 12 V. This enables the microcontroller to sample the battery voltage over time, giving an indication of the power consumption of the total system, and safely return the BV to surface before the battery becomes empty. But reading analog voltage on the microcontroller requires voltage

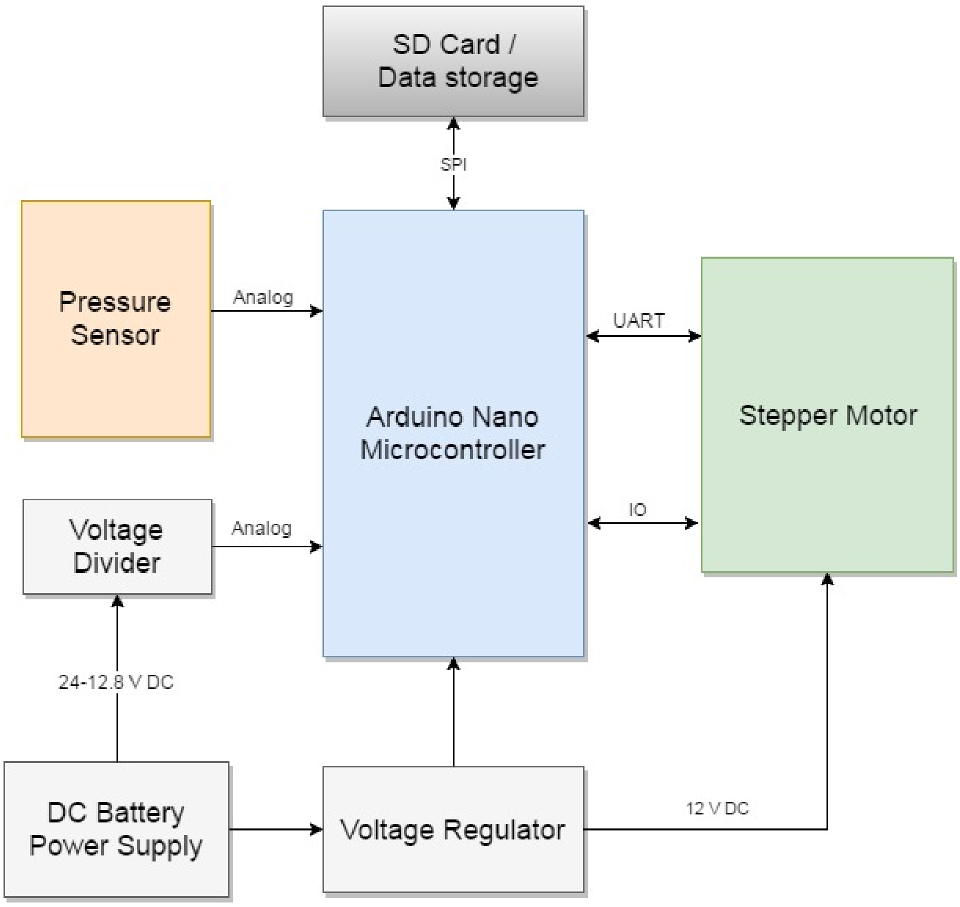


Figure 3.10: Overview of the hardware components used to control the BV, and their communication protocols.

between 0-5 V. To solve this, two series coupled resistors is connected in parallel to the battery voltage line. Figure 3.11 shows this setup, where $R1 = 10M\Omega$ and $R2 = 1M\Omega$. The connected analog pin will now read $\frac{1}{11}(\frac{R2}{R1+R2})$ of the original battery voltage, e.g. 24-12.8 V on the battery will translate to 2.18-1.16 V on the analog pin. The resolution on the battery voltage with this setup is 0.0536 V.

3.7.5 Pressure Sensor

The chosen pressure sensor is from the HoneyWells PX3 Series Heavy Duty Pressure Transducers. It uses a piezoresistive element to measure the ambient pressure, which is generated to an analog signal. The absolute pressure ranges from 101.3 kPa (1 atm) to 790.8 kPa, or approximately from 0 to 68.6 meters depth in sea water. Calibration is set to sealed gauge, e.g. minimum output on the analog signal translate to 1 atm absolute

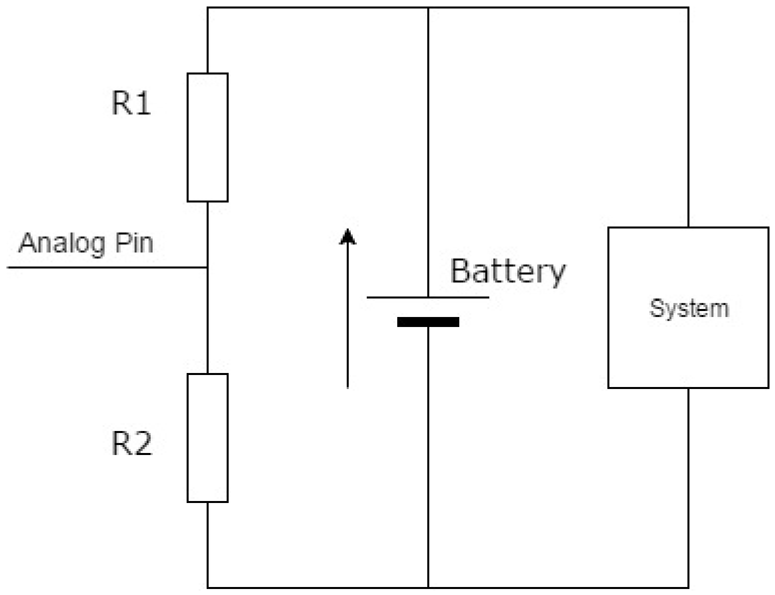


Figure 3.11: The Voltage Divider makes the Analog Pin read a fraction of the raw battery voltage. This makes it possible to read the raw battery voltage on the ADC.

pressure.

The 10-bits ADC on the microcontroller is used to read the pressure value. As the pressure sensor has a ratiometric output from 0.5 V to 4.5 V, the resolution per depth on the ADC is approximately 8.4 cm/step.

3.7.6 Stepper Motor

Trinamic Motion Control GmbHs PANdrive PD60-3-1161 was chosen as the stepper motor for the system. It is an integrated high-torque stepper motor with a motion controller and driver attached. Some of its integrated features includes:

- Absolute sens0step encoder with 1024 points per rotation, translating to 51,200 steps per full revolution on the motor shaft, useful for ensuring accurate position and less requirement for calibration
- coolStep sensorless load dependent current control, able to adjust the current consumption based on the required load. It can also be programmed to a standby current of 0A after 10 ms of inactivity
- Communication interfaces includes USB, RS232, RS485, and general purpose I/Os
- 10-30V DC input power supply / nom. 24V DC

The characteristic speed-torque curve of the stepper motor is given in figure 3.12. Average rotation speed was estimated by timing 50 full revolutions, e.g. move the linear actuator 5 cm.

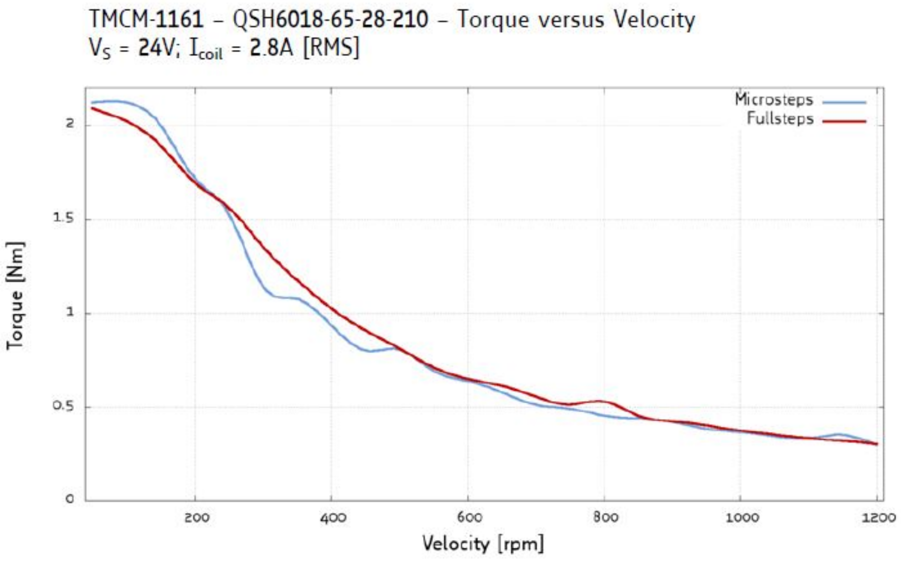


Figure 3.12: Torque vs. velocity characteristics at 24V / 2.8A of PD60-3-1161 stepper motor. Figure taken from datasheet "PD-1161 Hardware Manual"[Trinamic, 2013]

3.7.7 Data Storage

A standard SD-card module is used to read and write data to a non-volatile SD memory card. The module communicates with the microcontroller through SPI interface. This provides easily accessible data that is stored after the power turns off. The stored data is used to analyze experimental results, and is not used to read from during operation.

3.7.8 PCB

A PCB (printed circuit board) is essentially a customized board that contains lines and pads, connecting the electrical components and modules together. Its function is primarily to reduce or remove the need for wires between components. There are several software tools for designing PCBs, but EAGLE from Autodesk was chosen in this project. The components soldered on the PCB were the Arduino Microcontroller, SD-module, Voltage Regulator, and Voltage Divider. Additional pins were also added to connect the microcontroller with the motor, battery pack, and the pressure sensor.

3.8 Firmware

The firmware running on the Arduino Nano microcontroller was implemented through Arduino's integrated development environment (IDE). This platform conveniently have a large selection of open source libraries. It uses C as the programming language.

3.8.1 Flow Chart

Figure 3.13 shows the flow chart of the logic behind the implemented firmware. In ”Set run parameters”, the reference depths and run timers are set. ”Wait for assembly” stops the program and waits for a set timer, so the Outer Lid can be assembled before the program continuous.

Whenever a set sample time has passed, the program reads and updates the analog values of the pressure sensor and battery voltage. Then the PID controller computes a new ideal-piston position. The new piston position is sent to the motor driver, which changes the position. Piston position, battery voltage, depth, setPoint, and the time since startup is then stored to a SD card. This process continuous until the battery is low, or the set runtime is over. If this happens the BV is set to float to surface, setting its potential volume to maximum.

3.8.2 PID Controller

Implementing the PID controller was achieved using the ”Arduino PID library” [Beauregard].

3.8.3 Filter

Digital filters were added to the analog pressure and battery voltage signals. Both signals were implemented with an exponential filter. For the pressure signal, the smoothing can be set to a high value, as the maximum velocity of the BV is relatively low, and rapid movement changes should not naturally occur. This helps the derivative part of the PID regulator to work more efficiently, or with an increased impact.

As for the battery voltage signal, there is interest in reading a slow decaying battery voltage charge. Therefore the smoothing can be set to a high value here as well. The battery signal was also filtered with a moving average, post experimental results in Matlab.

The filters were implemented onto the microcontroller using the open library ”Filter.h” from ”MegunoLINK”[Meguno-LINK].

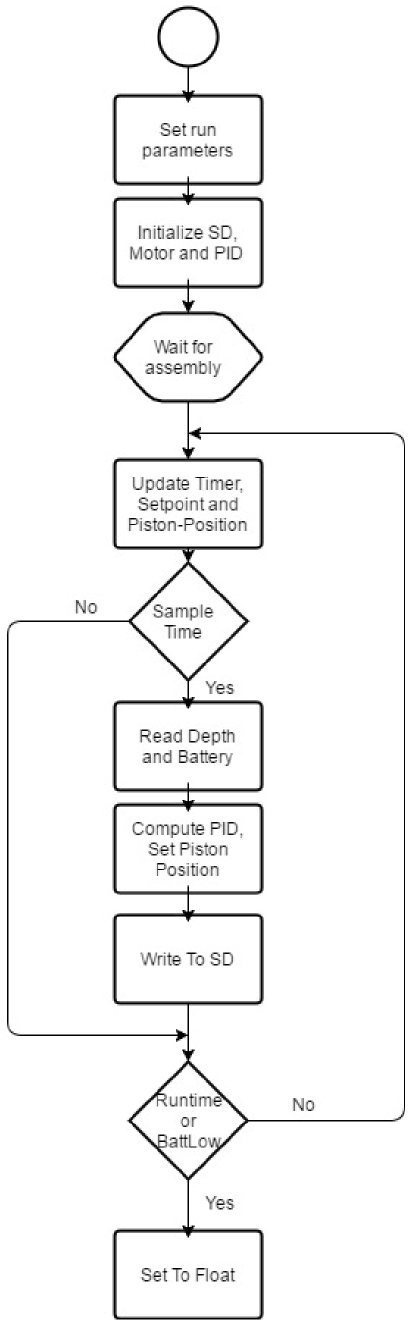


Figure 3.13: Flow Chart of the implemented firmware. After initialization, every Sample Time, the program reads two analog signals and then uses the PID controller to determine the piston position. It continuoues this process until the battery is low, or runtime is over.

3.9 Measurement Instruments

3.9.1 CTD

Sontek's Castaway-CTD was used to get a measurement of the density profile. The salinity accuracy of the instrument is 0.1 PSU and temperature accuracy is $0.05°C$.

3.9.2 Mulitmeter

A Amprobe AM-510-EUR is a multimeter used to measure battery voltages before and after the physical experiments.

Chapter 4

Results

4.1 Prototype Properties

Table 4.1 sums up the implemented properties of the BV. Figure 4.1 shows the BV being assembled, where the outer lid is to the left.

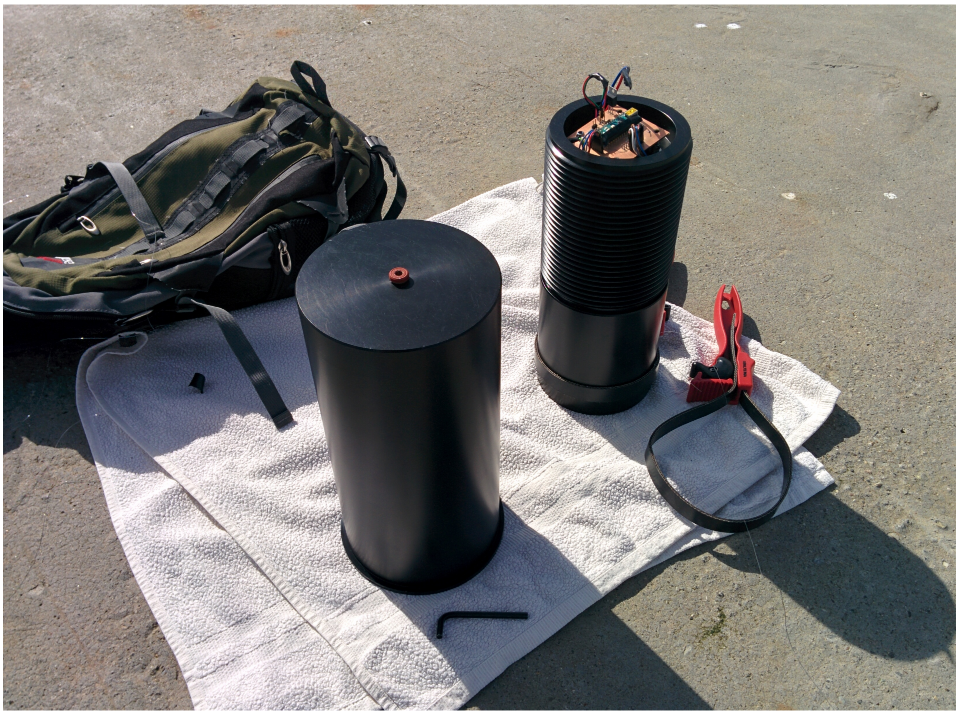


Figure 4.1: BV being assembled during physical experiment #1

Physical			Actuator/Piston		
Property	Value	Unit	Property	Value	Unit
m	6.101	kg	P_{mot}	12.0	W
V_{min}	5.930	dm^3	ω_{out}	15.32	rad/s
V_{max}	6.005	dm^3	v_{piston}	2.44	mm/s
V_{piston}	0.075	dm^3	p_{bolt}	1.0	mm
ρ_{min}	1016.0	kg/m^3	F_A	1022.3	N
ρ_{max}	1028.9	kg/m^3	d_{max}	80	m
$r_{vehicle}$	6.50	cm	r_{piston}	20.0	mm
r_{lid}	7.30	cm	h_{piston}	60.0	mm

Table 4.1: Final physical properties of the BV with outer lid position at 7.37 cm from the bottom of the BV.

4.1.1 Outer Lid Position

After construction was completed and all parts implemented, the BV was weighted to 6.101 kg. This is 0.818 kg more than the minimum required mass. The outer lid was moved to 7.37 cm from the bottom of the BV, to compensate and create neutral buoyancy on the BV. Additional placements of the outer lids position to compensate for different densities is given in table 4.2, where "$h_{add_{bot}}$" is the distance from the bottom of the BV to the bottom of the outer lid, and "rotations" is the amount of full rotations (from the top) the outer lid needs to reach the position.

$\rho_{min}[\frac{kg}{m^3}]$	$\rho_{max}[\frac{kg}{m^3}]$	$h_{add_{bot}}$ $[cm]$	Rotations
994.0	1006.4	8.37	$11 + 20°$
996.0	1008.4	8.27	$11 + 72°$
1000.0	1012.5	8.09	$11 + 186°$
1004.0	1016.6	7.91	$11 + 296°$
1008.0	1020.7	7.72	$12 + 45°$
1012.0	1024.8	7.54	$12 + 153°$
1016.0	1028.9	7.37	$12 + 260°$
1020.0	1033.0	7.19	$13 + 7°$

Table 4.2: Position of outer lid to compancate for various densities. $h_{add_{bot}}$ is the distance between the bottom of the BV to the bottom of the outer lid. Rotations is the amount of full rotations for the outer lid to reach the given position.

Payload

The maximum weight in fresh water ($m_{load_{fw}}$) of an attached payload to the BV is found from equation 2.14 to 0.857 kg. The maximum additional position on the outer lid is set to $h_{add} = 0.0633m$.

4.1.2 Linear Actuator Force

Using the speed-torque curve in figure 3.12, the output torque was estimated to $2.0Nm$ at the observed rotation speed ($\omega_{out} = 15.32rad/s$). The figure uses a maximum power input of $67.2W$, while the BV has a $12.0W$ input power to the motor. By assuming the power efficiency remains the same when changing the input power, the output torque will be $0.357Nm$. The linear actuator force on the piston was then estimated using equation 2.19 to be $1022.3N$.

Equation 2.21 was used to estimate the maximum depth to be $80.7m$. The maximum depth only considers the gage pressure on the piston, and not if the remaining construction of BV can withstand the gauge pressures it experiences. Simulation efforts or experimental data was not conducted to investigate this.

4.2 Hardware Design

The implemented PCB can be seen figure 4.2. The serial and IO busses from the stepper motor are connected respectively on the right and bottom pins in figure 4.2. The top three wires on the left pins (red, green and black) is the wires connecting the pressure sensor to the circuit. The remaining two wires on the left side (blue and red) connects the battery pack. The used microcontroller, Arduino Nano, can be seen in the middle of the figure, and a SD card module is soldered on the back side of the PCB.

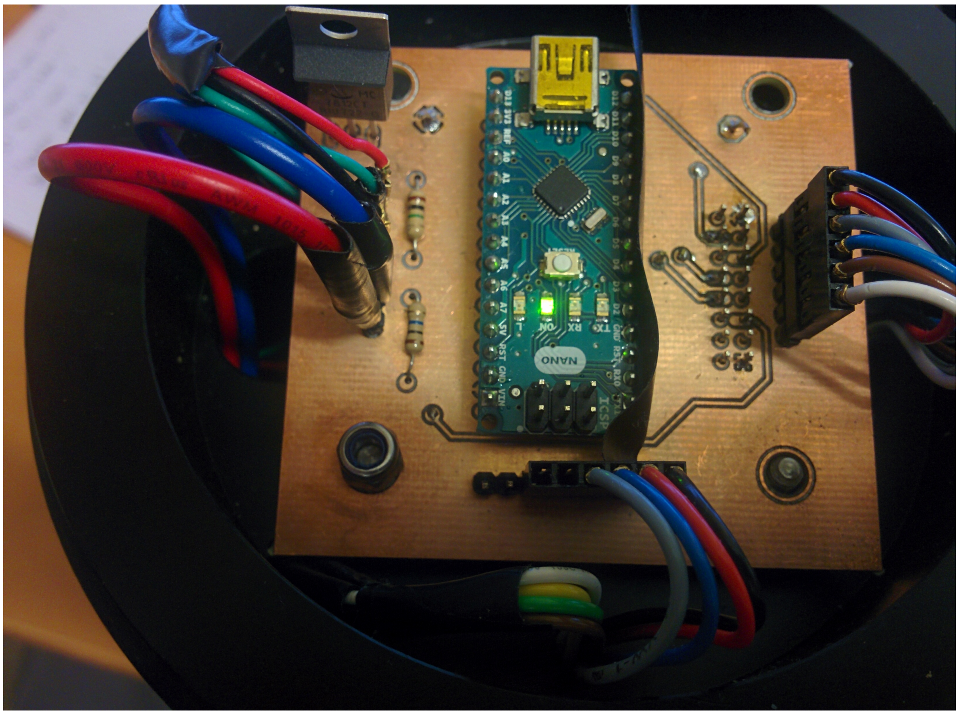


Figure 4.2: Photo of implemented PCB. Right and bottom pins are connected to the stepper motor. Top three wires on the left pins are connected to the pressure sensor, and the two last wires on the left side is connected to the battery pack.

4.3 Physical Experiment #1

4.3.1 Setup/Plan

The first physical experiment were executed from a dock on Børsa into Trondheimsfjoren. The maximum depth from the dock is approximately 12 m. Before starting the experiment, a CTD measurement was taken to find the density profile in the applicable area, and adjust the outer lid accordingly. The BV is then fasten with a fishing line to ensure the BV doesn't drift away horizontally, and as safety-feature to pull the BV out of the water if an error occurs.

Initially, the BV is to start floating on the surface, with maximum possible volume on the BV. Then it will start moving and regulating toward the first setpoint, at 10 m depth. After 5 minutes, the BV changes and moves to the second setpoint, at 2 m depth. It will cycle between 10 m and 2 m with 5 minutes intervals one more time.

4.3.2 Experimental Results

The experimental results can be seen in figure 4.3. The solid blue line in the graph "Depth" represent the measured depth by the pressure sensor, the dotted red line represents the set point, and the simulated response in the black dash-dot line.

"Piston Position" graph shows the pistons position during the experiment, where the position is defined as described in section 2.2.2.

Much wind and strong currents near the dock caused an inferior situation for conducting experimental tests. Currents moved the BV under the dock, and there was danger of the fishing line becoming tangled into a mooring. The fishing line was pulled on several times to ensure the BV wasn't stuck, which affected the dynamic behavior of the BV.

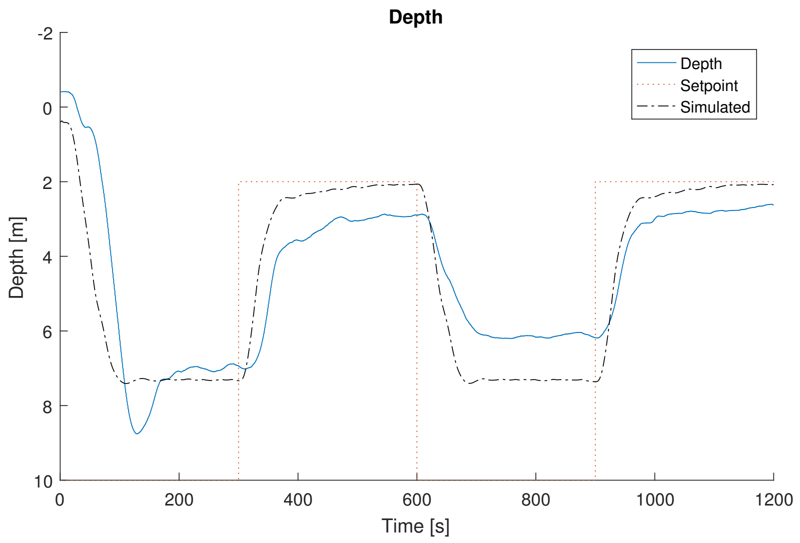
4.3.3 Density Profile

A density profile was conducted before the experiment started, and can be seen in figure 4.4.

The density of the ambient fluid was approximated when the BV remained vertically stationary (6m depth between t=700-900s and 3m depth between t=400-600s) by calculating the density of the BV. Table 4.3 shows the calculated density of the BV at the given depths, compared with the measured densities form the CTD.

Depth $[m]$	Approximated $[\frac{kg}{m^3}]$	Measured $[\frac{kg}{m^3}]$	Difference $[\frac{kg}{m^3}]$
6	1022.69	1022.20	0.49
3	1018.34	1017.82	0.52

Table 4.3: "Approximated" is the calculated density of the BV at the given depth. "Measured" is the measured density at given depth using a CTD probe one hour before the experiment.



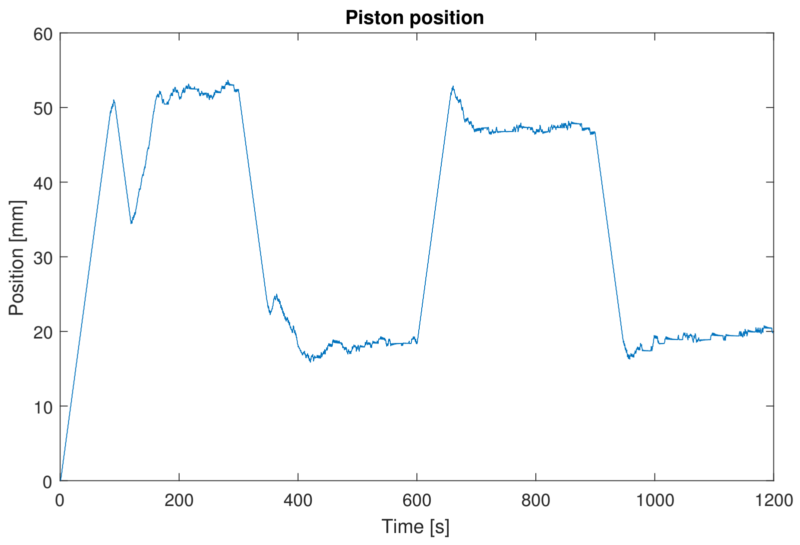


Figure 4.3: Experimental results of Physical Experiment #1. Top graph shows the logged depth of the BV in the solid blue line, setpoint in dottet red line, and the simulated response in dash-dot black line. Bottom graph shows the piston position. Both time axis are started simultaneously.

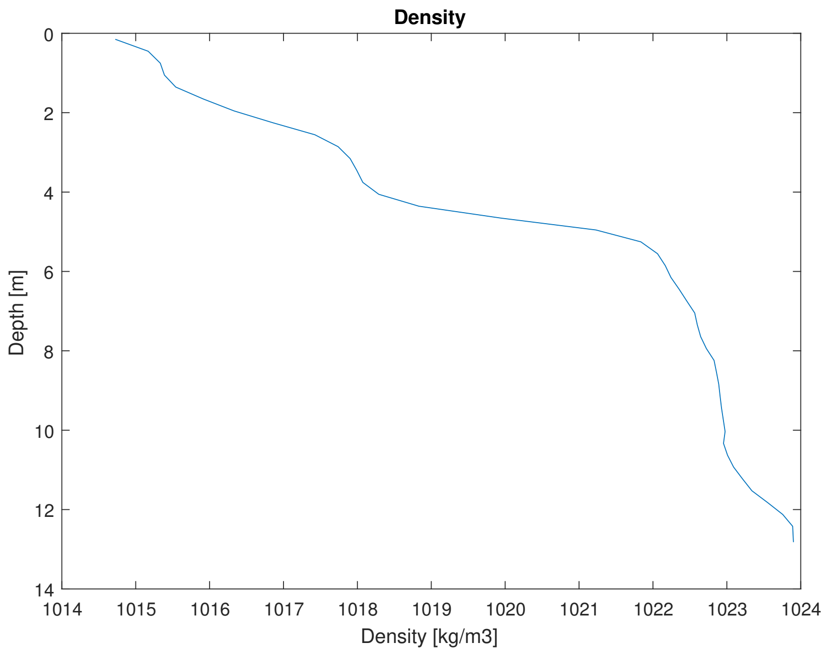


Figure 4.4: Density profile at the location of Physical Experiment #1, using a CTD probe one hour before the experiment

4.4 Physical Experiment #2

4.4.1 Setup/Plan

The second physical experiment was conducted inside, in a test pool at the facilities of SINTEF Marintek. The pool was 5 m deep, containing fresh water. A window at the bottom of the pool enabled to get a closer look of the vertical movements of the BV, see figure 4.5.

The BV is set to perform a similar program to physical experiment #1, however the first setpoint is at 4 m depth and the second setpoint is set to 1 m. This program was tested in two different segments, segment 1 with a decreased integration threshold of 1.5 m on the PID controller, and segment 2 with a regular integration threshold of 2.0 m. The total volume was also adjusted during segment 2, based on observation on segment 1.

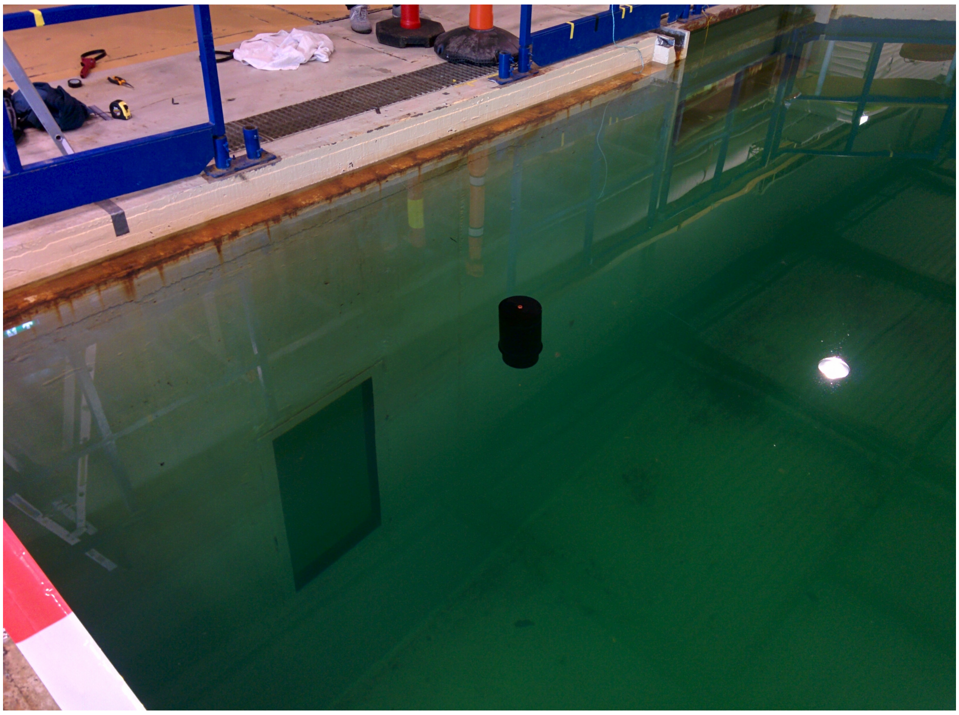
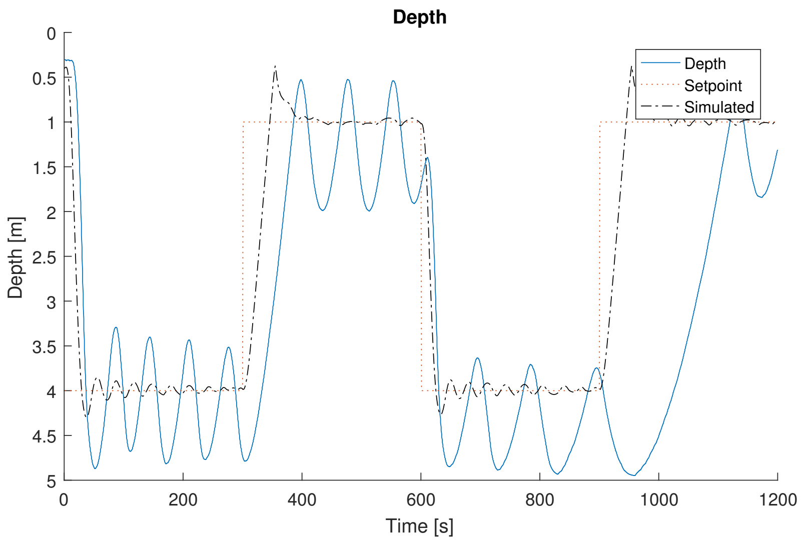


Figure 4.5: BV at physical experiment #2, in a test pool inside SINTEF Marintek facilities.

4.4.2 Experimental Results

Segment 1 of the experimental result can be seen in figure 4.6 and segment 2 is in figure 4.7.



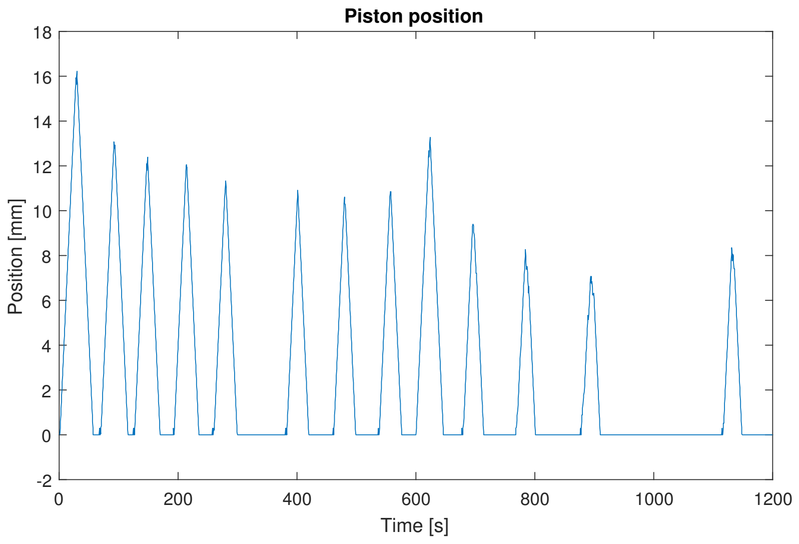
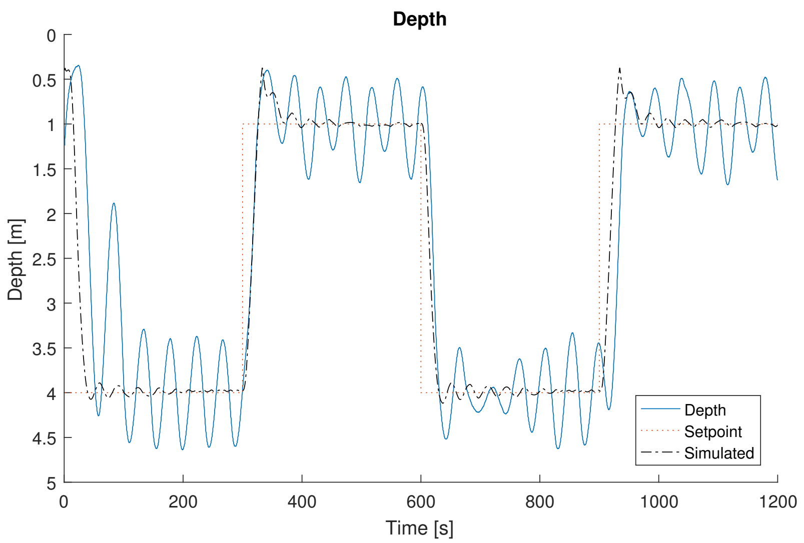


Figure 4.6: Physical Experiment #2 results, segment 1. Top graph shows the logged depth of the BV in the solid blue line, setpoint in dottet red line, and the simulated response in dash-dot black line. Bottom graph shows the piston position. Both time axis are started simultaneously.



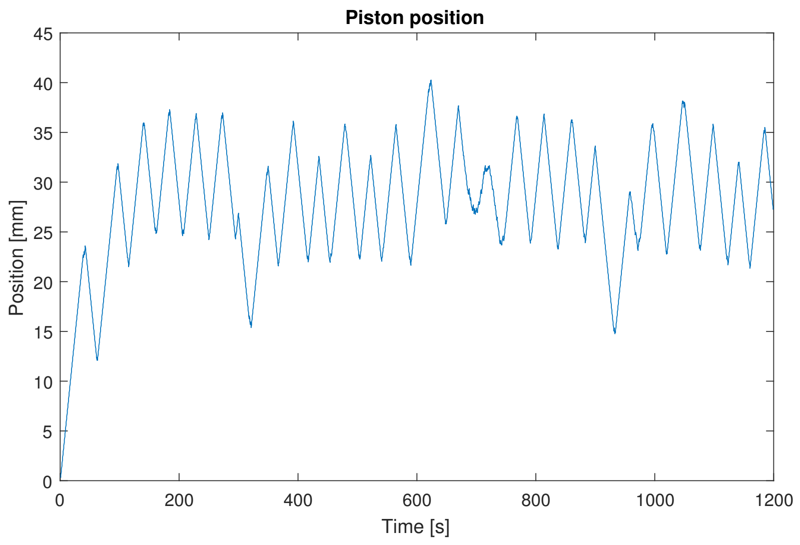
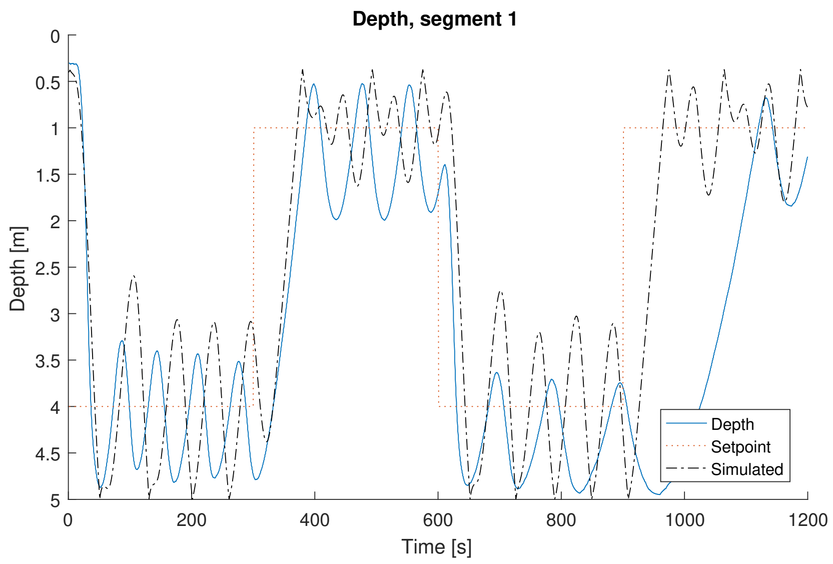


Figure 4.7: Physical Experiment #2 results, segment 2. Top graph shows the logged depth of the BV in the solid blue line, setpoint in dottet red line, and the simulated response in dash-dot black line. Bottom graph shows the piston position. Both time axis are started simultaneously.

As seen in the figures, the BV remained in standing oscillations around both setpoints, and did not behave as expected from the simulations. In segment 2, there was an attempt to manually dampen the oscillations during t=700-800s, by pulling the fish line. But the BV fell back into its standing oscillations as it was previously doing.

It was unclear during the experiment what caused this, but post analysis of the piston position from the experiment showed the piston velocity was set to its default speed, at $0.59\frac{mm}{s}$(24.2 $\%$ of the intended value), for an unknown reason. By using this value on the piston velocity on the simulations, one get a closer approximation to the physical experiment, see figure 4.8.



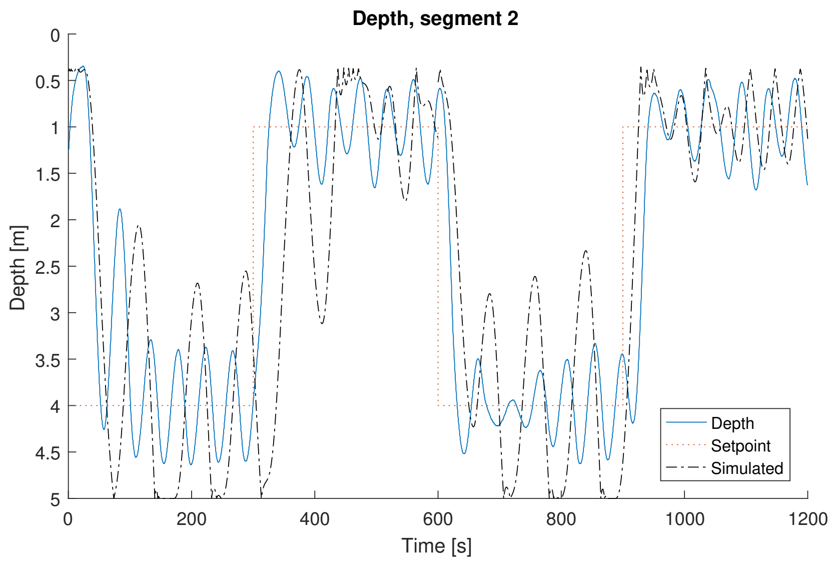


Figure 4.8: Physical Experiment #2 results, updated piston velocity on simulation

4.5 Battery Consumption

The raw battery voltage during physical experiment #1 can be seen in figure 4.9, and physical experiment #2 in figure 4.10. Because of the large amount of disturbance on the signal, a moving average filter was implemented post experiment in Matlab. A control check of the battery voltage was taken before and after each physical experiment with a precise and accurate multimeter with the total time the BV was powered during the experiments, given in table 4.4.

Experiment #	U_{start} [V]	U_{end} [V]	Duration [s]
1	24.20	23.58	4500
2	23.63	23.30	3780

Table 4.4: Battery Voltage before and after the two physical experiments

If the battery voltage is assumed to discharge linearly, using these measurements will give a total operation time of 22.6 hours from experiment #1, and 35.6 hours from experiment #2.

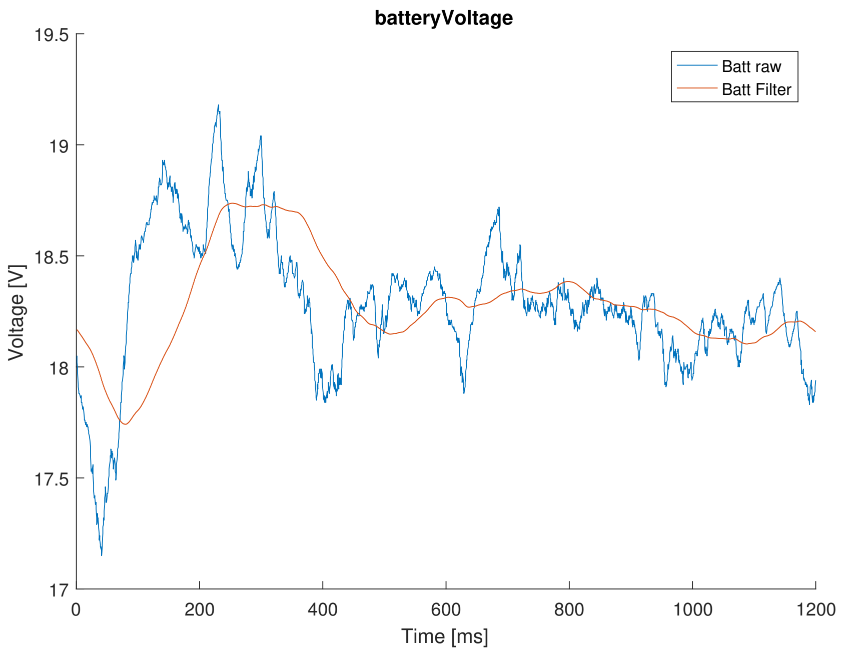


Figure 4.9: Battery voltage during Physical Experiment #1. Blue line is the raw battery voltage, and red line is a moving average filter.

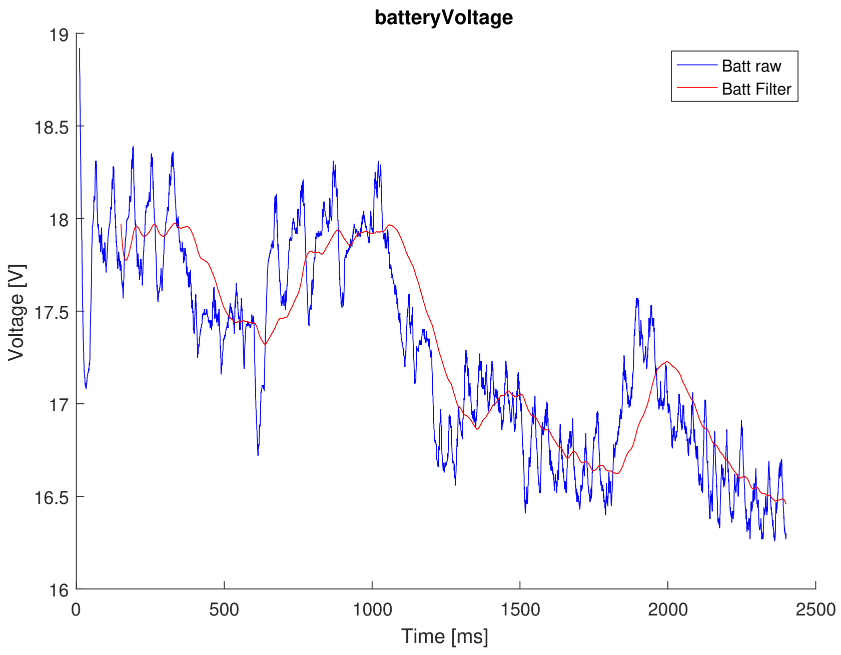


Figure 4.10: Battery voltage during both segments in Physical Experiment #2. Blue line is the raw battery voltage, and red line is a moving average filter.

Chapter 5

Discussion

5.1 Prototype Properties

5.1.1 Mass/Volume Properties

As mentioned in the results, the mass was underestimated from the physical prototype. There are several reasons this happened, and much of it boils down to the timeline of the construction. Initially, it was planned to construct the entire linear actuator first to get an exact measurement of its weight, and then adjust the volume of the BV before it was constructed. But because of a delay on the delivery of the stepper motor and shortage of time, it was necessary to start the construction of the BV prematurely with mostly calculated estimates on the mass.

A mass offset was anticipated, where the outer lid could solve this. On the one hand, the mass offset resulted in a decreased maximum attached payload-mass, while on the other it enables attaching floating payloads to the BV as well.

There were no physical experiments testing an attached payload, and additional dynamics associated with this were not implemented in the simulations. The attached payload therefore still stands as a theoretical option that requires physical experiments to verify it as ”plausible”.

5.1.2 Linear Actuator

The linear actuator was satisfying for this project. The motor driver was easily implemented, with convenient built in functions.

Actuator Force

Linear force delivered to the piston was theoretically adequate, but was not tested through physical experiments. As there was several assumptions with the linear force, like motor efficiency remains the same when changing input power, and torque out on the rotating motor shaft remains the same as the torque on the lead screw, there is most likely a difference between the theoretical calculation and the physical value. But in this project the accuracy is not as important, as the actuator force is only required to be larger than a certain value. Therefore, it is possible to estimate a value significantly larger than the minimum requirement to operate normally in the applicable area.

Limitations

One of the limitations of the implemented linear actuator was lacking a method for initializing its position after power was cut during operation. If the power was cut, and the piston position was not in a 0 position, the piston would try to move beyond the housing cap, which could possibly damage or destroy the BV. This problem could be solved in software, by continually storing its current position under operation. When initializing, the last stored value would be set to the current piston position. An electromechanical solution detecting if the piston tries to move out its physical limits could also be implemented as a redundant method of avoiding damages.

Another limitation to consider is the price of the stepper motor. Though convenient for a prototype, the stepper motor was by far the most expensive part of the electronics. Most of the cost in the stepper motor comes from its integrated motor driver, with its implemented software. Implementing a separate stepper motor with a cheaper motor driver could be a consideration.

5.2 Hardware Design

The hardware design was designed as an easily implementation for a prototype purpose. This gave room for little human error. One example of this is there was no short circuit protection for the microcontroller and motor driver circuit. Unfortiunatly an accident occurred during the project where the battery voltage short circuit through the implemented PCB, destroying both the microcontroller and the motor driver circuit. Both was replaced. Using a DC power connector on the wires from the batteries would reduce the risk of accidentally short circuiting, compared with using pins. Putting physical covers, separating battery wires from the rest of electrical parts could also be a solution. But this require more time to implement, which might be better used elsewhere.

5.3 Physical Experiment #1

5.3.1 Location

Although the location was not suited as a test environment (especially on the test day), it was providing a realistic test with larger density variation caused by the ongoing spring flow. In hindsight, the experiment should have been executed by attaching the BV to an anchored guide line, preventing horizontal movement. Releasing the BV from a free drifting boat is also an alternative possibility.

5.3.2 Result

During the experiment there was much uncertainties where the BV was located, both horizontally and vertically, or if it was tangled and stuck to a mooring. Expectations were small for achieving meaningful results, and not losing the BV was considered a success at one point. Even though, the BV did behave very similar to the simulations, considering the unideal conditions. It did not properly regulate to its setpoints. This is most likely caused by the tuning of the PID controller. First, the integral threshold were probably set too small (2 m threshold was used). Second, the gain on the integral could be slightly increased. This is seen in figure 4.3, t=400-600 s, the position moves towards the reference point, but very slowly. In the simulations the integral gain seems satisfactory, which could indicate the BV was influenced externally.

The presented simulations in figure 4.3 were done after the physical experiment with the density profile taken at the experiment. Therefore, it was not expected the BV would not stabilize on the setpoints.

Piston Position

The results were performed with the default piston speed of 0.59 mm/s (as intended) and tuned with a different set of PID controller parameters (Kp=0.006,Ki=0.00015,Kd=0.05). Increased piston speed (2.44 mm/s) was implemented after the experiment, which significantly improved the robustness of the PID controller in the simulations, in regard to varying density profiles and setpoints at arbitrary depths. This was intended to test in physical experiment #2.

5.3.3 Density

The density approximation was surprisingly close to the measured density, considering these uncertainties:

- CTD profile was taken approximately 1 hour before the physical experiment. There is a good reason to assume that the CTD profile was changed because of much winds and currents.

- Difference between calculated and actual used volume of the BV. A simple ruler was used for placement of the outer lid, which provides <1 mm accuracy. The volume from various screw ”bumps” and outside tape was not accounted for in volume calculations. A proper measurement/calibration of the exact volume could be conducted to get increased accuracy.

- Horizontal drifting of the BV, CTD measurement was not necessarily at the same horizontal location.

5.4 Physical Experiment #2

5.4.1 Location

Much more suited location for experimental testing, but does not have a large density variation, an unrealistic condition for most instances in oceans/sea waters.

5.4.2 Result

The results in physical experiment #2 does unfortunately not provide with much useful information of the expected properties of the BV, because of the error with the piston speed. What caused this error is still unknown, as the system was tested ”on land” both before and after the experiment, showing no signs of the same error occurring.

Simulations

Seeing the simulations have a similar response when updating the piston velocity is a promising sign for the validity of the simulations. A ”bouncing” dynamic was needed to implement in vehicle dynamics simulation, as the BV collided with the bottom floor of the pool several times, causing a more complex and more difficult system to simulate. Hitting different bottom materials will also alternate the dynamic behavior, and is best to make efforts to avoid bottom collision at all.

5.5 Battery Consumption

5.5.1 Battery Measurement Offset

First off, the battery voltage over time in figures 4.9 and 4.10 have clearly an offset from the measured battery voltage before and after the experiment. There are a couple reasons this might occur. One is the resistor tolerance, given at $\pm 5\%$ for each resistor. In the worst-case scenario, the battery voltage could be up to 12.05 ($\frac{10.5M\Omega + 0.95M\Omega}{0.95M\Omega}$) times the analog signal, while the analog signal was multiplied with 11.0 to estimate the battery voltage. But this is not substantial enough to cover the entire offset alone.

Another cause could be the analog reference level on the ADC. When reading from the ADC, it is assumed to get an integer between 0-1023 that represent 0.0-5.0 V. But often the reference voltage is not 5.0 V exactly. Say, for instance, if the reference voltage was at 5.2 V, and the ADC got an analog signal of 2.5 V. When read, this voltage would be represented as the integer 492 ($\frac{2.5*1023}{5.2}$). Assuming 5.0 V reference level, this integer would be translated in the microcontroller to 2.4 V ($\frac{492*5.0}{1023}$), approximately 4% difference. Arduino does have a built-in solution to this problem, using the analog reference pin (AREF) to connect an accurate reference voltage.

5.5.2 Noise

A large issue with the measured signal is the noise. Though a digital filter was implemented, this was clearly not sufficient. A physical low-pass RC filter is an easy implemented solution that should have been implemented to reduce the noise of the signal.

5.5.3 Result

One of the intended use of measuring battery voltage during operation was to get indications of power consumption from attempting various low-power modes and algorithms. But it is not possible to interpret much about the battery decay form figures 4.9 and 4.10, because of large variations and noise.

The total operation time of the batteries is far too small a sample size to conclude anything with. Considering the physical experiment #2 was more or less continuously with the stepper motor activated, an approximation of 35.6 hour operational time is a positive sign for future improvements. It is reasonable to believe that, by using power saving algorithms and using low-power microcontrollers, the power consumption could be reduced to at least 20% of what the experiment used, increasing the operational time by minimum five times.

Chapter 6

Conclution

A functional prototype, using buoyancy as a principle for vertical underwater movement, was successfully constructed, and two physical experiments were carried out with the unit. The system requirements was implemented with theoretically sufficiency, but is yet verified through additional physical experiments.

There is not enough experimental data on the unit to exacly determine the energy efficiency of using buoyancy as a principle of vertical movement underwater. But study of similar prototypes and theoretical power optimization solutions suggest there is a promising potential for the unit being energy efficient.

Among the physical difficulties includes constructing the correct vehicle density to reamin neutrally buoyant. One example of this was obtaining an accurate estimate of the implemented mass. This was partly caused by a lack of time, and sequencing the implementation of the different parts could have been structured better. Designing the maximum volume change large enough to cover a wide range of water densities also proved challenging. On top of this was designing a robust physical structure capable of withstanding high pressures underwater.

Some of the technical difficulties was creating a stable and accurate reading of the battery voltage. Other difficulties includes unexpected firmware failures, creating and tuning a control system, and implementing an accurate simulation of the dynamic behaviors.

6.1 Future Work

The implementation of the electrical design, specifically the electronics implemented on the PCB, functioned well as a prototype design, but with much potential for improvement. An integrated electric circuit with wireless communication would simplify performing experimental tests.

More physical experiments to document energy efficiency and physical properties remains. Designing a mechanism to ensure the prototype remains stationary in horizontal directions can be beneficial.

Bibliography

Richard L Saunders. Adjustment of buoyancy in young atlantic salmon and brook trout by changes in swimbladder volume. *Journal of the Fisheries Board of Canada*, 22: 335–352, 1965.

Gayathri K. Devi, B.P. Ganasri, and G.S. Dwarakish. Applications of remote sensing in satellite oceanography: A review. *Aquatic Procedia*, 4:579–584, 2015.

D. Roemmich, G.C. Johnson, S. Riser, R. Davis, J. Gilson, W.B. Owens, S.L. Garzoli, C. Schmid, and M. Ignaszewski. The argo program, observing the global ocean with profiling floats. *Oceanography*, 22:34–43, 2009.

B.L. Richard, F.J. William, and G.M. Roland. Buoyancy control, 1961. URL `https://www.google.com/patents/US2968053`.

D.C. Sparks, L. Belfie, D. Bruder, C.T. Werner, and J.W. Widenhofer. Buoyancy control system, 1995. URL `https://www.google.com/patents/US5379267`.

R.E. Davis, L.A. Regier, J. Dufour, and D.C. Webb. The autonomous lagrangian circulation explorer (alace). *Journal of atmospheric and oceanic technology*, 9:264–285, 1992.

M. Clarke. Buoyancy control as a function of the spermaceti organ in the sperm whale. *Journal of the Marine Biological Association of the United Kingdom*, 58:27–71, 1978.

H. Yamamoto and K. Shibuya. New small buoyancy control device with silicone rubber for underwater vehicles. *Advances in Cooperative Robotics*, pages 258–265, 2016.

Institution of Oceanography Scripps. Solo-trec (sounding oceanographic lagrangrian observer thermal recharging) configuration. URL `http://auvac.org/configurations/view/257`.

Jules S Jaffe, Peter JS Franks, Paul LD Roberts, Diba Mirza, Curt Schurgers, Ryan Kastner, and Adrien Boch. A swarm of autonomous miniature underwater robot drifters for exploring submesoscale ocean dynamics. *Nature communications*, 8, 2017.

Yunus A. Çengel and John M. Cimbala. *Fluid Mechanics Fundamentals and Applications*. McGraw-Hill, 2010.

Frank J Millero and Alain Poisson. International one-atmosphere equation of state of seawater. *Deep Sea Research Part A. Oceanographic Research Papers*, 28:625–629, 1981.

Olav Egeland and Jan Tommy Gravdahl. *Modeling and Simulation for Automatic Control, Chapter 3*. Marine Cybernetics AS, 2003.

Nanotec. Nanotec linear actuators. URL `https://en.nanotec.com/fileadmin/files/Tutorials/Linearactuators_Training.pdf`.

Trinamic. Pd-1161 frimware manual, 2014. URL `https://www.trinamic.com`.

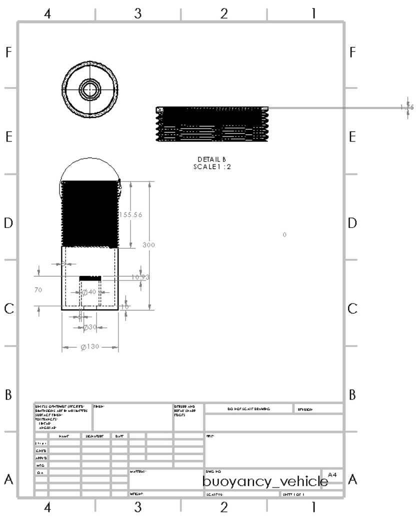
Trinamic. Pd-1161 hardware manual, 2013. URL `https://www.trinamic.com`.

Brett Beauregard. Arduino pid library. URL `http://playground.arduino.cc/Code/PIDLibrary`.

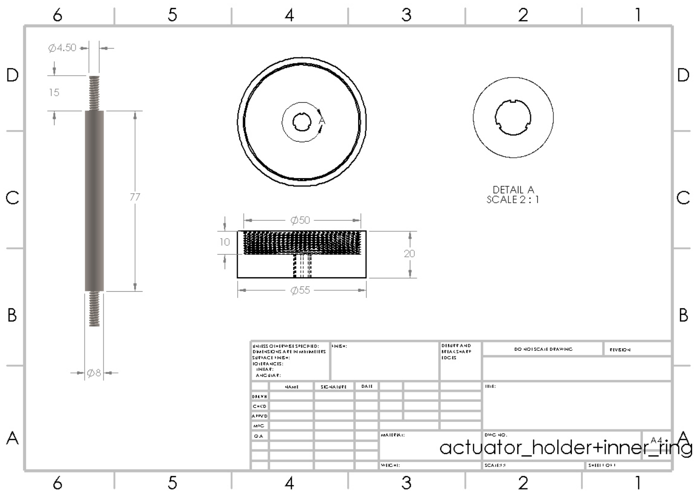
Meguno-LINK. Exponential filter. URL `http://www.megunolink.com/documentation/arduino-libraries/exponential-filter/`.

Appendix

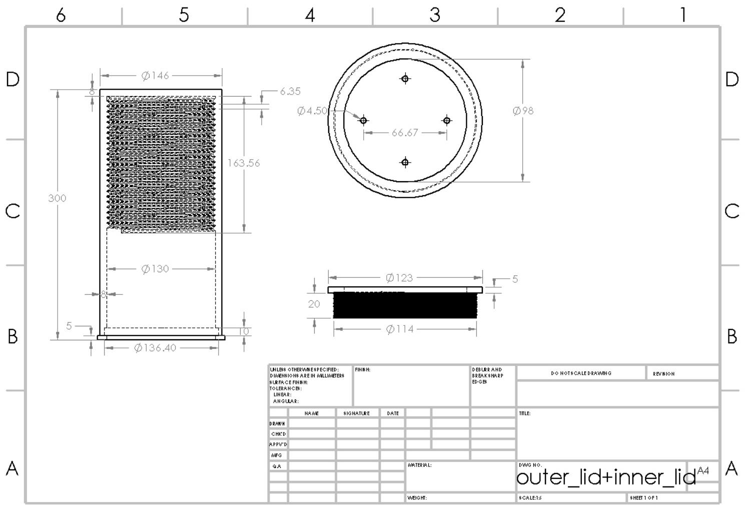
Appendix A - Solidworks Drawings



Vehicle House Drawing



Actuator Holder and Inner Ring Drawing



Outer lid and Inner Lid drawing

Appendix B - Matlab Simulation Code

```
clear all
clc

%%Reference parameters
D_ref1 = 5; % Reference depth 1
D_ref2 = 2; % Reference depth 2
t_change = 300; % Time for reference change
t_run = 600; %Total simulated run time

%Starting parameters
h0 = 0e-3;  %Starting position of piston (0-0.06)
x0 = 0.4;    % Starting depth

%%CTD profiles
sampleData = load('CTD_data/CC1437011_20170609_081403');
salinityProfile = sampleData.Salinity;
tempProfile = sampleData.Temperature;
depthProfile = sampleData.Depth;

%Fresh water example
% salinityProfile = [0 0];
% tempProfile = [17 16];
% depthProfile = [0 20];

g = 9.81;
rho_min = 1016;% Min BV Density

%%Vehicle parameters:
Cd = 0.9; % Drag coefficient

%Thickness pvc
t_pvc = 0.008;
t_pvc_piston = 0.005;
t_pvc_lid = 0.008;

h_cyl = 0.06; %Moveable height of piston
h_st = 0.01;     %Height piston block
h_base = 0.305;    %Height vehicle
h_lid = 0.30;    %Height outer lid

r_ve = 0.065;    %Radius Vehicle
r_piston = 0.02; % Cylinder radius
r_lid = r_ve+t_pvc_lid;
```

```matlab

F_piston_max = 1022.3; %Maxium linear actuation force

%%Calculations
%Volume/Area:
A_vehicle = pi*r_lid^2; % Projected area, vertical direciton
A_pis = pi*r_piston^2; % Piston area

V_lid = (h_lid*pi*r_lid^2)+((pi*0.005*(r_lid+0.005)^2)-(pi*0.005*r_lid^2));
V_bot = ((h_base+t_pvc_lid-h_lid)*pi*r_ve^2)-(pi*0.01*(0.015^2));
V_base = V_lid + V_bot;
deltaV = pi*h_cyl*r_piston^2; % Piston volume

%Mass/density:
m = 6.101; %Measured

V_max = m/rho_min;
h_additional = (V_max-V_base)/(pi*r_ve^2);
h_add = h_additional+0.013;
rotations = (0.15-h_add)/0.006;
h_ve = h_base + h_additional;
V_min = V_max-deltaV;

rho_max = m/V_min;
rho_interval = rho_max - rho_min;

%%Actuation parameters:


v_piston_max = 2.44e-3; %Movement Speed of Piston[m/s]
deadband_speed = 0.3e-3; % Minimum actuation movement [m]

depth_max = (F_piston_max/A_pis)/(rho_max*9.81); %Maximum depth from gauge pressure on piston
depth_terrain = 100.0; %[m]

pitch_bolt = 1e-3; % Space between bolt threads
delta_pos_piston_min = (1/(51200*pitch_bolt)); % Linear movement per step

%%Pressure sensor parameters
```

```
delta_res_pressure_sensor = 6.89/(2^10); %Resolution of
    analog pressure sensor (pressure range(bar)/(2^ADC-bits)
    )

%%Control parameters
Offset = 0; %Helps overshoting referance
sampleTime = 0.5;
%Ideal PID (with v_piston_max = 2.44e-3 and offset 0):
Kp = 0.02;
Ki = 0.001;
Kd = 0.06;
% Kp = 0.006;
% Ki = 0.00015;
% Kd = 0.05;

N = 1;

%Simulation
tspan = [0 t_run]; % Time span for simulation
options = simset('MaxStep', 0.5,'MinStep',1e-11, 'AbsTol',
    1e-11, 'RelTol', 1e-11);
set_param('buoyancy1','AlgebraicLoopSolver','LineSearch');
sim('buoyancy1.mdl',tspan,options);

%%Plotting
close all;
%Vehicle depth + ref
fig1 = figure(1);
hold on;
plot(Depth.Time, Depth.Data);
plot([0 t_change],[D_ref1 D_ref1],':');
plot([t_change t_run],[D_ref2 D_ref2],':');
title('Depth');
xlabel('Time [s]');
ylabel('Depth [m]');
set(gca,'YDir','reverse');
hold off;
fig1.Position = [20 520 1000 600];

%Piston Position
fig2 = figure(2);
hold on;
plot(pos_piston.Time,pos_piston.Data);
xlabel('Time [s]');
ylabel('Position [mm]');
```

```
title('Piston position');
fig2.Position = [20 50 700 500];
hold off;

%Piston Velocity
fig3 = figure(3);
hold on;
plot(v_piston.Time,v_piston.Data);
plot([0 t_run],[deadband_speed*10^3 deadband_speed*10^3],':
    ');
plot([0 t_run],[-deadband_speed*10^3 -deadband_speed*10^3],
    ':');
title('Piston Velocity');
xlabel('Time [s]');
ylabel('Velocity [mm/s]');
fig3.Position = [800 620 700 500];
hold off;

power_est_threshold = [];
for i = 1:length(v_piston.Data)
    if abs(v_piston.Data(i))>deadband_speed
        power_est_threshold(i) = v_piston_max;
    else
        power_est_threshold(i) = 0;
    end
end

power_est = trapz(v_piston.Time,abs(v_piston.Data));
power_est_t = trapz(v_piston.Time,power_est_threshold);
power_est_threshold_percent = 100*power_est_t/(t_run*
    v_piston_max); %Percent power use of maximum possible,
    when assuming equal power for different velocity
power_est_percent = power_est/(10*t_run*v_piston_max); %
    Percent power use of maximum possible

%Density
density = [];

for i = 1:800
    density(i)=Calculate_density(salinityProfile,
        tempProfile,depthProfile,i/10);
end
fig8 = figure(8);
plot(density,0.1:0.1:80);
title('Density');
```

```
xlabel('Density [kg/m3]');
ylabel('Depth [m]');
set(gca,'YDir','Reverse')
fig8.Position = [800 50 700 500];
```

Appendix C - Microcontroller Code

```
#include <SPI.h>
#include <SD.h>
#include <PID_v1.h>
#include <Filter.h>

File myFile;

/*DESCRIPTION*/
/*
 * This program is used to regulate and control the BV
   between two reference depths.
 * It will alternate between the two ref_depth every
   setPointTime until runTime is reached, or battery is
   low
 *
 */

/*TIMERS*/
unsigned long t_last1, t_last3, t_change1, t_change3, t_now
    ;
unsigned long sampleTime = 500; //Sample time of PID in ms
unsigned long startTime; //How much time before the program
    starts (ms)
unsigned long runTime,runTime2; //How long the program is
    running (ms)
unsigned long setPointTime; //Time between alternating the
    setPoint (ms)

/*BATTERY*/
unsigned int batteryLow = 0; //1 if battery is considered
    low voltage (almost empty battery)
double batteryVoltage;
int batteryStrike = 0;

/*PID Controller*/
double ref_depth_1 = 4.0; //reference depth 1 (m)
double ref_depth_2 = 1.0; //reference depth 2 (m)
double depth, idealPositionPiston, setPoint;
double Kp=0.02, Ki=0.001, Kd = 0.06, Offset = 0.0,
    threshold_ki = 1.5;
PID myPID(&depth, &idealPositionPiston, &setPoint,Offset,
    Kp, Ki, Kd, DIRECT);

```

```
/*MISC*/
const float pi = 3.14159;
const float g = 9.81;
const float m = 6.101; //Mass [kg]
double density = 996;
const float V_min = 0.0051244; //Volume min (positionPiston
    = 0) [m3]
const float r_piston = 0.02; //Radius piston [m]
const int analogPressurePin = A4;
const int analogBatteryVoltagePin = A7;
double rail;        //Rail value on pressure sensor, for
    error handling,
                    //rail <2.5 => EEPROM Corrupt or Low
                        Supply voltage
                    //rail >97.5 => Sensor Bridge Open, Sensor
                        Bridge Short or Loss of ground
                        connection
double pistonPosition = 0;
/*FILTERS*/
ExponentialFilter<double> ADCFilter(7, 0.3);     //Filter
    for pressure sensor
ExponentialFilter<double> BatteryFilter(5, 18);//Filter for
    battery voltage



void setup() {
  /*SERIAL INIT*/
  Serial.begin(9600);
  //pinMode(13, OUTPUT);
  //digitalWrite(13, LOW);

  /*
  while (!Serial) {
     ;// wait for serial port to connect. Needed for native
         USB port only
  }
  */
  /*RUN TIMERS INIT*/
  startTime = 10.0*(60.0*1000.0);
  runTime = 30.0*(60.0*1000.0);
  setPointTime = 5.0*(60.0*1000.0);

  /*INIT*/
  SDinit();
  PIDinit();
  MOTORinit();

  /*Wait time for assembly*/
  while(millis()<startTime){
    batteryVoltage = readBatteryVoltage();
    delay(1000);
  }

  t_now = millis();
  t_last1 = millis();
  t_last3 = millis();
  //enableInterrupt(51200);
}

void loop() {
  while(t_now<runTime && !batteryLow){
    /*UPDATE TIMERS*/
    update_time();
    update_setPoint();
    if(t_change1 >= sampleTime){
      pistonPosition = getPistonPosition();

      depth = readDepth();
      delay(20);

      batteryVoltage = readBatteryVoltage(); //Reads
          battery voltage (nom 24-12.8 V)

      computePID();             //changes idealPositionPiston

      setPistonPosition();      //sets actuator to
          idealPositionPiston

      //batteryCheck();
      SDwrite();
      t_last1 = t_now;
    }
  }
  //Serial.println("End of while");
  delay(50);
  SendCmd(1,4,0,0,0);    //Set to float to surface
  delay(setPointTime);
}

```

```
/*TIMER UPDATES*/

void update_time(void){
    t_now = millis();
    t_change1 = (t_now - t_last1);
    t_change3 = (t_now - t_last3);
}
void update_setPoint(void){
  if(t_change3 >= setPointTime){
    if(setPoint == ref_depth_1){
      setPoint = ref_depth_2;
    }else{
      setPoint = ref_depth_1;
    }
    t_last3 = t_now;
  }
}


/*BATTERY*/
double readBatteryVoltage(void){
    int battVoltageRead = analogRead(
        analogBatteryVoltagePin);
    double battVoltage = battVoltageRead*(5.0/1023.0)*11.0;
         //Value from voltage divider
    BatteryFilter.Filter(battVoltage);
    return BatteryFilter.Current();
}
void batteryCheck(void){
    if(batteryVoltage <14.0){    //Minimum requirement for
        voltage regulator
      batteryStrike++;
      if(batteryStrike > 5){  //Make sure consistent
          readings under minimum value
          batteryLow = 1;
      }
    }else{
      batteryStrike = 0;
    }
}

/*PRESSURE SENSOR*/
//Reads pressure sensor from ADC, converts signal to depth.
    Values based on Honeywell PX3AN1BH100PSAAX datasheet.
double readDepth(void){
  long pressureValue = analogRead(analogPressurePin);
  rail = 100*pressureValue/1024.0;
  double meter = (((pressureValue -102.4)/118.8151266)
      *100000.0)/(g*density)+0.509;
  //102.4 = 10% of 1024, eg lowest output value
  //118.8151266 = ramp number for ~1 bar->~7 bar from
      102.4->921.6
  //Bar is then converted to m depth
  //+0.509 [m] = slight offset tuning, based on observation

  ADCFilter.Filter(meter);
  return ADCFilter.Current();
}

/*SD Storage*/
void SDinit(void){
  if (!SD.begin(10)) {
    Serial.println("SD init failed");
  }
  else{
    Serial.println("SD init success");
  }
}
//Writes to SD card (depth, density, time,
void SDwrite(void){
  myFile = SD.open("test7.txt",FILE_WRITE);
  if (myFile){
    myFile.print(millis());
    myFile.print(" ");
    myFile.print(depth,5);
    myFile.print(" ");
    myFile.print(setPoint);
    myFile.print(" ");
    myFile.print(idealPositionPiston,5);
    myFile.print(" ");
    myFile.print(pistonPosition,5);
    myFile.print(" ");
    myFile.print(batteryVoltage);
    myFile.println(" ");
    myFile.close();
    //Serial.println("Written to SD.");
  } else {
    // if the file didn't open, print an error:
    //Serial.println("error opening test.txt");
  }
}


/*PID Controller*/
void PIDinit(){
  setPoint = ref_depth_1;
  myPID.SetControllerDirection(0);
  myPID.SetOutputLimits(0,0.055);
  myPID.SetSampleTime(sampleTime);
  myPID.SetMode(AUTOMATIC);
  delay(100);
}
void computePID(){
  double threshold = abs(setPoint-depth); //absolute
      distance away from setpoint
  if (threshold <= threshold_ki)   //Use Ki parameter when
      close to setpoint.
  {
      myPID.SetTunings(Kp, Ki, Kd);
  }
  else
  {
    myPID.SetTunings(Kp, 0, Kd);
  }
  myPID.Compute();
}

/*STEPPER MOTOR*/
void MOTORinit(){
  SendCmd(1,5,6,0,66); //set max current on PD-1161
  delay(100);
  SendCmd(1,5,7,0,0); //set Standby current (Value = 0-255
      [*4/255 A])
  delay(100);
  SendCmd(1,5,214,0,5); //set Power down delay to standby
      current (Value = 1-65535 [*10msec])
  delay(100);
  SendCmd(1,5,154,0,1); //set Pulse Divisor (changes
      rotation speed) CAREFUL CHANGING, MUST NOT EXCEED
      PHYSICAL VALUES
  delay(100);
}
//sets position of piston to idealPositionPiston in m
    (~0-0.055)
//1 mm linear movement = 51200 (256*200) MVP
```

```
//estimated MVP range: −2816000.0 (51200∗−55) (5.5 cm) to 0 (0.0 cm)
void setPistonPosition(void){
  long value = −(idealPositionPiston∗51200000.0);
  SendCmd(1,4,0,0,value); //MVP command, move to absolute value
}

double getPistonPosition(void){
  unsigned char∗ RxAddress, RxStatus;
  long RxValue;
  SendCmd(1,6,1,0,0);

  if(GetResult(RxAddress,RxStatus,&RxValue)){
    double value = −(RxValue/51200000.0);
    //Serial.println(value);
    return value;
  }
  else{
    return 1; //Impossible for piston to be at 1 m, used for error handling instead
  }
}
double getEncoderPosition(void){
  unsigned char∗ RxAddress, RxStatus;
  long∗ RxValue;
  SendCmd(1,6,209,0,0);
  if(GetResult(RxAddress,RxStatus,RxValue)){
    double value = (∗RxValue/51200000.0)+0.06;
    //Serial.println(value);
    return value;
  }
  else{
    return 1; //Impossible for piston to be at 1 m, used for error handling instead
  }
}

//Send a command to stepper motor (PD−1161), see PD−1161 firmware manual for more instuctions
void SendCmd(unsigned char Address, unsigned char Command, unsigned char Type, unsigned char Motor, long Value)
{
  unsigned char TxBuffer[9];
  unsigned char i;

  TxBuffer[0]=Address;
  TxBuffer[1]=Command;
  TxBuffer[2]=Type;
  TxBuffer[3]=Motor;
  TxBuffer[4]=Value >> 24;
  TxBuffer[5]=Value >> 16;
  TxBuffer[6]=Value >> 8;
  TxBuffer[7]=Value & 0xff;
  TxBuffer[8]=0;
  for(i=0; i<8; i++)
    TxBuffer[8]+=TxBuffer[i];

  Serial.write(TxBuffer,sizeof(TxBuffer));
}
//Recieve message from PD-1161, return 1 if checksum is
    okay, 0 otherwise
// The follwing values are returned:
//      *Address: Host address
//      *Status: Status of the module (100 means okay)
//      *Value: Value read back by the command

int GetResult(unsigned char *RxAddress, unsigned char *
    RxStatus, long *RxValue)
{
  unsigned char RxBuffer[9], Checksum;
  //DWORD Errors, BytesRead;
  //COMSTAT ComStat;
  int i = 0;
  int check = 0;
  //First, get 9 bytes from the module and store them in
      RxBuffer[0..8]
  //(this is MCU specific)
 /*
  while(i<9){
    if(Serial.available()>0){
      RxBuffer[i] = (char)Serial.read();
      i++;
    }
  }
  */
  while(!check){
    if(Serial.available()>0){
      Serial.readBytes(RxBuffer,9);
      i++;
    }
  }
  //Check the checksum
  Checksum=0;
  for(i=0; i<8; i++)
    Checksum+=RxBuffer[i];
  if(Checksum!=RxBuffer[8]) return 0;

  //Decode the datagram
  *RxAddress=RxBuffer[0];
  *RxStatus=RxBuffer[2];
  *RxValue=(RxBuffer[4] << 24) | (RxBuffer[5] << 16) | (
      RxBuffer[6] << 8) | RxBuffer[7];

  return 1;
}
void enableInterrupt(long postion){
  SendCmd(1,37,3,0,postion);   //Set interrupt vector (VECT)
  delay(10);
  SendCmd(1,25,3,0,0);   //Enable interrupt
  delay(10);
}
```