Automated Buoyancy Control System for the Remotely Operated Sea Crawler (ROSCo)

by

Arne Hendricks

Bachelor of Science
Computer Science
University of Erlangen
Erlangen, Germany
2012

A thesis submitted to
the Department of Marine and Environmental Systems
at Florida Institute of Technology
in partial fulfillment of the requirements for the degree of

Master of Science
In
Ocean Engineering

Melbourne, Florida
March, 2014
We, the undersigned committee, hereby recommends that the attached document be accepted as fulfilling in part the requirements for the degree of Master of Science in Ocean Engineering

Automated Lifting System for the Remotely Operated Sea Crawler (ROSCo)

by

Arne Hendricks

Stephen Wood, Ph.D., P.E.
Associate Professor
Ocean Engineering Program Chair
Marine and Environmental Systems
Major Advisor, Committee Chair

George A. Maul, Ph.D.
Professor and Department Chair
Marine and Environmental Systems
Committee Member

Munever Subasi, Ph.D.
Mathematical Sciences
Committee Member
Abstract

Automated Buoyancy Control System for the Remotely Operated Sea Crawler (ROSCo)

by

Arne Hendricks

Major Advisor: Stephen Wood, Ph.D., P.E.

The work in this thesis presents a system for automated lifting and buoyancy control for the Remotely Operated Sea Crawler (ROSCo). This system, consisting of hydraulic, electric and control subsystems, allows the operator to navigate the vehicle to any desired depth. The Automated Buoyancy Control System (ABCS) controls the amount of air within each bladder supplied by compressed air cylinders (i.e. scubatanks). To control the ascent/descent velocity, air valves are regulated by a control system consisting of sensors and microcontrollers.
"I've missed the breeze of my home shores
the frozen lakes and winter snow
but now my dreams start to unfold
father, I'm coming home"
Table of Contents

Abstract ...................................................................................................................................... v
Table of Contents .................................................................................................................... viii
List of Figures ........................................................................................................................ xii
List of Tables ........................................................................................................................ xv
List of Equations ................................................................................................................... xvi
List of Symbols .................................................................................................................... xvii
   Constants ........................................................................................................................... xvii
   Variables .......................................................................................................................... xvii
Acronyms ............................................................................................................................... xviii
Preface ....................................................................................................................................... xx

Part I Problem Introduction and Background ......................................................................... 1
Chapter 1 Introduction ........................................................................................................... 3
   1.1 Outline of the Thesis ................................................................................................. 3
   1.2 Motivation ................................................................................................................ 5
   1.3 State of the Art ........................................................................................................ 6
   1.4 Problem Statement ................................................................................................. 7
   1.5 Scope and objectives ............................................................................................... 7
Chapter 2 Project Background ............................................................................................... 9
   2.1 History .................................................................................................................... 9
   2.2 Classification of buoyancy devices ........................................................................... 16
Chapter 3 Fundamental Fluid Mechanical Basics ............................................................... 17
   3.1 Ideal Gas Law .......................................................................................................... 17
   3.2 Pressure in a resting fluid ........................................................................................ 17
   3.3 Buoyancy ................................................................................................................ 18
   3.4 Bernoulli’s Equation ............................................................................................... 20
      3.4.1 Free Jet Flow ................................................................................................... 20
   3.5 Drag ......................................................................................................................... 22
Part II Mathematical Modeling ............................................................................................... 23
Chapter 4 Mathematical modelling of an buoyant underwater ROV .................................... 25
   4.1 Governing equations for a buoyant ROV ............................................................... 25
List of Figures

Figure 1-1  Segment from digital logic of the ABCS for the ROSCo ............................... 6
Figure 1-2 Remotely Operated Sea Crawler (ROSCo) .................................................... 6
Figure 1-3 Schematic Flow Chart of ABCS ................................................................. 8
Figure 2-1 Buoyancy control filed March 14, 1946 (Patent# US 2968053 A) ..................... 9
Figure 2-2 Buoyancy control Filed March 14, 1946 (Patent# US 2968053 A) ................. 10
Figure 2-3 Buoyancy control systems and methods (Patent# US 20090178603 A1) ...... 11
Figure 2-4 Floatation structure for e.g. manned underwater craft, which has an actuator that is arranged for varying density of filling material partially filled in variable-volume chamber (Patent# DE 102011057091 A1 ) ................................................................. 12
Figure 2-5 Safe automatic buoyancy control device ................................................... 14
Figure 2-6 Buoyancy control system (US 2013/0327263 A1) ......................................... 15
Figure 3-1 Pressure at different points in resting fluid (Munson 2012) ........................... 18
Figure 3-2 Fluid pressure forces from below are larger than above thus resulting in positive vertical force (Munson 2012 ) .................................................................................. 19
Figure 3-3 Buoyant Force on submerged and floating bodies (Munson 2012) ................. 20
Figure 3-4 Vertical jet flow from a tank (Munson 2012) .............................................. 21
Figure 4-1 ROSCo with forces acting on it .................................................................. 26
Figure 4-2 ROSCo at time t ..................................................................................... 27
Figure 4-3 Deflation through bleeding valve at time t ................................................... 33
Figure 5-1 Exact solution(solid) and results with Euler method, smaller points means more steps(Endre 2003) ................................................................. 43
Figure 5-2 Depth VS Time – empty tanks over test period, negative buoyancy .......... 48
Figure 5-3 Force VS Time – empty tanks over test period, negative buoyancy .......... 49
Figure 5-4 Velocity VS Time – empty tanks over test period, negative buoyancy .... 50
Figure 5-5 Depth VS Time – Fixed amount air, positive buoyancy .............................. 50
Figure 5-6 Force VS Time – Fixed amount of air, positive buoyancy ......................... 51
Figure 5-7 Buoyant force VS Time – fixed amount of air, positive buoyancy ............... 52
Figure 5-8 Velocity force VS Time – fixed amount of air, positive buoyancy ................ 52
Figure 5-9 Velocity force VS Time – Varying amounts of air, fixed during test, positive buoyancy ........................................................................................................... 54
Figure 5-10 Force force VS Time – fixed amount of air, positive buoyancy ................. 54
Figure 5-11 Buoyant force force VS Time – fixed amount of air, positive buoyancy .... 55
Figure 5-12 Force VS Time – Varying frequencies ..................................................... 57
Figure 5-13 Frequency response Bode plot ............................................................... 58
Figure 5-14 Pole-Zero Map .................................................................................... 59
Figure 6-1 Simplified control system ( Nise 2010) ....................................................... 61
Figure 6-2 Open-Loop control system (Nise 2010) .................................................... 62
Figure 6-3 Closed loop system (Nise 2010) ............................................................... 62
Figure 7-1 PID controller with filtering of derivative gain (Simulink) ......................... 65
Figure 7-2 Simplified PID system .......................................................... 66
List of Tables

Table 4-1: Useful variables defined......................................................................................... 37
Table 5-1: Fixed parameters of the test .................................................................................. 46
Table 5-2: Parameters of the test ............................................................................................ 42
Table 5-3: Parameters of test #3 ............................................................................................ 49
Table 9-1: Fixed parameters of the test ................................................................................. 69
Table 10-1: Arduino with ATmega2560 (Arduino Mega Manual) ........................................ 106
List of Equations

3-1 Ideal gas law ....................................................................................................................... 17
3-2 Pressure at point n ............................................................................................................. 17
3-3 Buoyant force .................................................................................................................... 19
3-4 Buoyant force .................................................................................................................... 19
3-5 Buoyant force .................................................................................................................... 19
3-6 Bernoulli equation .......................................................................................................... 20
3-7 Bernoulli equation .......................................................................................................... 20
3-8 Free jet flow .................................................................................................................... 21
3-9 Drag .................................................................................................................................. 22
4-1 Summation of forces ....................................................................................................... 26
4-2 Differential equation of 2nd order ................................................................................ 28
4-3 v(t) ................................................................................................................................. 28
4-4 x(t) ................................................................................................................................. 28
4-5 V_{gas}(t) ........................................................................................................................ 28
4-6 p(t) ................................................................................................................................. 28
4-7 V_{gas}(t) developed ......................................................................................................... 28
4-8 F_{B}(t) ............................................................................................................................. 29
4-9 F_{D}(t) ............................................................................................................................. 29
4-10 x(t) ............................................................................................................................... 29
4-11 \int x(t) dt ..................................................................................................................... 30
4-12 \int \int x(t) dt .................................................................................................................. 30
4-13 Inflation/Deflation rate of ballast tanks ..................................................................... 31
4-14 Applied Bernoulli ........................................................................................................... 33
4-15 Airflow velocity into tank ............................................................................................ 34
4-16 Deflation rate of ballast tank ...................................................................................... 34
4-17 Deflation rate of ballast tank ...................................................................................... 35
4-18 Completed expression for airflow in/out of tanks ...................................................... 36
List of Symbols

Constants

\( \pi \) \quad 3.141 592 654
\( g \) \quad 9.81 m/s\(^2\)

Variables

\( C_D \) \quad Drag coefficient ............................................................... [-]
\( g \) \quad gravity .................................................................................. [m/s\(^2\)]
\( \rho \) \quad Density ................................................................................ [kg/m\(^3\)]
## Acronyms

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABSC</td>
<td>Automatic Boyancy Control System</td>
</tr>
<tr>
<td>ROSCo</td>
<td>Remotely Operated Sea Crawler</td>
</tr>
<tr>
<td>FIT</td>
<td>Florida Institute of Technology</td>
</tr>
<tr>
<td>PID</td>
<td>Proportional Integrative Derivative</td>
</tr>
<tr>
<td>FL</td>
<td>Fuzzy Logic</td>
</tr>
<tr>
<td>uC</td>
<td>Microntroller</td>
</tr>
<tr>
<td>LPM</td>
<td>Liter per Minute (Flow)</td>
</tr>
<tr>
<td>I/O</td>
<td>Input and Output</td>
</tr>
</tbody>
</table>
Preface

I am very thankful to my advisor, Dr. Stephen Wood, for all his support, advise and patience. I am also grateful to my thesis committee members, Dr. George Maul and Dr. Munever Subasi for their guidance and valuable suggestions.

I would also like to thank all the others who supported me writing my Masters thesis. Especially I want to thank Mat Jordan for being on hand with help and advice. I also want to thank my family, girlfriend and friends for reviewing my thesis discussing it and staying by me during the process of the thesis – may the Force be with you!

Arne Hendricks
March, 2014
Part I

Problem Introduction and Background
Chapter 1

Introduction

Mankind has always been interested in the oceans, which cover more than 70% of the earth (Nybakken and Webster 1998) and where more than 50% of all the people in the world live within 160 miles (Hinrichsen 1999).

The sheer size of the oceans and the technological issues of underwater exploration have prevented most of the oceans to be explored.

The Remotely Operated Sea Crawler is an attempt to give researchers, scientists and entrepreneurs the technical capability to explore more of the shallower regions. The Automated Buoyancy Control System for the ROSCo makes it more controllable and safer.

1.1 Outline of the Thesis

This thesis is divided into 11 chapters and an appendix.

Chapter 1 gives an introduction of the problem and the goals set for this thesis.

Chapter 2 presents an overview of important developments regarding this subject is given. This includes already existing control/embedded systems.

Chapter 3 presents fundamental Equations of important fluid mechanical principles.

Chapter 4 presents a mathematical model to describe the vehicle’s speed and position dependent on time as well as expression to analyze air in/out-take of the system.

Chapter 5 presents results of analyzing the physical system using an implementation of the mathematical model of Chapter 4 in Simulink.

Chapter 6, 7 and 8 analyzes different control algorithms in order to find a method to control the ascent and descent of the ROSCo.

Chapter 9 presents the results of tests on the implemented model with the different control methods presented.

Chapter 10 illustrates the electrical and mechanical design of the system.

Chapter 11 explains future work required for the ABCS to work more efficient, expand it and make it more robust. A conclusion of the work done in this document is given.
Appendices 1-4 illustrate various aspects of the system, including excerpts from the program code, schematics, and photos of the building process.
1.2 Motivation

The oceans contain a vast amount of sunken vessels. For example, in 1715, a storm sunk an entire fleet of eleven Spanish ships along with a huge amount of gold and treasure off the coast of Florida just north of Fort Pierce. In addition to the monetary value of these sunken ships vast amounts of scientific and historical knowledge can be obtained from the studies of sunken ships. The anthropological term for the study of underwater sites is Underwater Archaeology.

Underwater archaeology is more than researching sunken ships. It covers the exploration of once flourishing and now flooded “hotspots” of civilizations as well as study of the natural remains of mammals and other animals. (Muckelroy 1978). The fact that most of these sites are relatively untouched by human interaction is a consequence of one of the biggest challenges when exploring them: they are difficult to access and working on them is more dangerous than typical archaeological work. Trained divers have to be transported, supported and deployed from a surface drive vessel or platform, and diving at depths greater than 100 feet (30 m) becomes extremely hazardous and complicated. Saturation diving can be used and in special cases submarines, ROVs and remote sensing equipment are also used in underwater archaeological investigation. Logistical difficulties arise from constraints as divers and equipment have to be supported. A proper working platform, which can be either a ship or a platform, needs to provide air delivery, recompression chamber and medical facilities if saturation divers are used.

In order to search for and investigate underwater archaeological sites with the goal of retrieving and documenting lost artifacts and historic knowledge the Florida Institute of Technology recently launched a program under Stephen Wood Ph.D., P.E, to build and design a Remotely Operated Sea Crawler (ROSCo). In order to facilitate the operation of the ROSCo a new system is needed: an Automated Buoyancy Control System (ABCS). Figure 1-1 shows a segment from the digital control logic of the ABCS, while Figure 1-2 shows the ROSCo at the bottom of the sea.
1.3 State of the Art

Currently lifting heavy items from the bottom of the sea usually involves an air inflated lift bag, operated by a diver. As air in the lifting bags expands while the vehicle/object surfaces, an uncontrollable run-away ascent is inevitable if not carefully regulated. Thus a diver must be present and in close proximity to control the amount of air inside the lifting bags, to prevent runaway ascents. As the diver is in close proximity to the lift bag he is at risk of getting caught up in the run-away with fatal consequences. This worst case scenario is
1.4 Problem Statement

Unfortunately very real. As many diving operations are surface-supported, i.e. the air is provided to the diver by an umbilical support line, this line might get entangled with the vehicle.

1.4 Problem Statement

One of the primary capabilities of the Remotely Operated Sea Crawler (ROSCo) is to be able to “fly” from location to location in addition to “crawling”. To achieve the action of “flying” the crawler first needs to be able to adjust and control its buoyancy. To date there are no underwater crawlers that are capable of “flight”. It is the mission of this thesis to present an automated lift system that can be integrated into an underwater crawler system or any device / vehicle that requires active buoyancy control.

1.5 Scope and objectives

This thesis evaluates the proof of concept of an automatically controlled buoyancy system for the ROSCo crawler through simulation and the design and development of an prototype.

The following main objectives were set:

1. Design and development of the Automated Buoyancy Control System (ABCS) for deployment with the ROSCo. This includes:
   a. Mechanical system
   b. Instrumentation and control Hardware
   c. Autonomous Power Supply
   d. Control and data acquisition Software
   e. Human Machine Interface Software
2. Development of a mathematical framework for the ABCS
3. Simulator test and deployment of the ABCS in a controlled Test Environment with real-time data acquisition.

Two side objectives were set:

1. The system must be designed in a modular way, making it simple to expand and reproduce as well as easy to maintain.
2. The system must be made, at least initially, out of inexpensive off-the-shelf quality parts, yet providing a reasonable performance
Work flow of the ABCS system is as following: The control system controls the hardware, records instrumentation data, and sends data to valves that either inflate or deflate ballast tanks. The user is able to send commands remotely to the system. The ABCS is housed in a water resistant box attached to the ROSCo and its ballast tanks. Figure 1-3 shows a simplified schematic draw of the system, where the MCU = Microcontroller Unit and FC = Flow Control.

Figure 1-3 Schematic Flow Chart of ABCS

Compressed Air Tanks, as commonly used for recreational scuba diving, are connected to a valve controlled by a Master Controller. From the compressed air tank valve air hoses connect to separate units of the ABCS, where each houses an 8-Bit Microcontrollers and 12V-DC Solenoid Valves and an array of controlling and instrumentation hardware.
Chapter 2

Project Background

2.1 History

One of the first recorded patent for an automatic buoyancy control device was filed in Paris by Roland M Gogolick (US 2968053 A) in 1946 (see Figures 2-1 and 2-2).

The device was meant to guarantee a safe and quick descent of an underwater acoustic device to a certain depth where it would maintain its position. The underwater device to be lowered, an underwater noisemaker for example, would be attached to the device and then lowered to the desired depth.

Figure 2-1 Buoyancy control filed March 14, 1946 (Patent# US 2968053 A)
Further work was done by Joseph Imlach et al. in 2009 (US 20090178603 A1), who claimed to have invented a buoyancy device using a pump to change buoyancy (see Figure 2-3). It comprised a piston housing and member as well as a pump and a control and
working fluid. Operation of the pump then displaces working fluid within the working chamber to alter volume of the control chamber.

In 2011, Andreas Lonkai from Atlas GmbH filed another patent for a flotation structure for manned underwater vehicles that uses actuators (Patent # DE 102011057091 A1 see Figure 2-4). The device features a variable volume chamber, partially filled with a material such as petroleum or grease. The actuator then can vary the density of that fill material.
Figure 2-4 Floatation structure for e.g. manned underwater craft, which has an actuator that is arranged for varying density of filling material partially filled in variable-volume chamber (Patent# DE 102011057091 A1)
An invention focused on diver safety was presented in 2013 by David Bonzon (Patent # US US20130259579 A1 see Figure 2-5). The device presented supposedly may include features such as controlled ascent rates, imposition of a maximum depth limit and follow dive profiles for decompression. It basically consists of three or more venting valves that are supposed to even work when electrical power is shut down, ensuring manual functionality even in an emergency. An array of sensors measures diver velocity and speed.
A further patent was filed by Scuba Lab LLC, in 2013 (Patent# US20130327263 See Figure 2-6). It consists of a subsystem that allegedly dynamically controls lift move and hold an object of arbitrary size at a fixed depth by using a simulator based control algorithm. The inventors also claim to be able to pre-determine limits of instability of the
system. From the pictures the system seems to consist of several different units that
attach to the lifting bag and item.

Figure 2-6 Buoyancy control system (US 2013/0327263 A1)
2.2 Classification of buoyancy devices

The system uses electrically controlled solenoid valves and a control algorithm that relies on what is principally forecasting a change in the volume of the system, and thus reacting to that change by activating a deflation valve to lower the volume of air inside the bags.

2.2 Classification of buoyancy devices

As these presented patents and inventions show there is a wide range of concepts and applications that can be used for controlling buoyancy of either divers or ROVs. Consequently, the different devices can be divided into the following categories:

1. Target Location
   a. ROV: A device that is integrated in and designed to ensure controlled ascent of an ROV
   b. Underwater Craft: The device is integrated in a manned underwater craft and is supposed to control buoyancy.
   c. Attached to object: These devices are supposed to be attached to objects like mines etc to be able to safely surface them.
   d. All purpose: The device can be attached to any object, ROV etc that needs to incorporate the ability of controlled ascents and position holding

2. Buoyant Medium
   a. Gas: The device relies on the use of gas such as air to achieve positive buoyancy.
   b. Varying Material: The control device controls a material like petroleum or grease in order to change its filling density in a control volume.
   c. Piston-Pump: The device utilizes a pump to control a piston in a housing in order to alter volume of buoyancy.
   d. Thrust based: The system uses any means of propulsion of the craft etc to achieve positive buoyancy.
Chapter 3

Fundamental Fluid Mechanical Basics

Later sections of this thesis will focus on the design and modeling of the system. This chapter focuses on the physical concepts that are fundamental in understanding the mechanics behind a buoyancy control device.

3.1 Ideal Gas Law

In science it is a well known fact that gases are more compressible than liquids. While many fluid and hydrodynamic processes can be investigated by assuming incompressibility of the fluid, this is not possible for gases. The gas density changes in direct relation to a change in pressure, especially when assuming a constant temperature, as shown in equation 3.1.

\[ \rho = \frac{p}{R \cdot T} \]  \hspace{1cm} (3-1)

- \( p \): absolute pressure
- \( \rho \): density
- \( R \): gas constant

3.2 Pressure in a resting fluid

The pressure at a specific spatial point in the water column (see Figure 3-1) is governed by the following equation. Note that the ocean is regarded from a hydrostatic perspective, as well as incompressible. This is feasible as variations in gravity and fluid density are negligible for our purposes: depths of not greater than 100m.

\[ p_1 = \gamma h + p_2 \]  \hspace{1cm} (3-2)

- \( p_n \): pressure at point \( n \)
- \( h = z_2 - z_1 \)
- \( \gamma \): specific weight
3.3 Buoyancy

If a body is completely or partially submerged in a fluid one can call the resultant fluid force acting on it buoyant force. Since the pressure forces of the water increase with water depth, this leads to greater forces acting from the lower part on the body than the upper part, thus resulting in a positive net force (see Figure 3-2):
3.3 Buoyancy

Figure 3-2 Fluid pressure forces from below are larger than above thus resulting in positive vertical force (Munson 2012)

If noting the net resulting force as $F$ one can get a simple Equation that describes a body’s resulting net buoyant force:

$$F = F_B - W$$  \hspace{1cm}  \text{(3-3)}

$W =$ weight of body  
$g =$ gravity

$$F_B = \rho_{\text{water}} \cdot g \cdot V_{\text{body}}$$  \hspace{1cm}  \text{(3-4)}

If the buoyancy force of the body and the weight force are equal, the resulting force $F$ is zero, thus the body is neutrally buoyant and floats at an equilibrium. If the net force is negative the body sinks, and when positive the body ascends in the water column. Further simplifying yields an even simpler expression by using the specific weight of the fluid (see Figure 3-3):

$$F_B = \gamma \cdot V_{\text{body}}$$  \hspace{1cm}  \text{(3-5)}
3.4 Bernoulli’s Equation

When assuming an incompressible flow, the Bernoulli equation for a steady, inviscid flow is used:

\[ p + \frac{1}{2} \rho V^2 + \gamma z = \text{constant along streamline} \]  \hspace{1cm} 3-6

Applying this to any two points on a streamline then yields:

\[ p_1 + \frac{1}{2} \rho V_1^2 + \gamma z_1 = p_2 + \frac{1}{2} \rho V_2^2 + \gamma z_2 \]  \hspace{1cm} 3-7

### 3.4.1 Free Jet Flow

One typical application of this equation is the flow from a reservoir of a liquid. A jet then forms at the nozzle, flowing with the diameter \( d \) and velocity \( V \) (see Figure 3-4).
Applying Equation 3-7 between points (1) and (2) on a streamline yields:

\[ \gamma h = \frac{1}{2} \rho V^2 \]

Assuming of course that \( z_1 = h, z_2 = 0, V_1 = 0 \) and \( p_1 = 0 \), as well as \( p_2 = 0 \).

Solving for \( V \) gives:

\[ V = \sqrt{\frac{2\gamma h}{\rho}} \]

Solving this equation as if it was a closed reservoir, only the assumptions need to be modified.
3.5 Drag

If an object experiences movement and/or flow through an enclosing liquid, resistive forces arise, acting on the object in the direction of the relative fluid velocity (i.e., counteracting to the objects transition direction). The force depends on the objects form and generally, in order to apply the following equation, the object has to have a blunt form factor as well as the fluids reynolds number needs to be large enough. Both is the case for the ROSCo.

\[ F_D = \frac{1}{2} \rho v^2 C_D A \]  \hspace{1cm} (3-9)

\( F_D \) = Drag force  
\( C_D \) = Drag coefficient, dimensionless coefficient  
\( A \) = reference area
Part II

Mathematical Modeling
Chapter 4

Mathematical modelling of an buoyant underwater ROV

This chapter describes the mathematical framework needed to implement a simulation of an ascending or descending track driven vehicle like the ROSCo underwater. The actual simulation uses a Simulink model with a numerical solver. Numerical tools are well suited to yield relatively realistic simulation models of forces underwater and reactions of weighted object to them. This is especially important when the governing equations are higher order differential equations, or even non-linear.

A thorough analysis of the dynamics of building such a system, is needed to yield precise and significant results. When done correctly, these simulation models can save time and money, as resources for real-world prototyping are often limited.

4.1 Governing equations for a buoyant ROV

It is assumed that the system is unaffected by turbulences and currents, as well as the changes of parameters such as temperature. This is of course unrealistic, but the ROSCo is not meant to be deployed under severe weather circumstances or in situations where rip currents might be present. Thus the approximation is feasible. Since there exist no scaled models of the ROSCo, and the ROSCo itself is subject to change, it is assumed that the geometry of the vehicle is somewhat rectangular, with a drag coefficient of slightly less than rectangular, as the ROSCo is certainly not a solid rectangular mass. The ROSCo is assumed to have a mass of about 500kg, with a plow blade installed on the front and heavy payloads present. The buoyant medium will be lifting bags, capsuled in a containment case, thus even when they are fully inflated, they will not change the form of the ROSCo, as seen from the outside.
4.1 Governing equations for a buoyant ROV

Figure 4-1 presents the geometry of the ROSCo at the bottom of the sea at some time $t$ with a Free-Body-Diagram of the present forces.

Taking Newton’s 2nd Law into account one can follow that:

$$ F = m a = F_B - W - F_D $$

$m$ = mass of crawler

Replacing the Forces with $F$ and assuming the crawler is moving with a velocity $v$, and has already transitioned a distance $x(t)$ will result in Figure 4-2:
During ascent or descent the weight of the crawler stays more or less the same during the whole time, given that the amount of air the system looses by bleeding the ballast tanks is relatively small compared to the overall weight, an assumption that can be safely done. As air inside the ballast tanks follows the ideal gas law, and thus changes relative to the position \(x(t)\) one follows that

\[
F_B = f(t) \quad \text{as well as} \quad F_D = g(t)
\]

To understand the system, the interest is in both the crawlers velocity, as well as its position.
4.1 Governing equations for a buoyant ROV

Re-arranging Eq. 4-1 yields:

\[
\ddot{x}(t) = \dot{v}(t) = a = \frac{F_B(t) - W - F_D(t)}{m} \quad 4-2
\]

\[
v(t) = \int \frac{F_B(t) - W - F_D(t)}{m} \, dt \quad 4-3
\]

\[
x(t) = \int \int \frac{F_B(t) - W - F_D(t)}{m} \, dt \, dt \quad 4-4
\]

As noted before, \( F_B \) is a function of time, due to the ideal gas law. Investigating further at the buoyant force yields:

\[
F_B = \rho_{\text{water}} \cdot g \cdot V_{\text{gas}},
\]

In our example \( V_{\text{gas}} \) is the volume of air inside the ballast tanks. Assuming that initially it starts with a fixed volume, the ideal gas law equation can be re-arranged to the following expression for the volume of the air, where pressure changes around the ROSCo as it moves up in the water column:

\[
V_{\text{gas}}(t) = \frac{m(t) \cdot R \cdot T}{p(t) \cdot M} \quad 4-5
\]

- \( p(t) \) = ambient pressure at time \( t \)
- \( m(t) \) = mass of air inside bag at time \( t \).

Using the hydrostatic equation for pressure

\[
p = \gamma h + p_2 \quad \text{knowing that} \quad p_2 = 101325 \text{Pa} \quad \text{the expression of } p(t) \text{ can be:}
\]

\[
p(t) = \rho_{\text{water}} \cdot g \cdot (d_0 - x(t)) + p_2 \quad 4-6
\]

\( d_0 \) = initial depth of ROSCo.

Substituting 4-6 into 4-5:

\[
V_{\text{gas}}(t) = \frac{m(t) \cdot R \cdot T}{(\rho_{\text{water}} \cdot g \cdot (d_0 - x(t)) + p_2) \cdot M} \quad 4-7
\]
With a term describing the volume of the air inside the ballast bags as a function of time \( V_{\text{gas}}(t) \) is shown to be a non-linear function.

Substituting 4-7 into the Equation of the buoyant force caused by the air filled bags then yields:

\[
F_B(t) = \frac{m(t) \cdot R \cdot T}{\rho_{\text{water}} \cdot g \cdot \left( (d_0 - x(t)) + p_2 \right) \cdot M \cdot \rho_{\text{water}}} \\
= \frac{m(t) \cdot R \cdot T}{(d_0 - x(t)) + \frac{p_2}{(\rho_{\text{water}} \cdot g)} \cdot M} \quad 4-8
\]

As well as:

\[
F_D(t) = \frac{1}{2} \rho v(t)^2 C_D A = \frac{1}{2} \rho \dot{x}^2(t) C_D A \quad 4-9
\]

\( C_D = \text{Drag coefficient, dimensionless coefficient} \)
\( A = \text{reference area, constant, for simulation purpose set to 1} \)
\( v(t) = \text{velocity of ROSCo at time } t \)
\( \rho = \text{density of water} \)

Substituting both 4-9 and 4-8 into 4-2 yields:

\[
\ddot{x}(t) = \frac{\frac{m(t) \cdot R \cdot T}{(d_0 - x(t)) + \frac{p_2}{(\rho_{\text{water}} \cdot g)} \cdot M} - W - \frac{1}{2} \rho \dot{x}^2(t) C_D A}{m} \quad 4-10
\]

This is a 2\(^{\text{nd}}\) order ordinary differential equation term, describing the movement of ROSCo as a function of time and also taking gas expansion and drag into account.

Integrating once results in:
4.1 Governing equations for a buoyant ROV

\[ \dot{x}(t) = \int \frac{m(t) \cdot R \cdot T}{(d_0 - d(t)) \cdot \rho_{water} \cdot g \cdot M} \, dt \]

Integrating twice yields:

\[ x(t) = \int \int \frac{m(t) \cdot R \cdot T}{(d_0 - d(t)) \cdot \rho_{water} \cdot g \cdot M} \, dt \, dt \]
4.2 Inflation/Deflation Rate of Ballast Bags

Taking a closer look at equation 4-12, a(t), v(t) and x(t) have a discontinuity whenever
\((\rho_{\text{water}} \cdot g \cdot (d_0 - x(t)) + p_2) = 0\). Moreover, the volume experiences rapid growth as \(x(t)\) is nearing the surface, due to nonlinear nature of the function.

Thus, the non-linearity is what causes dangerous run-away ascents if a positively buoyant system is not actively controlled. In order to control the system, the ABCS alters the volume of the gas inside the bag. This is done by bleeding the bag, because a change in \(m(t)\), results in a change of the volume of the gas:

\[
\Delta V_{\text{gas}} = V_{\text{gas}}(t_1) - V_{\text{gas}}(t_2) = \frac{[m(t_1) - m(t_2)] \cdot R \cdot T}{(\rho_{\text{water}} \cdot g \cdot (d_0 - x(t)) + p_2) \cdot M}
\]

The difference of the mass of air at two different time steps is thus:

\[
m = [m(t_1) - m(t_2)]
\]

and

\[
m_2(t) = m_1(t) - \bar{m}
\]

For \(\bar{m}\) of an inflation, the mass flux is easily determined:

\[
\dot{m}_{\text{infl}} = \dot{V} \cdot \rho
\]

Where the volumetric flow rate \(\dot{V}\) is determined by the manufacturer of the valve, a function of the valves diameter, the operating pressure, as well as the temperature. The maximum flow is constant since these variables do not change. It is assumed to be the case when the vehicle ascends.

The mass flow rate of a bag deflation, \(\dot{m}_{\text{defl}}\), however does not remain constant, but is a time dependent function:

\[
k_{\text{defl}}(t) = \dot{m}_{\text{infl}} = A_{\text{valve}} \cdot v_{\text{flow}}(t) \cdot \rho_{\text{air}}(t)
\]

\(v_{\text{flow}}(t) = \text{velocity of flow when leaving pressurized tank}\)
4.2 Inflation/Deflation Rate of Ballast Bags
Assuming $v_{\text{flow}}(t)$ is steady and well developed during a time step $\Delta t$ gives the diagram depicted in Figure 4-3:

Using Bernoulli yields:

$$ p_1 + \frac{1}{2} \rho V_1^2 + \gamma z_1 = p_2 + \frac{1}{2} \rho V_2^2 + \gamma z_2 $$

$z_0 = z_1, p_2 = 0, v_1 = 0, \rho_1 = \rho_2$ as $\Delta t$ becomes small

Thus:

$$ p_1 = \frac{1}{2} \rho V_2^2 $$

Knowing that $dP = p_1 - p_2 = p_1 = \frac{F}{A} = \frac{F_{B(t)}}{A}$ and rearranging 4-13 then yields for the velocity of the outgoing flow:
4.2 Inflation/Deflation Rate of Ballast Bags

The velocity of flow going out of valve is given by:

\[
v_{\text{flow}}(t) = \frac{2 \cdot (d_p)}{\sqrt{\rho_{\text{air}}(t)}} = \frac{2 \cdot F_B(t)}{A_{\text{valve}}} \cdot \rho_{\text{air}}(t)
\]

Where:

- \( v_{\text{flow}}(t) \) = velocity of flow going out of valve
- \( d_p \) = differential pressure
- \( \rho_{\text{air}}(t) \) = density of air at time \( t \)
- \( A_{\text{valve}} \) = valve area
- \( F_B(t) \) = force acting on the ballast bag

Inserting 4-15 into 4-13 results in:

\[
k_{\text{defl}}(t) = A_{\text{valve}} \cdot \sqrt{\frac{2 \cdot F_B(t)}{A_{\text{valve}}} \cdot \rho_{\text{air}}(t)}
\]

With

\[
\rho_{\text{air}}(t) = \frac{p(t)}{R \cdot T} = \frac{\rho_{\text{water}} \cdot g \cdot (d_0 - x(t)) + p_2}{R \cdot T}
\]

and

\[
F_B(t) = \frac{m(t) \cdot R \cdot T}{((d_0 - x(t)) + p_2 / (\rho_{\text{water}} \cdot g)) \cdot M}
\]
4.2 Inflation/Deflation Rate of Ballast Bags

One can thus follow:

\[
\frac{\frac{m(t) \cdot R \cdot T}{\rho_{\text{air}}(t)} - 2 \cdot \frac{(d_0 - x(t)) + \bar{p}_2/(\rho_{\text{water}} \cdot g)}{A_{\text{valve}}} \cdot M \cdot A_{\text{valve}} \cdot \rho_{\text{air}}(t)}{\rho_{\text{air}}(t)}
\]

\[
= A_{\text{valve}} \cdot \sqrt{\frac{R \cdot T}{2 \cdot \frac{(d_0 - x(t)) + \bar{p}_2}{\rho_{\text{water}} \cdot g}} \cdot M \cdot A_{\text{valve}} \cdot \rho_{\text{air}}(t)}
\]

\[
= A_{\text{valve}} \cdot \frac{R \cdot T}{1} \cdot \sqrt{\frac{2 \cdot m(t)}{M \cdot A_{\text{valve}} \cdot \rho_{\text{air}}(t)}}
\]

\[
k_{\text{defl}}(t) = A_{\text{valve}} \cdot \frac{R \cdot T}{(d_0 - x(t)) + \bar{p}_2} \cdot \sqrt{\frac{2 \cdot m(t)}{M \cdot A_{\text{valve}} \cdot \rho_{\text{air}}(t)}}
\]

\[
= A_{\text{valve}} \cdot \frac{2 \cdot m(t)}{M \cdot A_{\text{valve}}} \cdot \rho_{\text{air}}(t)
\]

It will now be assumed that the flow is lowering the mass of air inside the bag, which then yields:

\[
k_{\text{defl}}(t) = -A_{\text{valve}} \cdot \frac{2 \cdot m(t)}{M \cdot A_{\text{valve}}}
\]
This provides an expression for the deflation rate of the ballast bag for time \( t \). For accurate testing and simulation the mass of air inside the bag must be monitored, it is one of the parameters that influences the volume of the air causing a buoyant force.

The amount of air that leaves or enters the bag will be defined as:

\[
\dot{m} \cdot \Delta t = m_{\text{step}}
\]

It can be assumed that at every timestep \( \Delta t \) there exist three mutually exclusive options of action:

1) Inflating the bag = \( \dot{m} \cdot \Delta t = k_{\text{infl}} \cdot \Delta t \cdot \text{input} \)
2) Deflating the bag = \( \dot{m} \cdot \Delta t = -k_{\text{defl}}(t) \cdot \Delta t \cdot \text{input} \)
3) Do nothing = \( \dot{m} \cdot \Delta t = 0 \)

Where \( \text{input} \) is a variable number : \(-1 \leq \text{input} \leq +1\), representing the percentage of the valve open. I.e. \( \text{input} = 0.5 \) means the valve is 50% open for the time step.

The mass of the air inside the bag at time \( t \) is a sum of inflations/deflations/no operations, which are added up over the series of \( n \) time-steps.

\[
m(t, \text{input}) = \sum_{t=0}^{n} [K_f \cdot \Delta t \cdot \text{input}_{t=n}] \quad \text{with} \quad \begin{cases} \text{input} < 0 : K_f = k_{\text{infl}} \\ \text{input} > 0 : K_f = -k_{\text{defl}}(t) \\ \text{input} = 0 : K_f = 0 \end{cases}
\]
4.3 Useful Variables

As the exact specifications are still changing, certain parameters of the system are assumed to be fixed and constant (see table 4-1). The exact numbers are of importance in a real live product, but for the sake of testing, developing and simulation convenient numbers where chosen whenever possible.

<table>
<thead>
<tr>
<th>Variable</th>
<th>Name</th>
<th>Value</th>
<th>Unit</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>W</td>
<td>Weight of ROSCo</td>
<td>4900</td>
<td>N</td>
<td>Based upon a mass of 500 kg</td>
</tr>
<tr>
<td>m</td>
<td>Mass of ROSCo</td>
<td>500</td>
<td>kg</td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>Orthogonal Area of ROSCo</td>
<td>1</td>
<td>m²</td>
<td>A convenient number that simplifies equations for simulation</td>
</tr>
<tr>
<td>ρwater</td>
<td>Density of water</td>
<td>1025</td>
<td>Kg/m³</td>
<td></td>
</tr>
<tr>
<td>T</td>
<td>Temperature of Water</td>
<td>20°</td>
<td>Degree Celcius</td>
<td>Typical Assumption for sea water temperature</td>
</tr>
<tr>
<td>p_{atmospheric}</td>
<td>Atmospheric pressure at sea</td>
<td>101325</td>
<td>Pa</td>
<td></td>
</tr>
<tr>
<td>d</td>
<td>Nozzle diameter of Valves</td>
<td>2</td>
<td>mm</td>
<td>This is the actual diameter of the solenoid valve used. Changing this value for convenience is highly discouraged as this will have huge impacts on the system</td>
</tr>
</tbody>
</table>

Table 4-1: Useful variables defined
4.3 Useful Variables
Part III

System Analysis and Control Design
Chapter 5

Analysis of the mathematical model of the physical system

Now with mathematical expressions for important parameters such as velocity and buoyant force over time, this section uses Simulink to analyze certain properties of the physical system. Because of the buoyancy force and the drag force

\[ F_B(t) = \frac{m(t)RT}{(\text{d} - x(t)) + \frac{p_2}{\rho_{\text{water}} g})} \cdot M \]

\[ F_D(t) = \frac{1}{2} \rho v(t)^2 C_D A \]

The underlying system non-linear system is known. So in order to obtain the velocity of the system the following equation is used:

\[ \dot{x}(t) = \left( \frac{m(t)RT}{(\text{d} - x(t)) + \frac{p_2}{\rho_{\text{water}} g})} \cdot M - W - \frac{1}{2} \rho x^2(t) C_D A \right) \cdot \frac{1}{m} \cdot \int \dot{x}(t) \, dt \]

Integrating twice yields the position of the vehicle:

\[ x(t) = \left( \frac{m(t)RT}{(\text{d} - x(t)) + \frac{p_2}{\rho_{\text{water}} g})} \cdot M - W - \frac{1}{2} \rho x^2(t) C_D A \right) \cdot \frac{1}{m} \cdot \int \int \dot{x}(t) \, dt \, dt \]

5.1 Short introduction to numerical analysis and application to ODEs

Simulink is used to analyze the system with a numerical solver named ode45. There exists a vast number of differential equations that cannot be solved analytically, especially in engineering and science. In many cases it is possible to find a numeric
approximation that suffice to solve a problem. With the computational power of an inexpensive desktop computer these problems can generally be solved quickly and accurately.

Ode45 is the default choice of Simulink for numerical solving nonstiff differential equations, and is based on the Dormand-Prince method. The Dormand-Prince method is a member of the family of the Runge-Kutta methods, which are a set of implicit and explicit iterative methods. They are a class of one-step algorithms. Most often used for discretizing in temporal domain, they are well suited for approximating solutions of ODEs. Even though invented around 1900 they are still heavily used.

Runge-Kutta Methods

Before analyzing the actual system in Simulink the Runge-Kutta methods and one-step algorithms are discussed:

For a first or second order differential equation of the form

\[ y' = f(x, y) \]

the general solution is found, that satisfies an initial condition

\[ y(x_0) = y_0, \]

if \( x_0 \) and \( y_0 \) are two real numbers.

Both together, differential equation and initial condition are called: initial value problem.

One-step Methods

One-step methods express a value \( y_{n+1} \) in means of the previous value \( y_n \). A simple one-step method is the Euler Method:

Beginning with

\[ y(x_0) = y_0 \]

and assuming to have calculated up to a value \( y_n \), where \( 0 \leq n \leq N - 1, N \geq 1 \).

This is defined as(Endre 2003):

\[ y_{n+1} = y_n + h \cdot f(x_n, y_n) \]
Calculated in succession from \( n = 0, 1, \ldots, N - 1 \) with a step at a time an approximation of \( y_n \) is obtained. As the number of steps per interval becomes larger, the accuracy increases (Figure 5-1):

![Figure 5-1](image)

**Figure 5-1 Exact solution (solid) and results with Euler method, smaller points means more steps (Endre 2003)**

It is possible to express a one-step method more generally when written in the form of:

\[
y_{n+1} = y_n + h \Phi(x_n, y_n; h), \quad n = 0, 1, \ldots, N - 1, \quad y(x_0) = y_0,
\]

with \( \Phi(x_n, y_n; h) \) just being a continuous function of its variables.

The Euler Method uses \( \Phi(x_n, y_n; h) = f(x_n, y_n) \) in the equation.

The Euler Method is efficient to calculate and simple, in fact one only needs to evaluate \( f \) once at \((x_n, y_n)\) to yield a value for \( y_{n+1} \). This comes with a price: The Euler Method is only first-order accurate. The order of accuracy is a quantification of the rate of convergence in numerical analysis. It describes how a numerical approximation converges to the exact solution. If a solution is \( n \)-th order accurate that means the error \( \sim h^n \). As the step-sizes become small it thus can make a huge difference of which order of accuracy a solution is, especially if accurate results are required.

For a more accurate solution Runge-Kutta methods can be used. These methods have decreased efficiency in favor of higher accuracy. A trade-off that seems feasible with regard to the computational capabilities of present computers. The difference is that a Runge-Kutta method does not just compute the \( f(x_n, y_n) \) but instead re-computes it for intermediate points between \((x_n, y(x_n))\) and \((x_{n+1}, y(x_{n+1}))\).
One example is this family of methods:

\[ y_{n+1} = y_n + h(ak_1 + bk_2) \]

With

\[ k_1 = f(x_n, y_n) \]
\[ k_2 = f(x_n + ah, y_n + bhk_1) \]

Looking closer at this methods, it can be seen that the Euler Method itself is a member of this family (e.g., choosing \( a = 1 \) and \( b = 0 \)).

One very famous and often used method is of the Runge-Kutta family is the following fourth-order method (Endre 2003):

\[ y_{n+1} = y_n + \frac{1}{6}h(k_1 + 2k_2 + 2k_3 + k_4) \]

with

\[ k_1 = f(x_n, y_n) \]
\[ k_2 = f(x_n + \frac{1}{2}h, y_n + \frac{1}{2}hk_1) \]
\[ k_3 = f(x_n + \frac{1}{2}h, y_n + \frac{1}{2}hk_2) \]
\[ k_4 = f(x_n + h, y_n + hk_3). \]

\( k_2 \) and \( k_3 \) are approximations to the first derivative \( y' \) at points on the solution that lie between \((x_n, y(x_n))\) and \((x_{n+1}, y(x_{n+1}))\) and \( \Phi(x_n, y_n; h) = \frac{1}{6}h(k_1 + 2k_2 + 2k_3 + k_4) \) weights and adds all the \( k_i \) values up to result in a weighted average.

After this brief excursion, the reader now has a better understanding what Simulink’s ode45 does:

- It is a numerical solver, thus evaluating the function in a timed series of very small time-steps in a manner that is based on the Dormand-Prince method (also known as DOPRI), which is a Runge-Kutta member that uses six functions.
- DOPRI obtains both a fourth- and a fifth-order accurate solution. It then calculates the difference between these two solutions to yield a result for the error of the fourth-order solution. This algorithm works especially well if one needs adaptive step sizes while integrating.
In addition to ode45, Simulink also offers other methods, such as Rosenbrock method for stiff differential equations, trapezoidal rule for moderately stiff differential equations and differential algebraic equations, as well as various Runge-Kutta and Dormand-Prince methods with fixed step sizes.
5.2 Analysis and results

Simulink/ode45 was tasked with the following scenario: The ROSCo is idling at a depth of 100 meter, which is defined as reference $d_0 = 0$. The operator either deflates or inflates the ballast tanks to a fixed amount of air. The response expected is either descending, corresponding to a deflation of tanks, or ascending, corresponding to inflation of tanks. What is unknown is how the system will respond. The primary parameters of interest are the velocity and depth of the ROSCo as well as the forces acting on it. These parameters will stay fixed during all tests, as they are not controllable by the system (see table 5-1).

<table>
<thead>
<tr>
<th>Variable</th>
<th>Name</th>
<th>Value</th>
<th>Unit</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>$W$</td>
<td>Weight of ROSCo</td>
<td>4900</td>
<td>N</td>
<td>Based upon a mass of 500 kg</td>
</tr>
<tr>
<td>$W_{\text{apparent}}$</td>
<td>Apparent weight of ROSCo</td>
<td>50</td>
<td>N</td>
<td>This number depends on the inherent buoyancy of the crawler, which is a function of its materials and geometry. 50N seems to be a reasonable assumption, as a diver can lift the crawler easily when underwater</td>
</tr>
<tr>
<td>$\rho_{\text{water}}$</td>
<td>Density of water</td>
<td>1025</td>
<td>Kg/m$^3$</td>
<td></td>
</tr>
<tr>
<td>$d_0$</td>
<td>Initial depth</td>
<td>100</td>
<td>m</td>
<td></td>
</tr>
<tr>
<td>$A$</td>
<td>Orthogonal Area of ROSCo</td>
<td>1</td>
<td>m$^2$</td>
<td>A convenient number that simplifies equations for simulation</td>
</tr>
<tr>
<td>$T$</td>
<td>Temperature of Water</td>
<td>20$^\circ$</td>
<td>Degree Celsius</td>
<td>Typical Assumption for sea water temperature</td>
</tr>
<tr>
<td>$P_{\text{atmospheric}}$</td>
<td>Atmospheric pressure at sea surface</td>
<td>101325</td>
<td>Pa</td>
<td>This value will be assumed to be subtracted from every pressure reading, thus having a relative zero pressure at sea surface.</td>
</tr>
</tbody>
</table>

Table 5-1: Fixed parameters of the test
5.2 Analysis and results
5.2 Analysis and results

5.2.1 Test #1 System response to positive and negative fixed buoyancy

Test #1 begins with fixed amounts of air, either having empty or filled tanks. The following parameters were used for this specific test (shown in table 5-2):

<table>
<thead>
<tr>
<th>Variable</th>
<th>Name</th>
<th>Value</th>
<th>Unit</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>( c_d )</td>
<td>Drag coefficient of ROSCo</td>
<td>0.45</td>
<td></td>
<td>Assumed value, based upon a rectangular shape, but a construction of frames rather than a solid shape.</td>
</tr>
<tr>
<td>( m_{air} )</td>
<td>Mass of air inside tanks</td>
<td>Either full or empty (0kg or 4kg)</td>
<td>kg</td>
<td></td>
</tr>
</tbody>
</table>

Table 5-2: Parameters for the test

Figure 5-2 Depth VS Time – empty tanks over test period, negative buoyancy
5.2 Analysis and results

Figure 5-2 describes the behavior of depth over-time, as the ballast tanks are empty the ROSCo quickly starts to descent, and does so in a linear manner once the force of drag and the weight and buoyant forces come to an equilibrium at a certain velocity as shown in Figures 5-3 and 5-4 respectively:

![Force VS Time - empty tanks over test period, negative buoyancy](image1)

Figure 5-3 Force VS Time – empty tanks over test period, negative buoyancy

![Velocity VS Time - empty tanks over test period, negative buoyancy](image2)

Figure 5-4 Velocity VS Time – empty tanks over test period, negative buoyancy
5.2 Analysis and results

Figure 5-4 Velocity VS Time – empty tanks over test period, negative buoyancy

The velocity \( y \) settles to a constant value that depends on \( C_d \) and the inherent buoyant force of the ROSCo as well as its weight. The system remains stable during the whole testing period, and once it settles continues to descent at a constant rate, until it collides with the bottom of the sea.

Starting the test with an already inflated bag, that is left un-altered during the course of the test. The following result is obtained (Figure 5-5).

Figure 5-5 Depth VS Time – Fixed amount air, positive buoyancy

Interpreting Figure 5-5, a noticeable difference to the previous test is observed: the rate at which the vehicle changes depth is subject to a positive change, i.e. the vehicle accelerates. This becomes evident looking at Figure 5-6:
The force on the crawler begins with a positive number, because of the initial air filling. It accelerates up to a force equilibrium when the drag force is in its full effect. But as it continues to rise $F_B$ reaches a region where the volume of the air inside the tanks starts to rapidly expand due to:

$$F_B(t) = \frac{m_{\text{air}}(t) \cdot R \cdot T}{\left((d_0 - x(t)) + p_2/(\rho_{\text{water}} \cdot g)\right) \cdot M}$$

Thus $F_B$ becomes large when the ROSCo comes in proximity of the surface, i.e. the point of relative zero pressure (see Figure 5-7 and 5-8).
5.2 Analysis and results

Figure 5-7 Buoyant force VS Time – fixed amount of air, positive buoyancy

Figure 5-8 Velocity force VS Time – fixed amount of air, positive buoyancy
This explains a sudden steep increase of the force acting on the crawler, and is accompanied by acceleration: After settling at the equilibrium velocity, the velocity starts to grow, up to a point where it leaves the charts, in close proximity of the surface. The system clearly shows unstable behavior, and when left unregulated will lead to a the runaway ascent. This can damage the vehicle and be potentially fatal for divers in the area.

5.2.2 Test #2 System response to various positive buoyancies

Wanting to control the ROSCos vertical position, the first goal is to hold the value of velocity constant. Assuming the drag coefficient, shape and weight of the crawler are fixed, only one value can change: the amount of air inside the bag. The following test thus assumes different filling masses of air, but does not change them over the course of the test. The four different numbers are chosen in a way that 4kg represents a relatively full tank, while 1.7kg is a tank that barely gives the ROSCo a positive buoyancy when positioned at 100m depth. Analyzing figures 5-8 to 5-11 shows that the system is unstable. It is irrelevant if the positive buoyant force is slightly or significantly greater than zero. As long as the system is positively buoyant the ROSCo rises, and proceeds to a point where it becomes unstable and “runs away”. Changing the initial amount of air in the bag just increases the rising time and thus delays the point of “tipping” time wise.
5.2 Analysis and results

Figure 5-9 Velocity vs Time – Varying amounts of air, fixed during test, positive buoyancy

Figure 5-10 Force vs Time – Fixed amount of air, positive buoyancy
5.2 Analysis and results

Figure 5-11 Buoyant force VS Time – fixed amount of air, positive buoyancy
5.2 Analysis and results

5.2.3 Test #3 System response to sinusoidal changing input

Test #3 tests the systems response to a sinusoidal changing input to achieve more knowledge about the systems’ behavior a simulation is made with a timed input signal to the system. A sinusoidal wave is a typical standard input to estimate a systems response to frequency changes (Golnaraghi 2009. The simulation uses the mathematical model for deflation/inflation rates of the tanks:

\[
m(t, \text{input}) = \sum_{t=0}^{n} [K_f \cdot \Delta t \cdot \text{input}] \quad \text{with} \quad \begin{cases} \text{input} < 0 : K_f = k_{\text{infl}} \\ \text{input} > 0 : K_f = -k_{\text{defl}}(t) \\ \text{input} = 0 : K_f = 0 \end{cases}
\]

Following parameters are chosen to stay fixed (see table 5-3):

<table>
<thead>
<tr>
<th>Variable</th>
<th>Name</th>
<th>Value</th>
<th>Unit</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>(C_d)</td>
<td>Drag coefficient of ROSCo</td>
<td>0.45</td>
<td></td>
<td>Assumed value, based upon a rectangular shape, but a construction of frames rather than a solid shape.</td>
</tr>
<tr>
<td>(A_{\text{input}})</td>
<td>Amplitude of input</td>
<td>0.2</td>
<td></td>
<td>Can range from -1 to 1, and expresses if valve fully opened to inflate or deflate.</td>
</tr>
<tr>
<td>(f)</td>
<td>Frequency of input signal</td>
<td>Varying</td>
<td>Rad/s</td>
<td></td>
</tr>
<tr>
<td>(m_{\text{air}})</td>
<td>Mass of air inside tanks</td>
<td>Varying</td>
<td>kg</td>
<td>The amount of air will be changing over time but only indirectly, depending on the frequency of the input change.</td>
</tr>
</tbody>
</table>

Table 5-3: Parameters of test #3
5.2 Analysis and results

Figure 5-12 Force VS Time – Varying frequencies

Clearly visible on figure 5-12 is a decay of the signal, which corresponds to a settling of the signal. This is more significant for higher frequencies, where the system settles as frequencies become large. However, if the frequency becomes low, the amplitude and damping becomes less. The system is then under damped. Any frequency lower than 0.04[rad/s] shows under damped behavior, eventually leading to unstable behavior and escalation.
5.3 Frequency response

After discussing the systems response to low frequencies of input changes in section 5.2 and having discovered the instabilities in the system, this section investigates the frequency response of the system. This is visible on figure 5-13 where the gain is 4.93db with a phase of -180deg.

Figure 5-13 Frequency response Bode plot

When identifying instabilities on a Bode plot, one technique is to look at the gain margin. The gain margin is found by locating the frequency when the phase reaches -180 deg. If the found frequency produces a gain of \( \geq 1 \), the system is unstable. To visualize the meaning of this: With a gain of 4.93db where phase is -180 deg, the system is unstable. This is congruent with the analysis in 5.2.3. Having a look at a pole-zero map on figure 5-14 there exist poles in the real half of the plot - this is a sign of instability (Golnagahri 2009). Real
poles on the positive right-half of the s-plane, indicate that the exponential part of the systems’ response rises towards infinity.

Figure 5-14 Pole-Zero Map

5.4 Conclusion

After doing a numerical analysis of the system and carrying out various methods to determine stability the system is observed to have an unstable behavior. This makes the task of designing an control algorithm difficult.
Chapter 6  Approaches to Control the System

Chapter 6 focuses on developing an approach to control the velocity and position of the system. Various approaches are investigated and their feasibility discussed, each are developed and tested using the implemented model/simulator.

6.1  Brief introduction to control systems

Electronic control systems are a standard method of controlling the actions of a system. They control temperature regulators in an air-conditioning system to the control systems in an airplane, as well as in industrial manufacturing and processing facilities. Generally a control system consists of some sort of device that regulates or directs the behavior of a system. See Figure 6-1 for a simplified control system.

![Figure 6-1 Simplified control system (Nise 2010)]

6.2  Open-Loop control

One type of control system is the Open-Loop system (Figure 6-2). This control system accepts an input with an input transducer which converts this input to a form that can be processed by the controller. The controller then drives a plant – the actual system that is controlled. Noise and disturbances can be added to the system to model real-life inputs during simulation. Open-Loop systems are simple and inexpensive to build and do very well on many occasions, a classic example being the household washing machine. Unfortunately Open-Loop systems cannot compensate noise and disturbances (Nise 2010). They are more or less on a “fixed” track towards their goal (e.g., a washing machine is programmed to wash 50 minutes. Whether the clothes are actual clean before that time is over, or getting destroyed in the process of washing - it does not affect the systems response, and it will continue washing until 50 minutes are over even).
6.3 Feedback control

In the case of the ROV, as seen with Test#1, an Open-Loop approach to controlling the system, as in filling the bags with a certain amount of air on the bottom and then waiting until the ROSCo rises to the surface, does not perform very well. Inflating the bag and then deflating at a fixed time-rate of change does not take any changes in crawler mass into account as well and might lead to the ROSCo deflating the bags to a point where it reaches negative buoyancy again and sinks to the bottom – a failed surfacing attempt possibly destroying valuable artefacts on the sea bottom.

6.3 Feedback control

In order to eliminate those drawbacks, a feedback loop has to be employed. See Figure 6-3 for a simple feedback control system. This is done by closing the open-loop system. The input transducer again converts input to controller input. In addition there now exists an output transducer as well, as output response is also converted to a form that can be used by the controller. Then the output signal is fed back via the feedback path and algebraically added to the input signal.

The system can thus compensate for noise and disturbance by utilizing the feedback path to compare a systems response to the desired response and actuating the signal to correct. Feedback systems are more accurate and less sensitive to noise and parameter changes as well as disturbance. They are however more difficult to implement and can be more
expensive. In some cases they are needed though, as an open-loop system does not deliver the results that are desired (Nise 2010).
Chapter 7

Introduction to Proportional Integral Derivative control

7.1 Introduction

A proportional-integral-derivative controller represents a certain type of feedback control system. It controls the system by computing an error term, based upon the output and the input and then adjusting the system to minimize the error. A typical PID inner block diagram might look like figure 7-1:

![PID controller with filtering of derivative gain (Simulink)](image)

Figure 7-1 PID controller with filtering of derivative gain (Simulink)
A simplified version of the system's implementation on the ROSCo is shown in figure 7-2:

![Simplified PID system](image)

**Figure 7-2 Simplified PID system**

The terms P, I, D can be explained as:

- **P** = present error
- **I** = sum of past errors over time
- **D** = range of change of errors

These three values are added up to a weighted sum to create the controllers output:

\[
\text{output}(t) = K_p e(t) + K_i \int_0^t e(\tau) d\tau + K_d \frac{d}{dt} e(t)
\]

with

- \( K_p \) = *proportional gain*
- \( K_i \) = *integral gain*
- \( K_d \) = *derivative gain*
- \( e \) = error
- \( t \) = time
- \( \tau \) = *integration variable*

### 7.2 Proportional Term

The proportional term represents the present error of the system. When the proportional gain is high, a given output will result in large change. Tuning the proportional gain must be done with care as the system can become unstable when proportional gain is too high. However, when the proportional gain is too small, it results in small gains, where even a large error will be only having small influences on the system resulting in decreased responsiveness or lag. Usually the proportional term represents the majority of the output result.
7.3 Integral Term

The integral term is defined as the sum of the momentarily errors over time, and represents the amount of error the system has collected over time. The integral term minimizes steady state errors such as when the system responds with small errors, these will overtime sum up and initiate the controller to drive the system differently.

7.4 Derivative Term

The derivative term is the rate of change of error. Increasing its gain can improve settling time and stability of the system. Unfortunately, it can also alter a system’s stability. A noisy environment can cause the derivative term to cause unwanted activity from the controller and thus result in erratic behavior, consequently lowering stability and also performance (Ang 2005).

7.5 Stability and optimal behavior

Optimally the PID regulates the controlled variable with rejection of disturbances and staying at the setpoint. It’s performance is also measured by how fast the controller reaches the region of setpoint, the so called rising time. Another criterion of measuring the PID’s performance is to analyze potential time it takes the signal to settle at the setpoint if overshoot occurred or the signal is not damped enough and oscillates into settling.

A PID does not always behave and perform well. If parameters are chosen incorrectly its output can start to oscillate and/or diverge, leading to potential mechanical failure or saturation. The system is then unstable, most often caused by exceeding high values of gain and especially in presence of significant lag.

7.6 Loop tuning

As different parameters can lead to undesired behavior, and changing those parameters can make the same system well controlled, adjusting the parameters is very important when designing a PID controller. The actual process of tuning might seem a simple task since it involves only three variables, but in fact the system can show different behavior and constraints on the mechanical side.

Some objectives such as short transient time and high stability are even conflicting with each other: As the controller shortens its transient by usually increasing the proportional gain, this will most likely lead to less stability. Iteration through different design stages until desired results are met is therefore necessarily. Nonlinearities can also rise to problems, causing PIDs to change their behavior if certain conditions are met. A set of certain
surrounding conditions defines a controllers operating point, this will be important for gain scheduling.
The task of finding the right parameters through manually tuning is greatly simplified with the already built mathematical model implemented in Simulink.

### 7.7 Gain scheduling

In order to overcome nonlinearities a technique named gain scheduling can be utilized. Instead of using one set of gain factors for the three terms of the PID there will be multiple sets. Each of these sets is a specific set of values that works well at a certain operating point of the controller and results in satisfying control. The controller uses a scheduling variable, which must be an observable variable, to determine at which operating point the controller is at the current time and then chooses the applying set of gain factors. Examples of the scheduling variable are the Mach number of a Space Shuttle, or the depth of the ROSCo.

### 7.8 Limitations and advantages of a PID controller

PID controllers are easy to build, simple and result in good performance for many applications. As many control algorithms a PID controller is reactive. This is because it is an observing controller that measures feedback responses rather than predicting future changes of the system, and thus it is a compromise itself. A controller that can result in better performance can be built by implementing the actual model of the process as part of the controller, to estimate future changes in the systems behavior, and then acting corresponding to those predictions. This so called feed-forward system can then be combined with an PID to regulate errors that still might occur. However, if the model of the system involves nonlinear differential equations, maybe even partial, this can become very difficult and might consume great amounts of computing capacities. The system then becomes unfeasible for many embedded systems, because microcontrollers do not offer the same amount of storage and computing performance as modern desktop computers or simulation workstations. Thus relying on an observing controller such as the PID might be a good choice if computational capacities are limited. As the goal of this thesis is to implement the system in a relatively confined space and with low-cost hardware, a feedback control will be used rather than implementing the mathematical model in a feed-forward manner.
Chapter 8

Introduction to Fuzzy logic

Another approach for a feedback system is Fuzzy Logic inferencing. Fuzzy logic is a feasible approach for non-linear problems where the exact mathematical models causing non-linear behavior are either not known or too complex to implement in a feed-forward loop system. Fuzzy logic bases on a set of language rules, converts these into mathematical equivalents by building a rule matrix. This makes fuzzy logic well suited to successfully control problems with imprecise and incomplete data and/or nonlinear systems of high complexity.

This concept was first introduced as a fuzzy inferencing system by Lotfi Zadeh from UC Berkley as a way of processing data. The method introduces a concept of partial set membership for data values instead of crisp set membership/non-membership. Unfortunately computing capacities on embedded systems were not powerful enough until the late 1970’s to implement actual fuzzy inference systems to apply the algorithm to control systems. Feedback controllers programmed this way can accept noisy and imprecise input and still be effective. In this chapter it will be illustrated how a Fuzzy Logic control system works, with the velocity control of the ROSCo as an example. Another fuzzy system can then be built, using the velocity control system to control position, i.e. achieve hovering at a certain depth.

Figure 8-1 Simplified fuzzy logic system
8.1 Basic principle of Fuzzy Logic

In general, FL is based on simple rules such as “IF X AND Y THEN Z”. This is in contrast to modeling a system mathematically. The model is thus relying on gathering empirical data and operator experience.

No-fuzzy control methods, such as PID, would define the following parameters:

- A setpoint Velocity $SV\,[m/s]$
- A current Velocity of $V\,[m/s]$

and base their control approach on mathematical expressions using those parameters.

FL chooses a different way, defining the process in terms of rules. These might be for example:

- IF (process too fast) AND (accelerating a lot) THEN (decelerate)
- IF (process too slow) AND (decelerating a lot) THEN (accelerate)

The terms are naturally imprecise as one does not have a crisp definition of what “process too slow” actually means. But they are very descriptive of what must happen in the system if certain circumstances are present. FL is thus able to mimic the way human beings control processes but of course at a very high rate.

In order to work FL still needs some numerical values. These values will be called fuzzy parameters or input, and for the ROSCo there will be two of them:

- error = The difference between setpoint Velocity and current Velocity
- $v$-dot = the rate of change of the current velocity

Some further knowledge about the system must be present as well: What is a significant error, what is a significant rate of change of error etc.? However, this is usually not difficult to establish. For the ROSCo, a significant error for the velocity would be 0.1 m/s, as the setpoint velocity is 0.2 m/s and everything that is more than 150% larger than this velocity is significantly larger.

8.2 The Rule Matrix and the Rules

Fuzzy logic uses linguistic variables. These are written variables that describe the fuzzy parameters. The error and error-dot values, for example, can be described as “negative”, “positive” or “zero” – based on the sign of their value. Different linguistic would be possible,
but for our case those three work out well as one can see. A matrix is needed with mappings of the different possible combinations of the linguistic variables applied to both of the fuzzy parameters (Figure 8-2).

The rule matrix is a handy tool to build the actual rules upon, because it makes imagination of different combinations easier. In our case acceleration corresponds to inflating the tanks while decelerating means deflating the tanks.

The rules for our system thus are:

1. IF \((\text{error} > 0)\) AND \((\text{v-dot} < 0)\) THEN \((\text{inflate bags})\)
2. IF \((\text{error} > 0)\) AND \((\text{v-dot} = 0)\) THEN \((\text{inflate bags})\)
3. IF \((\text{error} > 0)\) AND \((\text{v-dot} > 0)\) THEN \((\text{inflate bags})\)
4. IF \((\text{error} = 0)\) AND \((\text{v-dot} < 0)\) THEN \((\text{inflate bags})\)
5. IF \((\text{error} = 0)\) AND \((\text{v-dot} = 0)\) THEN \((\text{no command})\)
6. IF \((\text{error} = 0)\) AND \((\text{v-dot} > 0)\) THEN \((\text{deflate bags})\)
7. IF \((\text{error} < 0)\) AND \((\text{v-dot} < 0)\) THEN \((\text{deflate bags})\)
8. IF \((\text{error} < 0)\) AND \((\text{v-dot} = 0)\) THEN \((\text{deflate bags})\)
9. IF \((\text{error} < 0)\) AND \((\text{v-dot} > 0)\) THEN \((\text{deflate bags})\)
It is noteworthy that it is possible that because of the way one defined the membership functions, as done in chapter 8.3, some of those rules might never apply when the system is running. This is acceptable, as the system might change, membership functions might be tuned and at some time they might apply again, thus there is no need to delete them.
8.3 Membership Functions

Once the rule matrix has led to a set of rules, the next step is applying these rules. To do so, one needs membership functions. A membership function can be seen as a representation of the magnitude of participation of the input parameter in a linguistic variable (Kaehler 1998).

By applying these membership functions to the input parameters it is possible to achieve a weighted output result, which can include functional overlaps. Consequently the values of memberships of the inputs will be used to calculate final outputs which consist of fuzzy output sets. These sets are later defuzzified to generate a crisp output that can drive the controls. Membership functions can be different for inputs and outputs. They can be classified by certain aspects, which can be seen on figure 8-3 (Kaehler 1998):

- **Shape:**
  The shape of the function. A triangular shape is easy to implement and very widespread but there are others: Bell, trapezoidal, and exponential. The more complex a function gets the more computing capabilities needs to be present at the system, as these functions have to be computed several times per time step for several inputs.

- **Height:**
  The magnitude, this values is normalized to 1.

- **Width:**
  The range of values the input value can attain

- **Shouldering:**
  The value can be locked at a certain degree. This leads input values that lie out of the input width to attain the locked value.

- **Overlap:**
  Membership functions can overlap, thus an input value can be a partial member of more than one linguistic value. This is one of the strongest advantages and name giving for the fuzzy logic as it helps creating fuzzy sets.

- **Center:**
  The center of the function on the axis. Needed for defuzzification.
The concept of partial membership and overlapping functions might sound confusing to someone who is new to fuzzy sets, but a simple example should clarify it:

Wanting to describe persons by their size, the input parameter would then be the numerical size in feet and inch. If just using simple non-fuzzy rules this would be a very unresponsive and coarse system:

Tall > 6', Medium = 6'-5', Small < 5'.

The system would fail to yield satisfying results for people that border between a limit: A person measuring 5'11" would be medium, and a person at 6' would be tall – not a very good result.

But one can make the system more accurate by applying membership functions and fuzzy sets. First one needs to define 3 linguistic variables “small”, “medium” and “tall”. Now every person will be part of these three variables to a certain amount.

A person that measures only 4'5" might be 80% small, 20% medium and 0% tall.
A person that measures 7'0" will most certainly be 100% tall, 0% medium and 0% small.
A person that measures 5'9" will most likely be 0% small, 20% medium and 20% tall.

And so on. The amount of membership of any value in those three sets is now a simple question of the value of respective membership function. Thus this is a more accurate description of a person’s height, while still having the advantage of using a simple linguistic variable like “tall” or “small”.

A graphical representation of this simple system might look like figure 8-4, with the “medium” membership function in red. Note that this system uses a triangular membership function.

There are far more membership functions than a triangular function, like gauss membership functions for example, but a triangular membership function is easy to implement and suffices for illustrating the method.
The membership functions for the ABCS are shown in Figures 8-5 to 8-7:
8.4 Generating crisp Output – Defuzzification

Figure 8-6 Membership function for input variable "d(u)dt"

Figure 8-7 Membership function for output variable "output 1"
8.4 Generating crisp Output – Defuzzification

After the membership values have been calculated for the input parameters, the rules are applied. While some rules might be in effect some others might not. Thus they generate different output values. These values are now combined into logical products or sums. One method is choosing the minimum value of both functions. This way, non applying rules produce a zero output.

This will be illustrated by assuming the following input values, which are corresponding to the ROSCo not being fast enough to have reached the desired set point velocity, but also already accelerated to some degree. Also the ROSCo is slowing down, as the rate of change of velocity is negative. One can consequently expect the value of the output to be a high positive number, as the system needs to be accelerated to avoid sinking.

\[
\begin{align*}
\text{error} & = 0.15 \text{ m/s} \\
\text{d(v)dt} & = -0.2 \text{ m/s}
\end{align*}
\]

Thus:

- error: \( mf(\text{negative}) = 0.0, \quad mf(\text{zero}) = 0.0, \quad mf(\text{positive}) = 1.0 \)
- d(v)dt: \( mf(\text{negative}) = 0.6, \quad mf(\text{zero}) = 0.0, \quad mf(\text{positive}) = 0.0 \)

Now these rules are applied to these values, using the minimum method:

1. IF \((error > 0)\) AND \((v\cdotdot < 0)\) : \(\min(1.0,0.6) = 0.6\)
2. IF \((error > 0)\) AND \((v\cdotdot = 0)\) : \(\min(1.0,0.0) = 0.0\)
3. IF \((error > 0)\) AND \((v\cdotdot > 0)\) : \(\min(1.0,0.0) = 0.0\)
4. IF \((error = 0)\) AND \((v\cdotdot < 0)\) : \(\min(0.0,0.6) = 0.0\)
5. IF \((error = 0)\) AND \((v\cdotdot = 0)\) : \(\min(0.0,0.0) = 0.0\)
6. IF \((error = 0)\) AND \((v\cdotdot > 0)\) : \(\min(0.0,0.0) = 0.0\)
7. IF \((error < 0)\) AND \((v\cdotdot < 0)\) : \(\min(0.0,0.6) = 0.0\)
8. IF \((error < 0)\) AND \((v\cdotdot = 0)\) : \(\min(0.0,0.0) = 0.0\)
9. IF \((error < 0)\) AND \((v\cdotdot > 0)\) : \(\min(0.0,0.0) = 0.0\)

Every value yielded by this approach describes the firing strength of a rule.
Now these have to be combined to sums by inferencing and defuzzification to generate a crisp controller output.

Inferencing is the process of combining the rules. Multiple approaches exist (Kaehler 1998).

**Maximum-Minimum:**
A simple approach that selects the maximum or minimum value. The approach does not really take advantage of combining effects of all possible combinations of rules but is easy to implement.

**Averaging:**
This approach is simple too, but has one drawback: if more than one rule fire, this will not be represented in the output, as the average takes away this information.

**Root-sum-squares:**
This method is a good combination of all applicable rules. It is more complex but yields good results, and it will be used in this illustration.

Starting by ordering the rules by their effect on the output is feasible. This means a rule that produces a decelerating output, i.e. a negative value, will be part of the “negative” root sum term, while a rule yielding in a positive value will be part of the “positive” root sum term and so on.

Thus:

\[
\text{positive} = \sqrt{R_1^2 + R_2^2 + R_3^2 + R_4^2} = \sqrt{[0.6^2 + 0.0^2 + 0.0^2 + 0.0^2]} = 0.6
\]

\[
\text{negative} = \sqrt{R_6^2 + R_7^2 + R_8^2 + R_9^2} = \sqrt{[0.0^2 + 0.0^2 + 0.0^2 + 0.0^2]} = 0.0
\]

\[
\text{negative} = \sqrt{R_5^2} = \sqrt{0.0^2} = 0.0
\]

Now it is possible to compute the centroid of the area:

\[
\text{Output} = \frac{\text{center}_{\text{negative}} \cdot \text{strength}_{\text{negative}} + \text{center}_{\text{positive}} \cdot \text{strength}_{\text{positive}} + \text{center}_{\text{zero}} \cdot \text{strength}_{\text{zero}}}{\text{strength}_{\text{negative}} + \text{strength}_{\text{positive}} + \text{strength}_{\text{zero}}}
\]

\[
= \frac{-1 \cdot 0.0 + 0.0 \cdot 0.0 + 1 \cdot 0.6}{0.6} = +1.0
\]
The output is thus +1.0, which is a result that matches exactly our expectation. The inflating of the ROSCos ballast tanks is needed in order to avoid sinking to the bottom of the sea.

Note that output is defined as $-1 < \text{output} < +1$. It represents a percent value of the valve open, over a certain fixed time step. E.g. if $\text{output}_1 = 0.8$ this corresponds to the inflating valve 80% opened for the given time step.

### 8.5 Limitations and advantages of Fuzzy Logic Control

Fuzzy logic is a very robust approach, since it is not relying on noise-free and precise input data. It will also always result in a smooth output, even if the input parameters lie in a wide range, if implemented correct. Changing the way the system behaves can be simply done by changing the rules and/or the parameter ranges or membership functions. One can also simply add more rules and membership functions to the system to implement better performance or new behavior. These features make Fuzzy logic ideal for nonlinear control, as now knowledge has to be present on the control system about the mathematical details of the system that shall be controlled.

There are however some limitations to fuzzy logic control: The system has to do a computational overhead, which is significantly higher than feedback control techniques such as PID. This can be a problem when implementing fuzzy logic on hardware that is less capable in terms of computing capacities. When the complexity of the system increases, as more input parameters are involved, the rule matrix can become quite complicated leading to a further increase in computational overhead.
8.5 Limitations and advantages of Fuzzy Logic Control
Chapter 9  Test results of designed controllers

In the course of the thesis different controller designs were developed. This section will present results of model tests of those designs, achieved with the help of the mathematical model that was developed earlier.

Following parameters of the ROSCo will stay fixed during the test (table 9-1):

<table>
<thead>
<tr>
<th>Variable</th>
<th>Name</th>
<th>Value</th>
<th>Unit</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>W</td>
<td>Weight of ROSCo</td>
<td>4900</td>
<td>N</td>
<td>Based upon a mass of 500 kg</td>
</tr>
<tr>
<td>$W_{\text{apparent}}$</td>
<td>Apparent weight of ROSCo</td>
<td>50</td>
<td>N</td>
<td>This number depends on the inherent buoyancy of the crawler, which is a function of its materials and geometry. 50N seems to be a reasonable assumption, as a diver can lift the crawler easily when underwater</td>
</tr>
<tr>
<td>$\rho_{\text{water}}$</td>
<td>Density of water</td>
<td>1025</td>
<td>Kg/m³</td>
<td></td>
</tr>
<tr>
<td>$d_0$</td>
<td>Initial depth</td>
<td>100</td>
<td>m</td>
<td></td>
</tr>
<tr>
<td>$C_d$</td>
<td>Drag coefficient</td>
<td>0.45</td>
<td></td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>Orthogonal Area of ROSCo</td>
<td>1</td>
<td>m²</td>
<td>A convenient number that simplifies equations for simulation</td>
</tr>
<tr>
<td>T</td>
<td>Temperature of Water</td>
<td>20°C</td>
<td></td>
<td>Typical Assumption for sea water temperature</td>
</tr>
<tr>
<td>$p_{\text{atmospheric}}$</td>
<td>Atmospheric pressure at sea surface</td>
<td>101325</td>
<td>Pa</td>
<td>This value will be assumed to be subtracted from every pressure reading, thus having a relative zero pressure at sea surface.</td>
</tr>
</tbody>
</table>

Table 9-1: Fixed parameters of the test
Figure 9-1 illustrates a simple diagram of the testing system:

![Simplified PID control system for the ROSCo](image)

**Figure 9-1 Simplified PID control system for the ROSCo**

### 9.1 Cruise Control - PID controller

To initialize the cruise control of the PID controller an untuned response of the system to a PID controller was investigated. Parameters are chose to be:

\[
K_p = 1.0, \quad K_i = 1.0, \quad K_d = 1.0, \quad \text{Filter} = 1.0
\]

The response shows very unsatisfying results (figures 9-2 and 9-3), as the system is heavily oscillating and in fact never reaches a steady state. Note that the system exceeds the critical velocity of 0.33 m/s, which is the typical rate-of-ascent of a diver.
9.1 Cruise Control - PID controller

Figure 9-2 Depth VS Time – untuned PID controller

Figure 9-3 Velocity VS Time – untuned PID controller
This test clearly shows that one need to tune the parameters in order to yield better results. First the proportional gain has to be tuned, as this is usually the parameter that effects better performance most heavily. After doing this for a certain number of cycles, eventually the following combination was obtained:

\[ K_p = 150.0, K_i = 1.0, K_d = 1.0, \text{Filter} = 1.0 \]

The change proves successfully, as it results in significant performance gain.

Figure 9-4 Depth VS Time – tuned PID controller
Figure 9-5 shows that the velocity now always stays within the critical velocity of 0.33 m/s, and it even converges to a steady state velocity. However, considerable overshoot and a very poor rising time and steady state error is still present. Consequently there will be a change of the integrative parameter, and end up with the following combination after tuning:

\[ K_p = 150.0, K_i = 0.0001, K_d = 1.0, \text{Filter} = 1.0 \]
Figure 9-6 Depth VS Time – tuned PID controller

Figure 9-7 Velocity VS Time – tuned PID controller
Figure 9-7 shows the velocity vs time for the tuned PID controller, where the rising time is fairly quick compared to the other tests, and there is almost no overshoot. But one can still notice a little bend on the velocity, shortly before it approaches the steady state. To eliminate this, the derivative gain was increased:

\[ K_p = 150.0, K_i = 0.0001, K_d = 15.0, \text{Filter} = 1.0 \]

Figure 9-8 Depth VS Time – tuned PID controller
Figure 9-9 Velocity VS Time – tuned PID controller

Figure 9-9 shows the velocity vs time for the tuned PID controller where response now looks very promising, the bend is almost eliminated, the rising time is quick, and the steady state error is less than 1%.

The system is then analyzed to find a safe operating range. This is done by increasing the set point velocity \( v \) to a value that causes the system to behave unstable. Knowing a safe operating range of the system is important as instability can lead to run-way ascents and uncontrolled sinking of the ROSCo. From figures 9-10 and 9-11 it can be seen that the system starts to become unstable when the set point velocity reaches 0.5 m/s. Even with the velocity increasing during the last centimeters of ascent, it still does not become out of control, and ROSCo still surfaces. Thus 0.4 m/s is defined as the upper bound of the maximum operating range for the set point velocity, resulting in a safety factor of two for a set point velocity of 0.2 m/s.
9.1 Cruise Control - PID controller

Figure 9-10 Depth VS Time – different set point velocities

Figure 9-11 Velocity VS Time – different set point velocities
To further investigate the controller's abilities, another test was performed, this time with added Gaussian noise on two locations: One the one hand the system was subjected to perturbances of velocity, a typical example for this in reality would be an occurring current. On the other hand, the velocity measurement was subjected to noise, which represents a noisy sensor output. The PID controller still performs reasonably good, the system remains stable (Figure 9-12).

Figure 9-12 PID Controller with added noise
A fuzzy logic system has been developed as well, in order to analyze if it results in even better performance. The rule set and defuzzification methods already presented in the example from chapter 8 will be used. The system will thus look like Figure 9-13, which delivers the results shown in Figure 9-14 and 9-15:

Figure 9-13 Simplified fuzzy control system for the ROSCo
9.2 Cruise Control - Fuzzy Logic controller

Figure 9-14 Depth VS Time – tuned fuzzy logic controller

Figure 9-15 Velocity VS Time – tuned fuzzy logic controller
The system produces a noticeable steady state error, specially if it is compared to the PID controllers performance. One can also observe a longer rising time. When comparing both the systems in terms of broadness of possible inputs, one can conclude that the PID is performing better in that regard as well (see Figure 9-16 and 9-17).

Figure 9-16 Velocity VS Time – tuned fuzzy logic controller
To further investigate the fuzzy logic controllers abilities, another test was performed, this time with added gaussian noise on two locations: One the one hand the system was subjected to perturbances of velocity, a typical example for this in reality would be an occuring current. On the other hand, the velocity measurement was subjected to noise, which represents a noisy sensor output. The Fuzzy logic controller still performs reasonably good, the system remains stable, as seen in the former test with the PID controller (Figure 9-18 and 9-19).
9.3 Position Control – Combined Approach

Figure 9-18 Fuzzy logic controller with added noise

Figure 9-19 Comparison of fuzzy logic and PID controller with added noise
9.3 Position Control – Combined Approach

The last two sections of this document presented a feasible approach to achieve a cruise control accompanied with significant test data. However, the ROSCo must be able to perform a partial rise. This means that the ROSCo will stay at a certain depth, navigate to a different location, and then settle again at the bottom. A combined approach is developed to realize the position control capability, consisting of both the fuzzy logic and the PID control.

The general principle is the following:
A fuzzy logic controller will be used to receive an input command with the desired depth. It will then receive feedback value and calculate the defuzzified output, which represents a velocity value in m/s, that is needed to bring the ROSCo near the desired depth. This output will be used to drive a PID controller that regulates the velocity, by using the already developed cruise control approach.

Figure 9-20 illustrates the block model of this position control system.

The PID controller uses the following parameters:

\[ K_p = 150.0, K_i = 0.0001, K_d = 15.0, \text{Filter} = 1.0 \]
Figure 9-20 System model of position control
The Fuzzy Logic system uses the following rule set:

1. IF \((\text{error} > 0)\) AND \((v < 0)\) THEN (accelerate)
2. IF \((\text{error} > 0)\) AND \((v = 0)\) THEN (accelerate)
3. IF \((\text{error} > 0)\) AND \((v > 0)\) THEN (accelerate)
4. IF \((\text{error} = 0)\) AND \((v < 0)\) THEN (accelerate)
5. IF \((\text{error} = 0)\) AND \((v = 0)\) THEN (no command)
6. IF \((\text{error} = 0)\) AND \((v > 0)\) THEN (decelerate)
7. IF \((\text{error} < 0)\) AND \((v < 0)\) THEN (decelerate)
8. IF \((\text{error} < 0)\) AND \((v = 0)\) THEN (decelerate)
9. IF \((\text{error} < 0)\) AND \((v > 0)\) THEN (decelerate)

Membership functions must be defined, and to guarantee a safe ascension rate the fuzzy controller output is limited between a range of \(-0.25 \leq v \leq +0.25\). The controller is capable of doing faster velocities, but to ensure operator and technical safety the value is chosen to be within the safe operating range. The membership function is shown in Figure 9-21.

![Membership function plots](image)

Figure 9-21 Membership for input variable "error"
Figure 9-22 Membership function for input variable "u(t)"

Figure 9-23 Membership function for output variable "output 1"
Testing the system with different set depths yields the following results:

![Depth VS Time - position control system](image1)

**Figure 9-24 Depth VS Time – position control system**

![Velocity VS Time - position control system](image2)

**Figure 9-25 Velocity VS Time – position control system**
The system performs very well, with a good rising time and no overshoot. The velocity maintains within the specified range, even for very shallow depths and for depths that are barely hovering over the sea bottom. Oscillation is kept to a minimum, although it is not avoidable to some degree, as of the inertia in the physical system. The fuzzy logic control system, basically acting as the outer control loop, accepts a wide range of input parameters, and the PID is delivering accurate and fast results.
9.3 Position Control – Combined Approach
Part IV

Hardware Design
Chapter 10  ABCS - Electrical design

This chapter describes the actual electrical and mechanical design of the ABCS.

10.1 Complete Module

The system as a whole consists of the following functions:

1) Interfacing of sensors  
2) Implementation of control algorithm through microcontroller  
3) Data logging capability  
4) Provision of an interface for human interaction via LCD  
5) Provision of valve control logic that drives solenoid valves

Figure 10-1  System sketch

Figure 10-1 shows the overall diagram of the system with the main control logic, handling all communication, data acquisition and control is a microcontroller. Interfaced to the microcontroller are several modules. Parts of these single modules in this section will be represented as well as explanation why certain parts were chosen
For the prototype a microcontroller was chosen based upon the popular Atmega2560 and then soldered to a development board. This made the prototyping process easier. The Arduino with the ATmega 2560 was chosen (see table 10-1):

<table>
<thead>
<tr>
<th>Microcontroller</th>
<th>ATmega2560</th>
</tr>
</thead>
<tbody>
<tr>
<td>Operating Voltage</td>
<td>5V</td>
</tr>
<tr>
<td>Input Voltage</td>
<td>7-12V</td>
</tr>
<tr>
<td>Input Voltage (limits)</td>
<td>6-20V</td>
</tr>
<tr>
<td>Digital I/O Pins</td>
<td>54 (15 PWM output)</td>
</tr>
<tr>
<td>Analog Input Pins</td>
<td>16</td>
</tr>
<tr>
<td>DC Current per I/O Pin</td>
<td>40 mA</td>
</tr>
<tr>
<td>DC Current for 3.3V Pin</td>
<td>50 mA</td>
</tr>
<tr>
<td>Flash Memory</td>
<td>256 KB</td>
</tr>
<tr>
<td>SRAM</td>
<td>8 KB</td>
</tr>
<tr>
<td>EEPROM</td>
<td>4 KB</td>
</tr>
<tr>
<td>Clock Speed</td>
<td>16 MHz</td>
</tr>
</tbody>
</table>

Table 10-1: Arduino with ATmega2560 (Arduino Mega Manual)

Arduinos Mega 2560 is a member of the Arduino family of development boards. It is open source, and a very well suited platform for prototyping. The microcontroller onboard is a Atmega 2560 8-Bit processor. One of the advantages of the board and microcontroller is the support of a fairly large number of analog and digital IO’s. This gives the system a significant room for further expansion in the future, without having to change the hardware. The 4 serial ports using UART are another benefit, because one can easily interface additional peripheral hardware without having to use many IO pins. In this case there was a LCD backpack panel attached and a data logging device.

Programs for Arduino are most likely written in the Arduino IDE, but it is also possible to use native C or C++ and rely on AVR libraries. This usually results in better performance, which can be critical for fast response in a control system. It is, however, technically more involved as it requires the developer to be familiar with embedded system concepts such as timers, interrupts, serial communication and bit manipulation. The code for the system was thus written in C++.

The summary of the control system is as follows: The Atmega 2560 acts as a main controller, and it performs all communication, control and logging functions. It thus also handles low-frequency PWM of the valve control logic. To control the buoyancy the system works in cyclic manner. Every timestep involves a series of tasks:

1. Collect input from the pressure sensor
2. Filter readings and convert into values for the control algorithm
3. Apply control algorithm of choice
4. Send output of control algorithm to valve control logic

This logic is independent from user interaction, as it is controlled by interrupts. This way, users can still interact with the system by using the LCD panel and input switches, while the system also controls the ROSCo if so desired. Users can also send manual input to the valves, making it possible to control the ROSCo in the old fashioned manner of inflating/deflating the bags manually. This overrides any automatically generated output to be fail-safe.
10.2 Microcontroller

The system uses a Atmega2560 8-Bit microcontroller, manufactured by AVR (Figure 10-2). The Atmega2560 is based on a RISC architecture. It features 135 instructions, a lot of them single clock cycle executed. It also has plenty of storage: 256Kbytes of flash memory, 4Kbytes EEPROM and 8Kbytes internal SRAM.

10.3 Data acquisition module

The data acquisition module consists of a pressure sensor, measuring the ambient pressure surrounding the ABCS. This reading is then sent into a filter circuit to filter out noise from the power supply supplying the reference voltage and environment noise. The filtered signal then gets passed onto a 12-Bit Analog – Digital Converter. The converted digital signal is then transmitted to the microcontroller via Serial Peripheral Interface. The pressure sensor was chosen to be external as the build-in
10.3 Data acquisition module

10.3.1 Pressure Sensor

The pressure sensor that was chosen is Freescale MPX-5700 –AP. It is an integrated silicon pressure sensor, which is on-chip signal conditioned, temperature compensated and also calibrated. The sensor uses piezoresistive transducing and generates accurate analog output signals. It is of the absolute type, and the maximum Error is 2.5% on a temperature range of 0-85 degree Celsius.

Operating Characteristics

<table>
<thead>
<tr>
<th>Characteristic</th>
<th>Symbol</th>
<th>Min</th>
<th>Typ</th>
<th>Max</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pressure Range(1)</td>
<td>P_{OP}</td>
<td>0</td>
<td>15</td>
<td>700</td>
<td>kPa</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>700</td>
<td>kPa</td>
</tr>
<tr>
<td>Supply Voltage(2)</td>
<td>V_{S}</td>
<td>4.75</td>
<td>5.0</td>
<td>5.25</td>
<td>Vdc</td>
</tr>
<tr>
<td>Supply Current</td>
<td>I_{0}</td>
<td>7.0</td>
<td>10</td>
<td>mA</td>
<td>Adc</td>
</tr>
<tr>
<td>Zero Pressure Offset(3)</td>
<td>V_{off}</td>
<td>0.088</td>
<td>0.184</td>
<td>0.2</td>
<td>0.313</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(0 to 85°C)</td>
<td>(0 to 85°C)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Full Scale Output(4)</td>
<td>V_{FSS}</td>
<td>4.587</td>
<td>4.7</td>
<td>4.813</td>
<td>Vdc</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(0 to 85°C)</td>
<td>(0 to 85°C)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Full Scale Span(5)</td>
<td>V_{FS}</td>
<td>—</td>
<td>4.5</td>
<td>—</td>
<td>Vdc</td>
</tr>
<tr>
<td>Accuracy(6)</td>
<td>%V_{FSS}</td>
<td>—</td>
<td>—</td>
<td>±2.5</td>
<td></td>
</tr>
<tr>
<td>Sensitivity</td>
<td>V/P</td>
<td>6.4</td>
<td>—</td>
<td>mV/KPa</td>
<td></td>
</tr>
<tr>
<td>Response Time(7)</td>
<td>t_{rc}</td>
<td>1.0</td>
<td>—</td>
<td>ms</td>
<td></td>
</tr>
<tr>
<td>Output Source Current at Full Scale Output</td>
<td>I_{O+}</td>
<td>0.1</td>
<td>—</td>
<td>mA</td>
<td>Adc</td>
</tr>
<tr>
<td>Warm-Up Time(8)</td>
<td>—</td>
<td>20</td>
<td>—</td>
<td>ms</td>
<td></td>
</tr>
</tbody>
</table>

1. 1.0 kPa (kiloPascal) equals 0.145 psi.
2. Device is ratio metric within this specified excitation range.
3. Offset (V_{off}) is defined as the output voltage at the minimum rated pressure.
4. Full Scale Output (V_{FSS}) is defined as the output voltage at the maximum or full rated pressure.
5. Full Scale Span (V_{FS}) is defined as the algebraic difference between the output voltage at full rated pressure and the output voltage at the minimum rated pressure.
6. Accuracy (error budget) consists of the following:
   - Linearity: Output deviation from a straight line relationship with pressure over the specified pressure range.
   - Temperature Hysteresis: Output deviation at any temperature within the operating temperature range, after the temperature is cycled to and from the minimum or maximum operating temperature points, with zero differential pressure applied.
   - Pressure Hysteresis: Output deviation at any pressure within the specified range, when this pressure is cycled to and from the minimum or maximum rated pressure, at 25°C.
   - Tc Offset: Output deviation over the temperature range of 0°C to 85°C, relative to 25°C.
   - Tc Offset: Output deviation with minimum rated pressure applied, over the temperature range of 0°C to 85°C, relative to 25°C.
   - Variation from Nominal: The variation from nominal values, for Offset or Full Scale Span, as a percent of V_{FSS} at 25°C.
7. Response Time is defined as the time for the incremental change in the output to go from 10% to 90% of its final value when subjected to a specified step change in pressure.
8. Warm-Up Time is defined as the time required for the device to meet the specified output voltage after the pressure has been stabilized.

Table 10-3 Operating characteristics according to manual of MPX-5700-AP

As shown on table 10-3 the maximum pressure is 700000 Pa, which corresponds to a depth of 59m. This is not enough for going to depths of 100m, but as this is a prototype it suffices. A later version of the system can easily use a more expensive pressure sensor with higher maximal operating pressure, as the system offers enough room for expansion.
10.3 Data acquisition module

10.3.2 Filtering and Analog to Digital Conversion Circuit

The system uses a MCP-3201-C 12-Bit Analog-Digital-Converter (ADC) (Figure 10-5). This results in more accurate readings compared to the build-in ADC of the Atmega2560, which only provides 10-Bit of resolution.

The MPC-3201-C is a successive approximation ADC, that features on-board sampling and hold circuitry. Communication is done by a simple serial interface that is compatible with SPI protocol. Sampling rates of 100000 samples/s at a clock rate of 1.6MHz are possible. The device can be driven by a voltage ranging from 2.7-5.5V and uses 300microA operating current.
Figure 10-6 Characteristics of MCP-3201-C

The pressure sensor output is filtered before it is passed onto the MCP-3201-C, this is done by a simple RC circuit that is shown on Figure 10-6, using a 0.2kΩ resistor and a 20pF capacitor.
10.3 Data acquisition module

**10.3.3 Switch Array**

In order to process user input into the system, a switch array has been designed. The system uses simple momentary switches, that can consist of push-buttons or REED switches. Most switches, as momentary switches and reed switches, are made of springy metals. Thus every time a user activates the switch, instead of making steady contact upon activation they rather bounce apart multiple times when activated. This is due to their elasticity and momentum. Figure 10-7 illustrates a typical bouncing period before a signal settles to a HIGH value.

![Figure 10-7 Filtering circuit](maximintegrated.com)

While this is not an important factor in power circuits, it can, however, cause problems when used in a digital logic. Especially if the circuit responds with a sufficient high sample rate, as in a microcontroller's IO pin that has been set-up for registering rising/falling flanks (Walker 1988). As the microcontroller has most likely an interrupt enabled for that pin on a HIGH flank, what happens is that the user activates the switch once but the interrupt gets triggered multiple times. This leads to unresponsive response, or even malfunction, if the input is being counted upon activation etc. Multiple methods exist for debouncing, from simple software solutions to low-pass hardware filtering. To minimize software overhead for debouncing, a simple RC low-pass filtering circuit was developed to debounce the signal of the switches. A simple low-pass filter circuit usually consists of a resistor that is in series with a load, and a capacitor in parallel with the load (UNSW RC filters):
If applying a noisy signal to this circuit, the capacitor blocks low-frequency signals, as off the exhibited reactance. Thus they are forced into $V_{out}$. For higher frequencies the reactance is smaller, allowing them to pass through the capacitor as it basically acts as a short circuit. One can then apply electromagnetic laws to find the cut-off frequency, which is the boundary in the frequency at which the system starts reducing the energy of the signal.

$$f_c = \frac{1}{2\pi \tau} = \frac{1}{2\pi RC}$$

R = resistance
C = capacitance

Applying this filtering circuit to the switch output yields an effective debouncing. It was also combined with an additional resistor added to the circuit. This second resistor in series with the load allows the capacitor to discharge slowly if a sudden discharge happens and also acts as a current limiter for the switches’ contacts (Ganssle 2004).
When the switches are closed, Vout becomes zero, this input is then being forwarded to a hex Schmitt trigger, which is an IC consisting of six Schmitt triggers. The Schmitt trigger is a comparator circuit with hysteresis. This means the system is not only depending on its current state but also on past states. The Schmitt trigger acts as an active circuit that converts analog inputs into digital output signals. It has a threshold, thus for an inverting Schmitt trigger like the 74HCT19N, the Schmitt trigger outputs a logical HIGH of 5V if the input voltage drops under a threshold value. This makes the 74HCT19N ideal for the debouncing of the switch signal.

10.4 Valve Control Circuit

In order to control the valves a simple valve control module has been developed. The ABCS uses a simple and inexpensive solenoid valves with a fairly quick reaction time, the Sizto Tech Corp. solenoid valve 2P025. The system relies on designating valves for either inflating or deflating. The basic prototype therefore consists of one valve for inflating and
one valve for deflating the ballast tanks. As of the minor costs of one valve (of 20$), the system can be expanded to use multiple valves as well.

To drive the valve 12V DC are used. The opening/closing of the valve will be controlled by the use of a transistor circuit. The transistor in use is the reliable and inexpensive TIP120 a NPN transistor (Table 10-11).

Absolute Maximum Ratings*  *T_a = 25°C unless otherwise noted

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Parameter</th>
<th>Ratings</th>
<th>Units</th>
</tr>
</thead>
<tbody>
<tr>
<td>V_{CE0}</td>
<td>Collector-Base Voltage : TIP120</td>
<td>60</td>
<td>V</td>
</tr>
<tr>
<td></td>
<td>: TIP121</td>
<td>80</td>
<td>V</td>
</tr>
<tr>
<td></td>
<td>: TIP122</td>
<td>100</td>
<td>V</td>
</tr>
<tr>
<td>V_{CEO}</td>
<td>Collector-Emitter Voltage : TIP120</td>
<td>60</td>
<td>V</td>
</tr>
<tr>
<td></td>
<td>: TIP121</td>
<td>80</td>
<td>V</td>
</tr>
<tr>
<td></td>
<td>: TIP122</td>
<td>100</td>
<td>V</td>
</tr>
<tr>
<td>V_{FAB}</td>
<td>Emitter-Base Voltage</td>
<td>5</td>
<td>V</td>
</tr>
<tr>
<td>I_C</td>
<td>Collector Current (DC)</td>
<td>5</td>
<td>A</td>
</tr>
<tr>
<td>I_{CP}</td>
<td>Collector Current (Pulse)</td>
<td>5</td>
<td>A</td>
</tr>
<tr>
<td>I_B</td>
<td>Base Current (DC)</td>
<td>120</td>
<td>mA</td>
</tr>
<tr>
<td>P_C</td>
<td>Collector Dissipation (T_a=25°C)</td>
<td>2</td>
<td>W</td>
</tr>
<tr>
<td></td>
<td>Collector Dissipation (T_a=25°C)</td>
<td>65</td>
<td>W</td>
</tr>
<tr>
<td>T_J</td>
<td>Junction Temperature</td>
<td>150</td>
<td>°C</td>
</tr>
<tr>
<td>T_{STG}</td>
<td>Storage Temperature</td>
<td>-65 to 150</td>
<td>°C</td>
</tr>
</tbody>
</table>
### Electrical Characteristics

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Parameter</th>
<th>Test Condition</th>
<th>Min.</th>
<th>Typ.</th>
<th>Max.</th>
<th>Units</th>
</tr>
</thead>
<tbody>
<tr>
<td>$V_{CEO(sus)}$</td>
<td>Collector-Emitter Sustaining Voltage</td>
<td>$I_C = 100mA$, $I_B = 0$</td>
<td>60</td>
<td>80</td>
<td>100</td>
<td>V</td>
</tr>
<tr>
<td>$I_{CEO}$</td>
<td>Collector Cut-off Current</td>
<td>$V_{CE} = 30V$, $I_B = 0$</td>
<td>0.5</td>
<td>mA</td>
<td></td>
<td>V</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$V_{CE} = 40V$, $I_B = 0$</td>
<td>0.5</td>
<td>mA</td>
<td></td>
<td>V</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$V_{CE} = 50V$, $I_B = 0$</td>
<td>0.5</td>
<td>mA</td>
<td></td>
<td>V</td>
</tr>
<tr>
<td>$I_{CBO}$</td>
<td>Collector Cut-off Current</td>
<td>$V_{CB} = 60V$, $I_E = 0$</td>
<td>0.2</td>
<td>mA</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>$V_{CB} = 80V$, $I_E = 0$</td>
<td>0.2</td>
<td>mA</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>$V_{CB} = 100V$, $I_E = 0$</td>
<td>0.2</td>
<td>mA</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$I_{CEO}$</td>
<td>Emitter Cut-off Current</td>
<td>$V_{EB} = 5V$, $I_E = 0$</td>
<td>2</td>
<td>mA</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$h_{FE}$</td>
<td>* DC Current Gain</td>
<td>$V_{CE} = 3V$, $I_C = 0.5A$</td>
<td>1000</td>
<td></td>
<td>1000</td>
<td>V</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$V_{CE} = 3V$, $I_C = 3A$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$V_{CE(sat)}$</td>
<td>* Collector-Emitter Saturation Voltage</td>
<td>$I_C = 3A$, $I_B = 12mA$</td>
<td>2.0</td>
<td>V</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>$I_C = 5A$, $I_B = 20mA$</td>
<td>4.0</td>
<td>V</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$V_{BE(on)}$</td>
<td>* Base-Emitter On Voltage</td>
<td>$V_{CE} = 3V$, $I_C = 3A$</td>
<td>2.5</td>
<td>V</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$C_{ob}$</td>
<td>Output Capacitance</td>
<td>$V_{CB} = 3V$, $I_E = 0$, $f = 0.1MHz$</td>
<td>200</td>
<td>pF</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

* Pulse Test: Pulse Width <300μs, Duty Cycle <2%

Table 10-11 TIP 120, characteristics and maximum ratings according to datasheet.

In order to drive the valves the following circuit was built:

![Figure 10-12 Valve controlling circuit](image-url)
JP1 and JP2 are sockets for the valves, allowing to use any solenoid valve. When the solenoids are at work (i.e., current is flowing “down” from the positive terminal down to the ground), the transistor acts like a closed switch. Solenoids create a magnetic field, so the system sends a close signal to the valves, causing the transistor to act as an opened switch, which is accompanied by a sudden drop of current at the inductors, and causing them to use their stored magnetic field to create their own voltage. This leads to a high negative potential. Since the transistor is still at the voltage of the power supply and open this leads to a large potential difference between the collector and emitter gate of the transistor, which can damage the transistor, and allows electrons to jump over across the junction. In order to avoid this rectifier diodes of the 1n4004 type, have been inserted into the system as well. These solve as a so called fly back diode, protecting the transistors. When the transistors are now acting as opened switches, the rectifier diodes allow the inducing solenoids to draw current from themselves in a repeating cycle until the energy is used up. This is happening at some point in time due to energy loss in the wiring. When the transistor acts as a closed switch the rectifier diodes basically are not visible as they do not allow any current to pass through in that direction.

The base of the transistors has two input lines: one from the microcontroller, allowing for control techniques to drive the valves with a low frequency PWM, and one for manual control via a momentary button.
10.5 LCD Panel

A low-cost, low power consumption LCD panel is used for user input into the system. To save IO Pins, a serial backpack is used (Figure 10-13). The backpack is soldered to the LCD Panel, and can be attached to the system with only 3 pins in use: VCC, GND and TXn. The backpack itself relies on serial communication via UART and uses an Attiny2313a microcontroller on a board. It presents an inexpensive and simple communication method while saving IO pins on the main microcontroller.

Figure 10-13 LCD serial backpack with HD44780 controlled LCD (Jaycon)

10.6 Power Supply circuit

The system is powered by a 12V DC battery pack of eight AA batteries. To drive digital logic components and ICs the voltage will be regulated to 5V DC. The solenoid valves are driven by the full 12V DC (Figure 10-19).
The battery pack will be attached to the board with a barrel jack connector. A LM7805 voltage regulator converts the voltage from 12V DC to 5V DC. Capacitors of 1 microfarad are added to ensure a noise free voltage supply.
10.7 Serial Communications

A simple serial communications interface was built to convert the built in UART serial signals, also referred to as TTL (transistor-transistor logic), which uses voltages between 0V and Vcc (5V) to a signal complying with RS-232. A ICL3232 chip with a standard DB9 connector was used for this purpose (Figure 10-15).

![Figure 10-15 Serial communication interface](image)

10.8 Power consumption

The ABCS is meant to be operating underwater and the system is based on an autonomous power supply. Hence an approximation of the systems power usage must be calculated. The power consumption is based on the average ascent to surface at a safe velocity from 100m of depth. As the tests in chapter 9 shows, this takes about 10 minutes at a safe rate of 0.2 m/s. First, the systems current consumption are approximated. Assuming peak currents whenever possible to yield a result for the worst possible battery runtime, table 10-15a shows the
current consumption and running time with a 8xAA 12V DC battery pack of the ABCS, continuous runtime.

<table>
<thead>
<tr>
<th>Component</th>
<th>Quantity</th>
<th>current[mA]</th>
<th>Voltage[V]</th>
<th>Power Consumption [Watt]</th>
</tr>
</thead>
<tbody>
<tr>
<td>ATmega2560</td>
<td>1</td>
<td>500</td>
<td>5</td>
<td>2.5</td>
</tr>
<tr>
<td>Solenoid Valve 12V DC &quot;STC 2p025&quot;</td>
<td>2</td>
<td>500</td>
<td>12</td>
<td>12</td>
</tr>
<tr>
<td>Pressure Sensor MPX 5700 AP</td>
<td>1</td>
<td>10</td>
<td>5</td>
<td>0.05</td>
</tr>
<tr>
<td>ADC MCP 3201 C</td>
<td>1</td>
<td>0.15</td>
<td>5</td>
<td>8E-04</td>
</tr>
<tr>
<td>Hex Schmitt trigger</td>
<td>1</td>
<td>0.01</td>
<td>5</td>
<td>5E-05</td>
</tr>
<tr>
<td>TIP 120 NPN Transistor</td>
<td>2</td>
<td>120</td>
<td>5</td>
<td>0.6</td>
</tr>
<tr>
<td>HD44780 LCD panel</td>
<td>1</td>
<td>3</td>
<td>5</td>
<td>0.015</td>
</tr>
<tr>
<td>ATtiny2313a</td>
<td>1</td>
<td>12</td>
<td>5</td>
<td>0.06</td>
</tr>
<tr>
<td>Open log serial data logger</td>
<td>1</td>
<td>6</td>
<td>5</td>
<td>0.03</td>
</tr>
<tr>
<td>Overall</td>
<td></td>
<td>1151.16</td>
<td></td>
<td>15.26</td>
</tr>
<tr>
<td>Running time</td>
<td></td>
<td>1.73737795 h</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8xAA Battery pack 12V DC</td>
<td>2000</td>
<td>mAh</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 10-16a Power consumption of ABCS

When continuously driven on full load, the system has enough power to operate for 1.73 hours, which is equal to 102 minutes under full load. With the said 10 minutes per ascent, this gives 10 ascent operations until the batteries have to be changed. This can easily be extended by installing another battery pack into the housing, or supplying voltage from the crawlers main power supply.
10.9 PCB Design of schematics

In order to facilitate further research, a Printed Circuit Board has been developed. A PCB offers several advantages:

1. They use less space than single-wire/breadboarded prototypes
2. They are easy to mass produce
3. They are more robust usually
4. They are more reliable (can be coated to resist corrosion etc.)

It thus made sense after a prototype was soldered to develop and layout an all-in-one PCB.

The finished boards layout can be fitted on a relatively small size (Figures 10-17, 10-18, 10-19):

![Figure 10-17 Layout for PCB](image-url)
Figure 10-18 Edged board, top view

Figure 10-19 Edged board, bottom view
10.10 ABCS – Mechanical design

10.10.1 Introduction to the mechanical system

Having now explained the electrical design, and the instrumentation of the ABCS this section now illustrates the details of the mechanical design. The general idea of the mechanical system is rather simple: Connect two solenoid valves to a T-port, and connect the T-port to the ballast tank. Then connect the solenoid valves to the electrical system. Finally, attach a low pressure hose from a scuba tank to one of the solenoid valves. The ABCS will be placed inside a housing. Inside the housing the mechanical and electrical components are arranged in a modular and space saving manner. As one of the requirements was to build the system with readily available, inexpensive off-the-shelf parts, the piping and connections are all hardware-store grade, thus they come with all the typical disadvantages these parts have. This is mainly a problem when fitting them together, as obviously a custom engineered connection system for the tanks and valves would have been more efficient. The same applies to the housing, a commercial dry-box was chosen. It was then modified with tools and sealed afterwards, to fit all the components inside, and make it ready for connection to a scuba regulators first stage low-pressure hose and ballast tank.

10.10.2 Valves

As already described the valves used for the prototype where STC 2P025 solenoid valves. These valves are inexpensive, and can be driven with 12V DC current. They produce up to 22 cubicfoot/minute flowrates @ 100 psi pressure, and have a reaction time of under 20ms, which makes them still feasible for a lowfrequency PWM (which will be employed in the implementation of the valve control on the Arduino Mega):

22 SCFM @100 PSI
Pressure: Vacuum to 115 PSI
Table 10-20 Valve specifications according to manual

<table>
<thead>
<tr>
<th>Valve Model</th>
<th>2P025</th>
</tr>
</thead>
<tbody>
<tr>
<td>Valve Type</td>
<td>2 Way, Normally Closed (NC)</td>
</tr>
<tr>
<td>Action</td>
<td>Direct Acting (Poppet), Response Time &lt;20msec</td>
</tr>
<tr>
<td>Cv (Orifice)</td>
<td>0.23 (2 mm)</td>
</tr>
<tr>
<td>Operating Pressure</td>
<td>0 to 115 PSI</td>
</tr>
<tr>
<td>Operating Temperature</td>
<td>14-122°F (-20 to 50°C)</td>
</tr>
<tr>
<td>Port Size (NPT)</td>
<td>1/8” to 1/4”</td>
</tr>
<tr>
<td>Body Materials</td>
<td>Engineered Nylon</td>
</tr>
<tr>
<td>Seal Materials</td>
<td>NBR (Buna)</td>
</tr>
<tr>
<td>Coil Power</td>
<td>3W-BW (Pressure Dependent)</td>
</tr>
<tr>
<td>Coil Duty</td>
<td>100% ED</td>
</tr>
<tr>
<td>Voltage/Coil (Power)</td>
<td>± 10% of Specified Voltage</td>
</tr>
<tr>
<td>Service</td>
<td>Air, Liquid, Water</td>
</tr>
</tbody>
</table>

The solenoid valve proved to be reliable during the course of testing and prototyping (Table 10-20). However, no longterm fatigue testing tests were done. Solenoid valves have a finite lifespan, thus if cycled too often, they will degrade, since the system is designed to be modular, changing the valves is relatively inexpensive, and if there is a need for faster valves, they can quickly and easily be replaced.
10.10.3  Connections and Piping

Standard brass connectors and threaded fittings were used to assemble the T-port. A schematic of the system is shown in figure 10-20. One can see that the system uses a commercial grade scuba tank and first stage, as well as a low pressure hose. The T-port with the attached solenoid valves and all the outlets are placed inside a housing and sealed. The system is plugged into the control/instrumentation system and is ready for use.

Figure 10-20 Schematic of mechanical design of the ABCS
Part V

Conclusion
Chapter 11

Conclusions and future research

11.1 Conclusion

One of the main goals of work on the ABCS was to design and develop a basic setup for the evaluation of the potential of such a device. The control system designed as part of this thesis work, demonstrated the theoretical capabilities of these approaches.

The designed mechanical and electrical system has all the components required for an effective evaluation in real world tests. The selection of open-architecture based control components facilitates any future work regarding the control system design. The selected controller Atmega2560 (used on the Arduino) also is supported in Matlab and Simulink, which can be helpful for modeling and simulation. Off-the-shelf instrumentation helps in reliable operation and maintenance. External connectivity and Human Interfacing such as LCD panel and Switches facilitate the task of operating the device, and save time when field testing.

The mathematical work shows that a control approach is possible, despite the non-linear nature of the system. The numerical simulation also proves that a position control of the ROSCo is possible, and not to complicated to implement by combining the already implemented PID with a fuzzy logic controller.

To conclude, the ABCS was able to fulfill its design objectives. The numerical model successfully demonstrated the possibility of such a system. The system is now ready for real world testing.

This work in this thesis presents the design and development for a prototype ABCS. It was built to demonstrate a basic working design. There are many places where this system can be improved and design can continue. Some suggestions will be presented regarding future work in this section.

11.2 Simulation and Modeling

The simulation model is fairly advanced, as it does calculate time based deflation rates and approximates the non-linear response of the dynamical system fairly reasonable. But this
does not mean it is perfect, as it does not account for noise and perturbances so far. The next step for model of the ABCS and the ROSCo should be the development of an enhanced simulation model that contains all parts needed random noise and disturbance that might affect the system. For results close to the real world system, this model would also need to include changes in horizontal and vertical current. The fundament of the system is very structured, it should thus be possible without to much hassle to implement those features. A good simulation model can be very helpful in good and efficient and fast system design, as it also implies that the physical system has been understand fully

11.3 Mechanical system

As of now the mechanical system consists of inexpensive parts. A future prototype might include more expensive and faster solenoid valves, as well as a custom made housing with fitted connectors and piping.

11.4 Electrical System

The system build was a prototype board, that consists of three single boards. The contents of the boards represent the schematic presented in the appendix. However, a layout for a PCB has been designed in Chapter 10. One further step could be the production of several of these prototypes to have more than one ABCS.

11.5 Control Methods

One last topic for future research might be the design and implementation of an actual fuzzy logic system on the microcontroller. As of now, the system just uses a PID, as this is a stable technique. In order to implement a position control, this PID might be controlled by a fuzzy logic controller. All the rules and parameters to yield a working system have been identified, and tested on the model simulation in this thesis(refer to chapter 8 and 9). The next step would be to implement them on an external system and attach it to the ABCS, which already has the hardware capability of processing external inputs.
References


Nybakken, Webster, Life in the Ocean, 1998


Norman S. Nise , Control Systems Engineering , 6th, (Dec 14, 2010)

Predavanja, Intelligent Control Techniques in Mechatronics

J.Ganssle, A Guide to Debouncing, 2004

RC filters, integrators and differentiators , UNSW, Australia


http://www.seattlerobotics.org/encoder/mar98/fuz/flindex.html

http://eprints.gla.ac.uk/3817/1/IEEE3.pdf

Ogata, Modern Control Engineering , 5th


http://www.control-systems-principles.co.uk/whitepapers/frequency-response-analysis1.pdf
References


Bard, Jochen, and Juergen Schmid. 2005. Crannogs, downloaded on 04/24/2014 from The scottish crannog center (http://www.crannog.co.uk/)


Appendix 1: Microcontroller Code

A1.1 Main Program

```c
/* DEFINES AND INCLUDES */

#define F_CPU   16000000UL
#include <avr/io.h>
#include <util/delay.h>
#include <avr/interrupt.h>
#include <stdlib.h>
#include <stdio.h>
#include "spi.h"
#include "lcdshortcuts.h"
#include "uart.h"

#define SELECT_ADC PORTB &= ~(1<<PB4)
#define DESELECT_ADC PORTB |= (1<<PB4)

#define P_GAIN 0.5
#define I_GAIN 0.003
#define D_GAIN 0.01

int showVoltageAndPressureonLCD(float voltage,float pressure);
int showButtonHits();
int initialize();
int showMenu();
int showTimerHits();

unsigned short read_adc(void);

int cliFlag = 0;
int extraTime = 0;
int timer1Counter = 0;
int dutyCyclePortB6 = 20;
int dutyCyclePortB5 = 20;
int counter = 0;

int filterCounter = 0;
int seconds = 0;

unsigned int adcvalue;
```
uint8_t menuState = 0x42;
uint8_t controllerMode = 0;  // 0 = manual, 1 = PID
uint8_t menuStateLowerFourBits = 0x00;
uint8_t controllerState = 0x00;  // 0x00 means idle
uint8_t nextStateByButton1 = 0x00;
uint8_t nextStateByButton2 = 0x01;
uint8_t howManyMenuOptions = 0x04;

float voltageRead;
float currentPressure;
float previousPressure;
float realPressure;

float pressureValues[10];
float currentVelocity;
float previousVelocity;
float currentDepth;
float previousDepth;
float gravity = 9.8;
float density = 1025.0;

float error;
float previous_error;
float setVelocity = 0.20;
float I_error;
float D_error;
float output;
float valveInput;

int main(void)
{
    initialize();  // initialize SPI, interrupts etc

int add = +1;
while(1)
{
    showMenu();
    // showButtonHits()
    // showMenu();
    // showVoltageAndPressureOnLCD(voltageRead, currentPressure);
    // dutyCyclePortB5 = dutyCyclePortB5 + add;
    counter++;
    if(counter >= 100)
    {
        add = add * (-1);
        counter = 0;
    }
    _delay_ms(100);
}
}
```c
int initialize()
{
    /*----------------- SETUP VALVECONTROL PINS AND TIMERS FOR PWM */
    DDRB |= (1<<PB6);
    DDRB |= (1<<PB6);

    /*---------------- SETUP PRESSURE SENSOR READING & LCD PANEL */
    /* chip select for ADC */
    DDRB |= (1<<PB4);
    /* initial select - clear the bit on PB4 to activate it */
    SELECT_ADC;
    /* Enable SPI */
    setup_spi(SPI_MODE_0, SPI_MSB, SPI_NO_INTERRUPT, SPI_MSTR_CLK16);
    /* Enable Serial Communication */
    USART_Init(MYUBRR);

    /*----------------- SETUP BUTTON SHIELD */
    PORTD |= (1 << PORTD1);
    EIMSK |= (1 << INT1);
    /* set to rising edge */
    EICRA |= (1 << ISC11) | (1 << ISC10);
    /*PD1 is now an input with pull-up enabled */
    PORTD |= (1 << PORTD2);
    EIMSK |= (1 << INT2);
    /* set to rising edge */
    EICRA |= (1 << ISC21) | (1 << ISC20);
    /*PD2 is now an input with pull-up enabled */

    sei(); // enable interrupts

    TCCR0A = (1<<WGM01); // set CTC mode on Timer : clear timer on compare
}```
OCR0A = 250; // Compare and match to 250 Ticks which means every 1ms there will be an interrupt
TIMSK0 = (1<<OCIE0A); // Enable the Timer-Counter/Compare Match - Interrupt
//TCCR0B = (1<<CS02) | (1<<CS00); // set PreScale to 1024
TCCR0B = (1<<CS01) | (1<<CS00); // set PreScale to 64

TCCR1B = (1 << WGM12);
OCR1A = 25000;
TIMSK1 = (1 << OCIE1A);
TCCR1B |= (1 << CS11) | (1 << CS10);
return 0;
}

int showVoltageAndPressureOnLCD(float voltage, float pressure)
{
  LCD_CLEAR
  char buffer[21];
sprintf(buffer, "%06.3f", pressure);
SendString("P: ");
SendString(buffer);
SendString(" [kPa]");
LCD_2NDLINE
SendString("V: ");
sprintf(buffer, "%01.3f", voltage);
SendString(buffer);
SendString(" [V]");
}

int showButtonHits()
{
  LCD_CLEAR
char buffer[21];
sprintf(buffer, "0x%02x", menuStateLowerFourBits);
SendString("menuState: ");
SendString(buffer);
/*
SendString(" clicks");
itoa(cliFlag,str,10);
SendString(str);*/
return 0;
}
}
Appendix 1: Microcontroller Code

```c
int showTimerHits()
{
    char str[21];
    LCD_CLEAR
    SendString("dutyCycle ");
    itoa(dutyCyclePortB5,str,10);
    SendString(str);
    return 0;
}

int showMenu()
{
    char buffer[21];
    sprintf(buffer, "0x%02x",menuState);
    switch (menuState)
    {
    case 0x22:
    {
        nextStateByButton2 = 0x22;
        nextStateByButton1 = 0x21;
        LCD_CLEAR
        SendString("PID MODE");
        LCD_2NDLINE
        SendString("Stop Ascent   >");
    }
    break;
    case 0x21 :
    {
        nextStateByButton2 = 0x20;
        nextStateByButton1 = 0x22;
        LCD_CLEAR
        SendString("PID MODE");
        LCD_2NDLINE
        SendString("Start Ascent   >");
    }
    break;
    case 0x20 :
    {
        nextStateByButton2 = 0x21;
        nextStateByButton1 = 0x02;
        LCD_CLEAR
        SendString("PID MODE");
        LCD_2NDLINE
        SendString("Back to Menu   >");
    }
    break;
    case 0x11 :
    {
        nextStateByButton2 = 0x10;
        nextStateByButton1 = 0x11;
```
Appendix 1: Microcontroller Code

```
247.       LCD_CLEAR
248.       SendString("MANUAL MODE");
249.       LCD_2NDLINE
250.       char buf[21];
251.       sprintf(buf, "%06.1f", currentPressure);
252.       SendString("P: ");
253.       SendString(buf);
254.       SendString(" ");
255.       SendString("V:");
256.       sprintf(buf, "%01.2f", currentVelocity);
257.       SendString(buf);
258.       break;
259.   case 0x10 :
260.       {  
261.       nextStateByButton2 = 0x11;
262.       nextStateByButton1 = 0x01;
263.       LCD_CLEAR
264.       SendString("MANUAL MODE");
265.       LCD_2NDLINE
266.       SendString("Back to Menu >");
267.       break;
268.   case 0x42 :
269.       {  
270.       nextStateByButton2 = 0x40;
271.       nextStateByButton1 = 0x42;
272.       LCD_CLEAR
273.       SendString(" output ");
274.       char buf[21];
275.       sprintf(buf, "%06.2f", valveInput);
276.       SendString(buf);
277.       LCD_2NDLINE
278.       SendString("In= ");
279.       itoa(dutyCyclePortB5,buf,10);
280.       SendString(buf);
281.       SendString("% ");
282.       SendString("Out= ");
283.       itoa(dutyCyclePortB6,buf,10);
284.       SendString(buf);
285.       break;
286.   case 0x41 :
287.       {  
288.       nextStateByButton2 = 0x42;
289.       break;
290.   case 0x41 :
291.       {  
292.       nextStateByButton2 = 0x42;
```
Appendix 1: Microcontroller Code

```c
nextStateByButton1 = 0x41;
LCD_CLEAR
char buf[21];
sprintf(buf, "%06.1f", currentPressure);
SendString("P:");
SendString(buf);
SendString(["kPa"]);
LCD_2NDLINE
SendString("V:");
sprintf(buf, "%01.1f", voltageRead);
SendString(buf);
SendString(["V"]);
SendString(">");
}
break;
case 0x40 :
{
nextStateByButton2 = 0x41;
nexstateByButton1 = 0x00;
LCD_CLEAR
SendString("SENSOR DISPLAY");
LCD_2NDLINE
SendString("Back to Menu ");
break;
}
case 0x04 :
{
nextStateByButton2 = 0x00;
nexstateByButton1 = 0x40;
LCD_CLEAR
SendString("Main Menu ");
LCD_2NDLINE
SendString("DISPLAY SENSORS ");
break;
}
case 0x03 :
{
nextStateByButton2 = 0x04;
nexstateByButton1 = 0x30;
LCD_CLEAR
SendString("Main Menu ");
LCD_2NDLINE
SendString("FUZZY MODE ");
break;
}
case 0x02 :
{
nextStateByButton2 = 0x03;
nexstateByButton1 = 0x20;
LCD_CLEAR
SendString("Main Menu ");
LCD_2NDLINE
```
Appendix 1: Microcontroller Code

355.    SendString("PID MODE >");
356. }
357.    break;
358.    case 0x01 :
359.    {
360.        nextStateByButton2 = 0x02;
361.        nextStateByButton1 = 0x10;
362.        LCD_CLEAR
363.        SendString("Main Menu ");
364.        LCD_2NDLINE
365.        SendString("MANUAL MODE >");
366.    }
367.    break;
368.    case 0x00 :
369.    {
370.        nextStateByButton2 = 0x01;
371.        nextStateByButton1 = 0x00;
372.        LCD_CLEAR
373.        SendString("Main Menu ");
374.        LCD_2NDLINE
375.        SendString("CHOOSE MODE >");
376.    }
377.    break;
378.    default:
379.    {
380.        // error codes that require no additional action
381.    }
382.    break;
383.    }
384.    }
385.    unsigned short read_adc(void)
386.    {
387.        /*select ADC wait 100 microseconds then read two bytes */
388.        SELECT_ADC;
389.        _delay_us(100);
390.        unsigned char one = send_spi(0xFF);
391.        _delay_us(100);
392.        unsigned char two = send_spi(0xFF);
393.        DESELECT_ADC;
394.        /*12 bits of ADC value is bottom 5 bits of first
395.        byte and top 7 bits of second, move into 16 bit int */
396.        return ((0x1F & one) << 7) | (two >> 1);
397.    }
398.    ISR (INT1_vect)
399.    {
400.    }
Appendix 1: Microcontroller Code

```c
408.     //menuState = nextStateByButton1;
409. }
410.
411. ISR (INT2_vect)
412. {
413.     //menuState = nextStateByButton2;
414. }
415.
416. ISR ( TIMER0_COMPA_vect)
417. {
418.     extraTime++;
419.
420.     if(extraTime >=dutyCyclePortB6)
421.         {
422.             PORTB &= ~(1<<PORTB6);
423.         }
424.     if( extraTime < dutyCyclePortB6)
425.         {
426.             PORTB |= (1 << PORTB6);
427.         }
428.     if(extraTime >= dutyCyclePortB5)
429.         {
430.             PORTB &= ~(1<<PORTB5);
431.         }
432.     if( extraTime < dutyCyclePortB5)
433.         {
434.             PORTB |= (1 << PORTB5);
435.         }
436.
437.     if(extraTime >= 100)
438.         {
439.             extraTime = 0;
440.         }
441.
442. }
443.
444. ISR (TIMER1_COMPA_vect)
445. {
446.     previousPressure = realPressure; // save previous value
447.
448.     /*read the ADC input */
449.     adcvalue = read_adc();
450.     voltageRead = ((float)adcvalue / 4093) * 5.0;
451.     currentPressure = (( voltageRead/5.0) - 0.04)/0.0012858;
452.     currentPressure = currentPressure * 1000;
453.
454.     /* simple software LOW-
455.     PASS filter to minimize high frequency noise on pressure readings */
456.     float actualPressure;
457.     pressureValues[filterCounter] = currentPressure;
458.     filterCounter++;
459.     if(filterCounter >= 10)
460.         {
```
filterCounter = 0;

for(int i = 0; i < 10; i++)
{
    actualPressure = actualPressure + pressureValues[i];
}
realPressure = actualPressure/10.0;

/* calculate velocity from pressure differential*/

previousVelocity = currentVelocity;
currentVelocity = (previousPressure - realPressure) / (gravity * density);
currentVelocity = currentVelocity / 0.1;

error = setVelocity - currentVelocity;
I_error += (error);
D_error = (error - previous_error);

output = (P_GAIN * error) + (I_GAIN * I_error) + (D_GAIN * D_error);
previous_error = error;

if(output > 1.0)
{
    valveInput = 1.0;
}
else if(output < -1.0)
{
    valveInput = -1.0;
}
else
{
    valveInput = output;
}

if(valveInput > 0)
{
    dutyCyclePortB5 = valveInput * 100;
dutyCyclePortB6 = 0;
}
else if (valveInput < 0)
{
    dutyCyclePortB5 = 0;
dutyCyclePortB6 = valveInput * 100;
}
else
{
    dutyCyclePortB5 = 0;
dutyCyclePortB6 = 0;
}
A1.2 Serial Communication

```
1. #ifndef _uart_h_
2. #define _uart_h_
3. #include <avr/io.h>
4. #ifdef __cplusplus
5. extern "C"{
6. #endif
7. #include "uart.h"
8. #define F_CPU   16000000UL
9. #include <avr/io.h>
10. /* SETUP UART */
11. void USART_Init( unsigned int ubrr)
12. {
13. /*Set baud rate */
14.   UBRR0H = (unsigned char)(ubrr>>8);
15.   UBRR0L = (unsigned char)ubrr;
16. /*Enable receiver and transmitter */
17.   UCSR0B = (1<<RXEN0)|(1<<TXEN0);
18. /* Set frame format: 8data, 2stop bit */
19.   UCSR0C = (1<<USBS0)|(3<<UCSZ00);
20. }
21. /* Simple methods to make UART read and transmit more readable - Extremely unnecessary*/
22. void USART_Transmit( unsigned char data )
23. {
24.   while(!(UCSR0A & (1<<UDRE0)));
25.   UDR0 = data;
26. }
27. unsigned char USART_Receive( void )
28. {
29.   #endif
30. #endif
```
26.      return UDR0;
27. }
28. 
29.      void SendString(char *StringPtr)
30. {    
31.        while(*StringPtr != 0x00)
32.        {  
33.            USART_Transmit(*StringPtr);
34.            StringPtr++;
35.        }  
36.    }
Appendix 2: Schematics

Figure 11-1  Schematics of control system
Figure 11-2  Schematics of Microcontroller pins

Figure 11-3 Schematics of microcontroller pins
Figure 11-4 Schematics of power input

Figure 11-5 Schematics of switch array
Figure 11-6 Schematics of switch array

Figure 11-7 Schematics of valve control logic
Figure 11-8 Schematics of filtering and sensor logic

Figure 11-9 Schematics of programmer connector
Figure 11-10 Serial communication interface
Appendix 3: Printed Circuit Board

Figure 11-11 PCB layout
Figure 11-12 Layout of Power connector

Figure 11-13 Layout of switch array
Figure 11-14 Layout around the microcontroller

Figure 11-15 Layout of sensors and A/D converter
Figure 11-16 Layout of valve control logic
Appendix 4: Graphical representation of Simulink Models
Figure 11-17 Simulink: physical simulation model
Figure 11-18 Simulink: physical simulation model of valve
Figure 11-19 PID design in Simulink
Figure 11-20 Position control design in Simulink
Appendix 5: Building process photos

Figure 11-21 Instrumentation board, early stage
Figure 11-22 Switch shield and instrumentation board, early stage

Figure 11-23 Developing a transistor control for the solenoid valves
Figure 11-24 Mid stage shot of switch shield and instrumentation board with soldering
Figure 11-25 LCD driven by HD44780 and Atiny microcontroller
Figure 11-26 Finished Instrumentation board with datalogging capability
Figure 11-27 Finished switch shield with hardware debouncing