FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Development of an Autonomous Underwater Profiler

João Manuel Matos Monteiro

FOR JURY EVALUATION

Mestrado Integrado em Engenharia Electrotécnica e de Computadores

Supervisor: Nuno Alexandre Cruz

July, 2017

Resumo

A monitorização dos oceanos é crítica para perceber o nosso planeta sendo necessário recolher dados sobre as propriedades da água para melhor compreender os fenómenos dos oceanos. Uma das formas de recolher estes dados é através de perfis verticais da coluna da água que permitem perceber a variação das propriedades da água com a profundidade. Tendo em conta a dificuldade de obter medidas por operadores humanos dentro de água, vários dispositivos autónomos têm vindo a ser desenvolvidos com o intuito de realizar estas operações de forma sistemática, fiável e com a mínima intervenção humana.

Um *profiler* é um veículo que se move predominantemente ao longo do eixo vertical. Este tipo de veículo autónomo move-se principalmente na coluna vertical de água e deriva no plano horizontal onde é sujeito a correntes. Este é particularmente indicado para aplicações onde se pretende deslocamento predominantemente na vertical, uma vez que este tipo de sistema é mais eficiente devido ao seu desenho que minimiza o arrasto na direção vertical. Sendo um veículo autónomo, permite reduzir a logística associada a uma missão, sendo por isso mais versátil.

Para fazer descer o *profiler* existem dois sistemas de propulsão que são tipicamente usados: ajustar a flutuação do veículo para o deslocar para cima e para baixo ou usando propulsores para mover o sistema.

Os *profilers* que se movem através de flutuação fazem-no alterando a sua flutuação que faz com que estes afundem ou subam, sendo que a principal vantagem deste tipo de propulsão é o facto de apenas necessitar de energia nas alturas de alterar a sua flutuação, o que faz com que sejam bastante eficientes para grandes descidas. No entanto, uma vez que a flutuação do veículo depende da densidade da água, este não é indicado para ambientes a baixas profundidades, uma vez que, a densidade da água varia significativamente, afetando o desempenho do veículo. Além disso, em ambientes com baixas profundidades, as descidas e subidas são de menor duração aumentando a frequência dos ajustes de flutuação e, consequentemente, o consumo.

No caso dos *profilers* que usam propulsores para mover o sistema, o desempenho durante a descida não é tão afetado pela densidade da água uma vez que é mais fácil controlar a velocidade de descida controlando os motores devidamente. A principal desvantagem deste sistema de propulsão é o facto de precisar sempre de energia para mover o veículo, o que diminui a sua eficiência para mergulhos a grandes profundidades.

Neste trabalho apresenta-se o desenvolvimento de um novo *profiler* autónomo para realizar perfis em zonas costeiras. O objetivo principal é que este permita adquirir perfis até 200 m de profundidade. Além disso, este veículo deve ser fácil de transportar para facilitar a logística durante uma missão e permitir adicionar sensores.

Abstract

Ocean's monitoring is critical for understanding our planet and data on water properties needs to be gathered to better understand oceans phenomena. One of the ways of collecting these data is through vertical profiles of the water column that allow to understand variation of water properties with depth. Taking into account the difficulties of obtaining measurements by human operators in the water, various autonomous profiling vehicles were developed that allow to perform these tasks in a systematic and more reliable way and with minimal human intervention.

An underwater profiler is a vehicle that moves predominantly along the vertical axis. This type of autonomous vehicle moves mainly in the vertical column of water and drifts along the horizontal plan subjected to currents. It is particularly suitable for applications where the movement needed by the vehicle is predominantly vertical, where these systems are more efficient because of their design that minimizes drag on the vertical direction. Being an autonomous vehicle greatly improves the versatility of the system and reduces logistics during the mission.

To drive a profiler downwards in the vertical direction there are two propulsion systems commonly used: adjusting the buoyancy of the vehicle to make the profiler move up or down or using thrusters to drive the profiler.

Buoyancy driven profilers move by changing their buoyancy that causes the vehicle to either sink or rise. The main advantage of this type of propulsion is that it only uses energy to change its buoyancy making them extremely energy efficient on great descents. However, since the buoyancy of the vehicle depends on water density, this type of propulsion is not suitable for shallow water environments where water density varies greatly and affects the performance of vehicle. Moreover, in shallow waters, the frequency of buoyancy adjustments is greater due to the small descents causing more power consumption in these conditions.

Thruster driven profilers use thrusters to move the vehicle. The downwards performance of the thruster is not affected by the water density as much as the buoyancy, since it's easier to control descent rate with the appropriate thruster control. The main disadvantage of using a thruster is that it is an active propulsion that constantly needs energy to drive the profiler making it inefficient for dives in greater depths.

In this work it is presented the development of a new autonomous profiling vehicle for coastal areas. The main objective of the vehicle is to acquire profiles up to 200 m of depth. This vehicle must be portable to ensure low logistics during deployment and allow to easily add payloads.

Agradecimentos

Antes de mais, quero agradecer à minha família, e de uma forma especial aos meus pais e ao meu irmão, por me acompanharem ao longo da minha vida.

Além disso, agradeço ao meu orientador, o professor Nuno Cruz, que me acompanhou ao longo desta dissertação e me orientou ao longo de todo o processo, bem como a todo o pessoal do CRAS que esteve sempre disponível para me ajudar ao longo deste caminho.

Não posso deixar de agradecer também de uma forma especial ao André Pinto e ao Vitor H. Pinto pois, além de contribuirem com muitos momentos de descontração, tiveram também mão neste trabalho durante as longas discussões sobre os problemas que foram aparecendo.

Agradeço também a todos os amigos que fiz durante estes 5 anos de faculdade por estarem sempre disponíveis para ajudar e por todos os momentos que passámos em conjunto.

Por último mas não de forma menos especial, agradeço aos meus amigos de sempre pela amizade e disponibilidade para os bons e menos bons momentos desta vida.

Um muito obrigado a todos. Esta dissertação também é vossa.

João Monteiro

"How many roads must a man walk down
Before you call him a man? "

Bob Dylan

Contents

List of Figures

List of Tables

Abbreviations

ADC	*Analog-to-Digital Converter*
ALACE	*Autonomous Lagrangian Circulation Explorer*
APEX	*Autonomous Profiling Explorer*
ASIP	*Air Sea Interaction Profiler*
ATX	*Advanced Technology Extended*
AUV	*Autonomous Underwater Vehicle*
AVP	*Autonomous Vertical Profiler*
BBB	*Beaglebone Black*
CB	*Center of Buoyancy*
CG	*Center of Gravity*
CPU	*Central Processing Unit*
CRC	*Cyclic Redundancy Check*
CTD	*Conductivity Temperature Depth*
DOF	*Degrees of Freedom*
ESC	*Electronic Speed Controller*
FIFO	*First In First Out*
GPIO	*General Purpose Input/Output*
GPS	*Global Positioning System*
HUP	*Hybrid Underwater Profiler*
I2C	*Inter Integrated Circuit*
IBPS	*Intelligent Battery and Power System*
IMU	*Inertial Measurement Unit*
IPC	*Inter Process Communications*
LQR	*Linear Quadratic Regulator*
PID	*Proportional Integral Derivative*
PMIC	*Power Management Integrated Circuit*
PWM	*Pulse Width Modulation*
RAM	*Random Access Memory*
RMC	*Recommended Minimum Data for GPS*
RPi	*Raspberry Pi*
SBC	*Single Board Computer*
SPI	*Serial Peripheral Interface*
TTL	*Transistor-Transistor Logic*
UART	*Universal Asynchronous Receiver/Transmitter*
USB	*Universal Serial Bus*

Chapter 1

Introduction

In a ever changing world where most of the surface is covered by water, there is a need to understand, monitor and explore the oceans and rivers to have a better knowledge of our planet.

The oceans and rivers house innumerable ecosystems that balance the world. In order to protect the richness of those ecosystems we need a deeper understanding of the conditions that house them [1, 2]. In the depths of the world's oceans there are also unexplored regions that have a great profit potential like deep sea mining, oil and gas [2]. There's also the need to survey the currents and the interaction between atmosphere and the sea to better understand meteorological phenomena [3, 4]. Data collected in oceans is also used to track climate change and estimate future changes using forecast models. In order to have a better knowledge on this issue, we need accurate measurements that allow to understand the changes that are happening in the weather [5]. Finally, there's also the need to inspect underwater structures like cables and pipelines or foundations of constructions like bridges, to diagnose faults and prevent malfunctions in these critical systems [6].

All these possibilities present challenges related to underwater exploration since, in most cases, it is undesirable to have humans in such conditions. Therefore, there's the need to develop systems that allow us to overcome those challenges.

1.1 Context

In order to tackle the above mentioned challenges, a variety of solutions have been developed. Taking into account the difficulties of obtaining underwater measurements in a systematic way, autonomous vehicles that allow to perform these tasks in a systematic and reliable way and with minimal human intervention were developed. These vehicles can acquire data in both horizontal and vertical plans and in zones that have different constraints.

To gather a better knowledge of the water properties, there is the need to have measurements along the vertical profile of the water column to understand its properties variation with growing depth. These measurements allow to know how the different conditions vary in a certain region as depth grows. Similarly, in order to descend to deepest zones of the ocean, there's the need of

systems that allow predominantly vertical movement along the water column. In order to achieve these goals, different solutions were explored.

The Rosette conductivity–temperature–depth (CTD) device is a system that allows to obtain a profile using a winch in a boat but these method is labour intensive and slow [7].

To acquire profiles with minimal human operation autonomous profilers have appeared, like the Autonomous Profiling Explorer (APEX) or the Autonomous Lagrangian Circulation Explorer (ALACE). This type of profilers move in the water column by changing their buoyancy, but have low speeds of descent, that make them unsuitable for coastal areas profiles because these areas change more dynamically than open sea zones [7]. Additionally, this kind of profiles are not indicated for coastal profiles due to the higher variation of water densities in shallow water zones that causes problems on the propulsion system [8]. Moreover, since in shallow waters the descents are smaller, the frequency of the buoyancy adjustments increases making this propulsion system less efficient in these conditions.

1.2 Motivation

The superficial layers of the oceans are key areas to understand the world that surrounds us. To gather that knowledge, we need to study and monitor the coastal areas in order to understand the ecosystems [8, 7] as well as the interactions between the atmosphere and the ocean [4].

To acquire data on the superficial layers of the oceans, a new approach was used that allows to minimize the problems presented by buoyancy driven profilers. By using a thruster [4, 7] on the profiler, the problem introduced by the variability of water densities is minimized.

Following this approach, it will be presented a thruster driven profiler that minimizes the disadvantages of the buoyancy driven profilers in these areas and develops the work already done to propose a different solution from those already explored. Additionally, the proposed vehicle also allows limited horizontal motion by controlling the heading of the vehicle.

1.3 Goals

The main goal of this work is to develop an autonomous underwater profiler, ie, an autonomous vehicle that moves predominantly in the vertical column of water.

The secondary goal of this work is to build a system that can be expansible and implement additional control algorithms that allow more maneuvers.

1.4 Document Organization

This document describes the work done during the development of the profiler from the conception of the system until testing and conclusions on the work done.

Therefore, chapter 2 and chapter 3 can be understood as the conception phase, chapter 2 presents the state of art while chapter 3 defines the requirements and proposes a possible solution.

Chapters 4, 5 and 6 correspond to the implementation of the system. Chapter 4 focuses on the implementation of the prototype, chapter 5 presents the model used to simulate the vehicle behavior and chapter 6 presents the controllers designed to maneuver the vehicle.

In chapter 7 the profiler is tested and the achieved results are presented.

Finally, chapter 8 concludes this work: the results are discussed and some suggestions of future work are presented to further improve the system developed.

Chapter 2

State of the Art

An underwater profiler is a vehicle that moves predominantly along the vertical axis. This type of autonomous vehicle moves mostly in the vertical column of water and drifts along the horizontal plan, subjected to currents. They are particularly suitable to applications where the movement needed by the vehicle is predominantly vertical, where these systems are more efficient due to their design that minimizes drag on the vertical direction.

Vertical profiles are important for identification of water properties variation with depth for instance the temperature-depth typically used in oceanographic studies. Typically, this kind of systems are used to profile the water column to gather information about water properties at great [9] or low depths [4] and to maintain a fixed distance to seabed [10]. An autonomous vehicle greatly improves versatility and reduce logistics during the mission compared to other alternatives, like Rosette CTD systems that depend on a support vessel. This characteristic allows to deploy the vehicle and recover it after a mission plan sent to a profiler is executed without human intervention.

The main disadvantage of this type of vehicle is that it is not controlled along the horizontal plan, being subjected to drift caused by currents which makes it difficult to control its position.

A profiler is composed by a set of subsystems that allow the vehicle to move according to a planned mission. To achieve this, a profiler is typically composed by the following subsystems:

- a propulsion system, usually based on buoyancy or thrusters, as described in section 2.1;
- a power system (composed of batteries and voltage converters like the described in [4]) responsible for providing power to all the subsystems. Typically it is used a set of rechargeable batteries like Lithium-ion batteries [4, 7];
- a set of sensors that are used to control the movement of the profiler and to acquire the desired data during the mission. This set can be divided in two: navigation sensors, to know the vehicle position and payloads, to acquire data on water properties. The pressure sensor can be considered a navigation sensor since it allows to measure the depth and, at the same time, a payload since it allows to measure depth for Temperature-depth profiles;
- a communications system to send gathered data during the mission and the location of the profiler;

- a central processing unit with one (like the Netburner MOD5234 used in [8]) or more processors (in a distributed architecture like the used in [7]) that controls the mission and processes data

More subsystems can be added to provide more functionality like a mooring station like the one used in [11]. In [4], a light is also mounted in the vehicle to locate the vehicle at night as well as a magnetic switch to wake up the system. In [12], an airdrop device is added to safely drop the vehicle from the air.

2.1 Propulsion

To drive a profiler downwards it is necessary to have a propulsion system. Therefore, two propulsion systems are the most commonly used: adjusting the buoyancy of the vehicle to make the profiler move up or down [13, 12, 14, 15] or using one or more thrusters to drive the profiler down and combining a positive buoyancy with propulsion to move up [4, 7].

Buoyancy driven profilers move by changing their volume, which causes the vehicle to either sink or rise. The main advantage of this type of propulsion is that it only uses energy to change its buoyancy, making them extremely energy efficient on great descents and, therefore, more suitable to use in great depths [13, 9]. However, since the buoyancy of the vehicle depends on water density, this type of propulsion is not suitable for shallow water environments where water density varies greatly and, therefore, affect the performance of vehicle. This limitation causes the vehicle to have greater settling times [8, 15] that can be critical for the performance of the system in environments as dynamic as coastal waters. In these zones these vehicles also spend more energy since they need to adjust buoyancy more often than in greater descents. Moreover, changing the buoyancy of the vehicle takes longer than actuating the thrusters making it more difficult to control the profiler.

Another approach used to minimize the limitations of the buoyancy driven profilers is to use a set of thrusters as propulsion, normally associated with a positive buoyancy [4, 7]. The positive buoyancy is used as safety measure in case of electronics failure making the profiler always come back to the surface [16], although having the drawback of requiring more energy to drive the profiler downwards. The downwards performance of the thruster is not affected by the water density as much as the buoyancy and it allows to achieve greater speeds and shorter settling times as shown in [10]. The main disadvantage of using a thruster is that it is an active propulsion, that constantly needs energy to drive the profiler, making it inefficient for dives in greater depths or to stop in the bottom of the sea, which can be required to some applications (like the suggested in [17]).

2.2 Software Architecture

The profiler software that allows to execute a planned mission is similar to other AUV's and typically can be divided in two main subsystems: Mission Control and Data Logging. The software can be implemented in either a distributed architecture like seen in [7], where different processing

units perform different tasks or in a centralized architecture where there is only one processing unit that runs all the tasks [18].

2.2.1 Mission Control

Mission control executes the mission plan acquiring data from sensors related with the vehicle movement (like depth and attitude of the vehicle). It also controls the vehicle propulsion according to the mission plan. In [4], mission control is also responsible of controlling remote communications and wake up the system. In [7], a distributed architecture is proposed based on nodes (each node is a different processing unit) where one is responsible for controlling the mission and the communications, another is responsible for data acquisition of vehicle payloads and thruster control and the other node acquires all scientific data. In [18], a main process estimates the vehicle position and executes the mission plan.

2.2.2 Data Logging

The data acquired from sensors needs to be logged to a storage device to be analyzed after the mission. The data is typically logged to an SD card [8] or a hard disk [4].

In [18] this is achieved using an independent process to increase the system modularity and robustness. The other processes send a message to the logging process that is responsible to log the data. Each measure is typically associated with a timestamp to ensure data validation and synchronization between multiple processes that run concurrently.

2.3 Motion Control

To perform the required maneuvers, a profiler must run an algorithm that controls the propulsion system.

In the buoyancy driven profiler described in [8], a task implements a PID controller using the measurements of depth and altitude above seabed. This controller computes an intermediate depth trajectory using basic filtering to keep the depth changes within the dynamic capabilities of the profiler.

To acquire a profile of the column of water, the vehicle described in [7] runs at a constant speed and, as it nears the target depth, ramps down the thruster voltage to zero bringing the profiler close to the target depth. Then, the profiler ascends using the positive buoyancy defined on construction. The vehicle described in [4] operates in a similar way. The AVP [7] can also hover at a fixed depth. It uses a controller comprising of Linear Quadratic Regulator (LQR) in combination with a complementary filter as described in [10].

In the horizontal plan, a profiler normally drifts with the current. To have some control along the horizontal plan, [17] have proposed a method to select the currents in order to move to the desired location, using the ones that move in that direction or wait in the seabed until the currents point in the desired direction. To overcome the limitation on horizontal movement [14] proposed

a buoyancy driven approach that combines characteristics of a vertical profiler and an underwater glider to correct the horizontal displacement caused by the influence of currents, allowing to persistent monitor a certain region without drifting.

Another approach to solve the drift problem introduced by the currents is the moored profiler like the ones presented in [19], [20] or [21]. Since these vehicles are moored, they don't drift with the current allowing to generate water column profiles in the desired location, even in adverse weather conditions ([21]). However, since these are moored in the seabed they are not as versatile as other profilers.

2.4 Operational Examples

As seen along this chapter, many different approaches have been used in the construction of autonomous profilers. In table 2.1, characteristics of some operating vehicles are presented and compared.

From the table 2.1 it is easy to confirm that profilers driven by motors are more suitable for lower depths applications, while buoyancy driven profilers are more suitable for greater depths. It is also possible to conclude that the weight of profilers varies considerably between devices. In this case the added weight is related with more payloads and the fact that, in the heaviest system, are used two sets of sensors for the same measures [4]. This system also has the largest battery, which adds more weight.

Table 2.1: Comparative table of different profilers

	AVP [7, 11]	HUP [14]	Airdropped profiler [12]	ASIP [4]	APEX Argo [22]	APEX Deep Float [9]
Length	1.17m	2.1m	0.70m	2m	1.27m	0.432m
Diameter	0.184m	0.23m	N.P	0.19m	0.165m	0.432m
Weight on Air	∼13Kg	N.P	7Kg	80Kg	25 Kg	N.P
Max. Depth	200m	N.P	1000m	100m	2000m	6000m
Propulsion	DC Motor	Buoyancy	Buoyancy	DC Motor	Buoyancy	Buoyancy
Speed (m/s)	Nominal =0.4; Max =1	N.P	N.P	Ascend ∼0.25	N.P	N.P
Endurance	7 days@4 dives per day (100 m)	30 days@3 dives per day	50 profiles	N.P	4 years, 150 profiles	300 profiles
Power	Li-Ion 324 Wh	N.P	N.P	Li-ion 1000Wh	N.P	N.P
Pressure	✓	N.P	✓	✓	N.P	N.P
CTD	✓	N.P	N.P	✓	✓	✓
Temperature	✓	✓	✓	✓	✓	✓
Depth	✓	✓	✓	✓	✓	✓
GPS	✓	N.P	✓	✓	N.P	N.P
Turbidity	✓	✓	N.P	✓	✓	✓
Fluorometer	✓	N.P	N.P	✓	✓	✓
Echo-sounder	✓	N.P	N.P	N.P	N.P	N.P
Dissolved Oxygen	✓	N.P	N.P	N.P	✓	✓
Attitude	N.P	✓	✓	✓	N.P	N.P
Acoustics	N.P	✓	N.P	N.P	✓	N.P
Shear	N.P	N.P	N.P	✓	N.P	N.P
PAR	N.P	N.P	N.P	✓	N.P	N.P
Radiometers	N.P	N.P	N.P	N.P	✓	N.P
Nutrients	N.P	N.P	N.P	N.P	✓	N.P
Transmissometers	N.P	N.P	N.P	N.P	✓	N.P
Satellite Communications	Optional	✓	✓	✓	✓	✓
RF communications	✓	N.P	N.P	N.P	N.P	N.P

*N.P - Information not provided

Chapter 3

System Overview

In this chapter it will be presented an overview of the system that was developed, as well as the usage scenarios that lead to a set of requirements and assumptions needed to develop the system. It will also be presented a functional diagram that allowed to implement the presented requirements and the software architecture.

The presented system was designed to operate mainly in coastal areas. It aims to be a portable and easy to deploy autonomous vehicle, with enough energy for a 24 hour mission. It will also provide more versatility than similar vehicles by allowing to perform some movement along the horizontal plane, hover on a defined distance to the sea bottom and profile on a specific location (compensating the displacement caused by the currents).

3.1 Usage Scenarios

In order to understand the addressed challenges, a set of usage scenarios was identified to illustrate the use of the vehicle in real world application. Some of those usage scenarios are:

- Profile the coastal area to acquire data for oceanographic studies;
- Profile along a pollution spot to measure the thickness of the plume;
- Acquire profiles on zones with maritime traffic (for instance in a port) to acquire data for oceanographic studies without disturbing maritime traffic;
- Acquire images of the sea bottom with a fixed distance (benthic fauna and flora);

3.2 Requirements and Assumptions

From the analysis of the usage scenarios were identified a set of requirements and assumptions that can be divided in three subsections:

- prototype requirements, related with the characteristics that the prototype must have;

- operational requirements, related to the maneuvers that the profiler must perform;
- safety requirements, that should guarantee the fault handling of the vehicle.

These assumptions allow to define requirements like battery life and weight of the vehicle.

3.2.1 Prototype Requirements

The list of prototype requirements takes into account the following guidelines: the profiler must be an autonomous system with a modular design and that allows low logistics operation during deployment and recovery. This system should be designed to operate on coastal areas without wired connections. The defined guidelines imposed the following requirements:

- Battery life should be sufficient to perform 10 profiles to 200m depth with a velocity of 1m/s along 24 hours;
- At the end of the mission, the vehicle should communicate its position on water to be recovered;
- The profiler should have a configuration that allows to be easily recovered from the water;
- The mission plan must be communicated using a wireless connection;
- The prototype must allow to add new sensors if needed;
- The vehicle must be designed to profile the coastal area;
- The system should have a weight on air and length not greater than 30 Kg and 1.5 m respectively;

3.2.2 Operational Requirements

The system should allow a defined set of maneuvers that take into account the usage scenarios considered. The list of requirements related with operation requirements is as follows:

- The controllers should allow to profile in the same position (in case of drift caused by currents, the profiler should recover to its original position);
- The profiler should allow to acquire different profiles along a horizontal direction ;
- The vehicle should allow hovering at a fixed distance above the bottom of the sea;
- The vehicle should allow to profile within 2 different depths without needing to surface;
- The profiler must need only one operator to successfully launch and recover the vehicle;
- The vehicle should work on calm conditions;

3.2.3 Safety Requirements

The profiler must detect situations that can cause damage to the system. The main situations identified are: water entrance on the pressure housing, overheating caused by malfunction of the system, collision with the bottom of the sea and battery levels insufficient to conclude the mission plan. The profiler should then be able to detect the following faults and act accordingly:

- Detect water entrance;
- Detect overheating inside the pressure case;
- Sea bed detection to avoid collision;
- Monitor battery status;

3.3 System Design

To better understand the overall design of the profiler, figure 3.1 illustrates the main subsystems and interconnections.

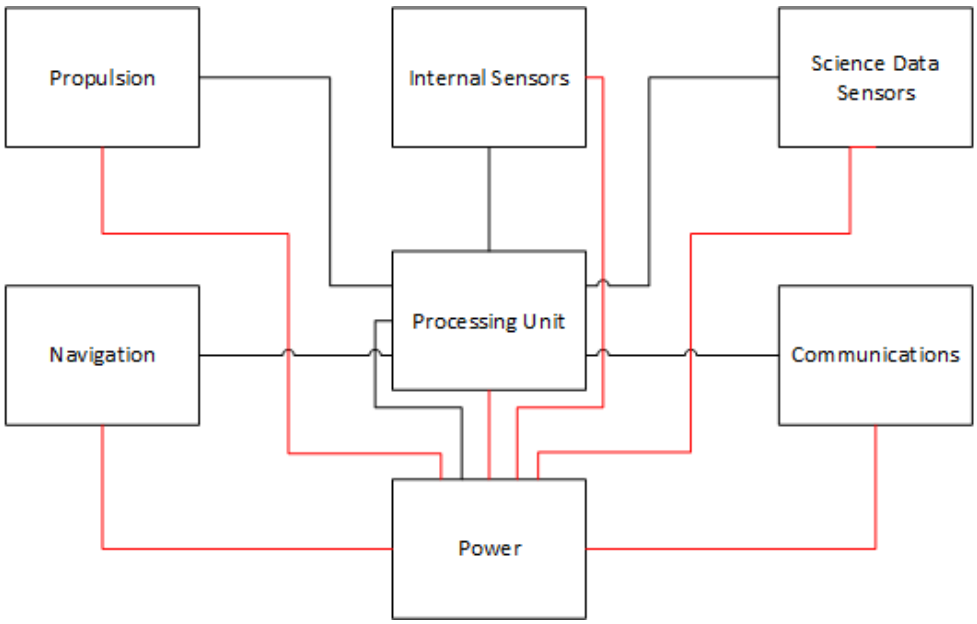

Figure 3.1: Subsystems that constitute the profiler

These include:

- Power System - Powers all the subsystems. Includes the batteries, voltage converters and power buses needed to power the other systems;
- Processing Unit - Runs the onboard software described in section 3.4;
- Propulsion System - Responsible for propelling the profiler;

- Navigation System - Includes the sensors that allow the profiler to know it's position, heading and attitude;

- Scientific Data System - Set of sensors used to acquire scientific data (for instance depth and temperature);

- Communication System - Responsible for communicating with the outside world. Allows the profiler to receive the mission plan and communicate its position in the end of the mission;

- Internal Sensors - Set of sensors to monitor possible faults inside the pressure housing;

3.3.1 Physical Diagram

In this system, the position of the subsystems is crucial to their correct performance since some subsystems only work above the water level while others need to be in contact with the water and others can't be in contact with water.

Additionally, the passive attitude of an underwater vehicle is determined by the relative position of his centre of mass and centre of buoyancy. In order to achieve the desired attitude, (vehicle pointing downwards) the gravity centre of the vehicle has to be below the buoyancy centre. For this to be possible, the weight must be concentrated in the underside of the vehicle.

Taking into account these constraints, a physical diagram was designed to illustrate the position of each subsystem in the vehicle:

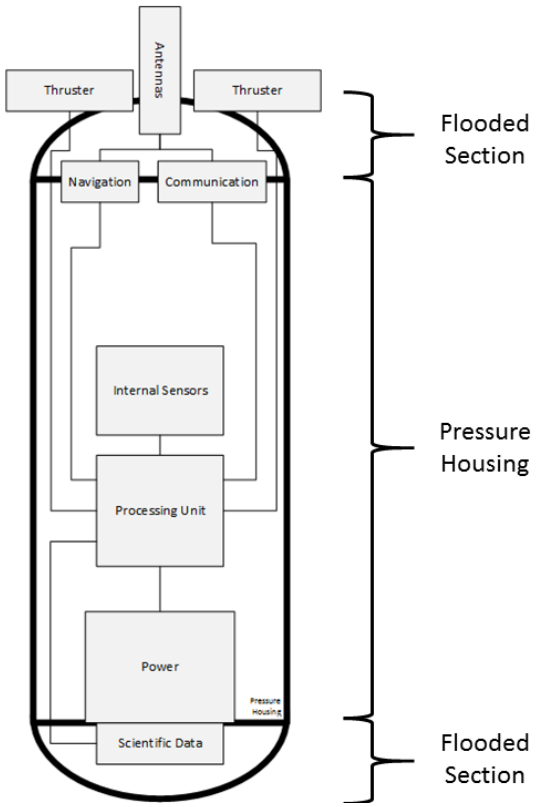

Figure 3.2: Physical diagram

Both the top and bottom are free flooding and only the pressure housing needs to be waterproof. In figure 3.2, thrusters are mounted on the top of the profiler to drive the profiler down. As seen in chapter 2, the vehicle should have a positive buoyancy for safety purposes. The set of antennas also need to be in the upper side of the profiler to ensure that when the profiler comes to the surface, the antennas are outside of the water to guarantee proper functioning. The navigation and communication systems are mounted near the antennas and need to be waterproof, since electronics can't be in contact with water. The power system, the processing unit and internal sensors are mounted inside the pressure housing to avoid contact with water. The power system is placed at the bottom since it represents a great weight. This placement contributes to balance the profiler. The scientific data sensors are mounted in the interface with the outside because they need to be in contact with water to accurately measure the water properties.

3.3.2 Functional Diagram

Taking into account the defined requirements in section 3.2, a proposed designed is presented in figure 3.3, that details the main subsystems presented in figure 3.1.

In the following design, the subsystems' position is the same as in the system to be implemented. This is an important aspect of the design, since the position of the components has an

impact on the correct operation of the subsystem (for instance, the GPS antenna needs to be at the top of the profiler because GPS only works if the antenna is above the water surface).

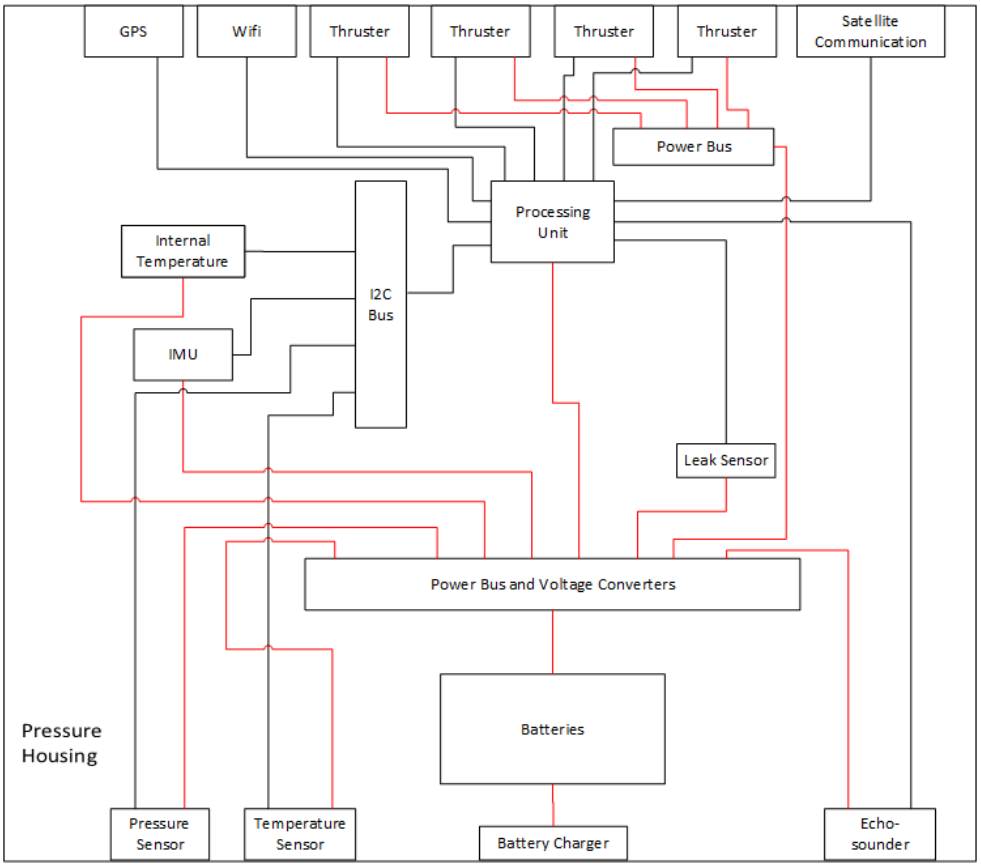

Figure 3.3: Functional diagram

The functional diagram presented in 3.3 is composed by:

- a Processing unit, responsible for processing data, control the thrusters and mission management;
- four thrusters to drive the profiler and allow to control the heading;
- a GPS module to obtain the position of the vehicle when its antenna is above the water level;
- a Wi-fi module to receive the mission plan before deployment and retrieve data from the profiler;
- a satellite communication module to transmit vehicle position in the end of the mission;
- a pressure sensor to obtain pressure and calculate depth;
- an echo-sounder to measure distance to the bottom;
- an internal temperature sensor to detect overheating;

- an IMU to estimate vehicle attitude and heading;
- an external temperature sensor to measure water temperature;
- batteries to power the system;
- a charger controller to charge batteries (only used off mission);
- power converters and power buses to power the different devices;
- an I2C bus where the devices that use I2C are connected, to communicate with the processing unit;

During the lifetime of the vehicle, more sensors or modules can be added as needed. Additional sensors are restricted by space on the device, by the number of interfaces that the processing unit offers and the compatibility with the processing unit.

3.4 Software Architecture

The software architecture should be implemented in a way that allows an easy accommodation to new sensors, since the modularity of the system is also a considered requirement. This software must run in real-time and have multiprocesses to ensure that different tasks are performed in useful time.

3.4.1 Software Modules

To satisfy the set of requirements defined in section 3.2, a set of software modules is needed to interconnect all the subsystems mentioned in section 3.3 and to perform tasks such as reading data from the sensors, control the thrusters, communicate position, save data to a storage device, etc.

Having these guidelines in mind, the software to be developed consists of three separate processes as shown in figure 3.4: Mission Control and Supervision, Data Logger and Sensors.

The process Sensors should acquire data from all the sensors available and communicate these measurements to the processes that use them. The time associated with each acquired measurement should be registered and also communicated to other processes.

Since writing data to the memory is a slow task, the process Data Logger is responsible for logging the data read from the sensors. This process should implement a queue that guarantees that no data received is lost. The data values in the queue will then be written to disk, allowing the user to see the evolution of all data acquired in the end of the mission. All data should be timestamped with the time of acquisition. This process runs periodically and each time the configured period elapses, this module accesses the data read from the sensors and saves it to the queue to be written in a logging file.

The process Mission Control and Supervision is responsible for executing the mission plan received by the user, taking into account the data acquired by the Sensors process. This process

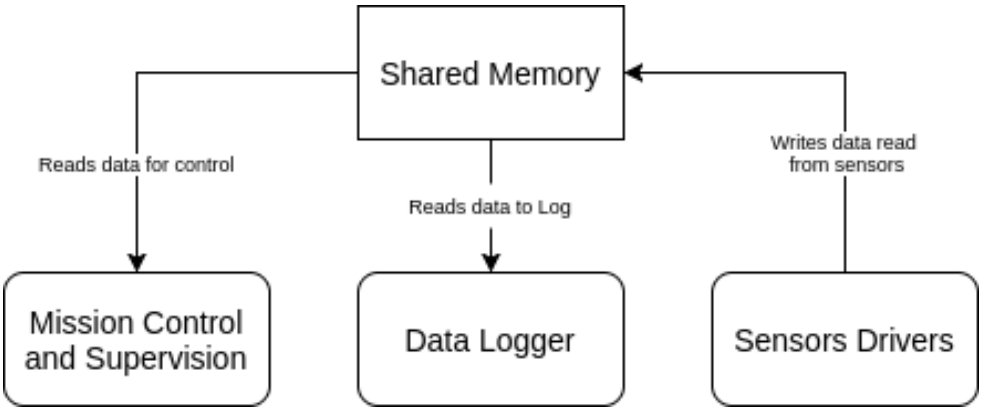

Figure 3.4: Software modules

needs to estimate the position of the vehicle, based on sensor readings, and control the thrusters to perform the desired maneuvers. This process is also responsible for, in case of fault detection or in the end of the mission, surface and communicate the vehicle position.

Each mission is planed using a configuration file where each maneuver to be performed is described. The Mission Control and Supervision process builds a state machine based on the information present on the configuration file that executes the mission plan defined sequentially. For each maneuver is required to define a timeout as a safety measure to guarantee that, in case the maneuver is not completed, the profiler does not get lost trying to complete it.

3.4.2 Device Drivers

To communicate with each device attached to the processing unit is necessary to use a device driver.

Therefore, for each device, it will be implemented a class with methods that provide an interface for the software modules interact with each device. Each class has to be capable of initialize a device and allow to read data from this.

3.4.3 Inter Process Communication

Since each software module is implemented in a different process, it is necessary to use methods to communicate and share data between processes and, at the same time, guarantee that the shared data is valid.

Therefore, the data will be shared between processes using shared memory that allows to share a memory segment between the different processes running.

As this method does not guarantee synchronization between processes, semaphores will be used to guarantee synchronization between processes.

Chapter 4

Implementation

During this chapter, it will be presented the implementation of the vehicle shown in figure 4.1. It will describe the bill of materials used in the implementation of the vehicle, the criteria considered in their selection, the onboard software implemented in the processing unit and the design of the controllers that allow to perform the required maneuvers.

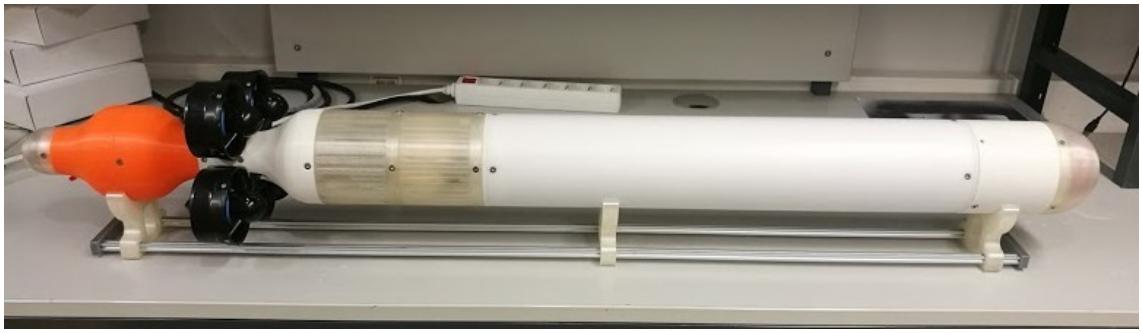

Figure 4.1: Prototype of the Autonomous Underwater Profiler

4.1 Bill of materials

This section describes the criteria considered in the selection of the materials. The materials used take into account the requirements described in Chapter 3.

4.1.1 Processing Unit

Scientific literature on autonomous underwater vehicles describes several works using Linux as operating system [18, 15, 23]. This operating system is free and open-source and is suitable for this kind of application since it allows multitasking, inter process communication and allows to interact with other devices using device files.

For the selection of the processing unit the main selection criteria were: power consumption, that should be as low as possible to allow more mission time, interfaces available to communicate

with the other components, weight, since the system weight is a requirement and size, because space available is limited.

Single board computers are typically small size and lightweight. Having this in mind, two single board computers were considered: the Raspberry Pi (RPi) [24] and the Beaglebone Black (BBB) [25], since both are capable of running a Linux distribution, consume low power and provide various interfaces. The choice between these two embedded systems was made based on the set of requirements defined in Chapter 3 and on the Table 4.1 that compares both their main features.

Table 4.1: Comparative table of Single Board Computers [24],[25]

Characteristics	Raspberry Pi 3 Model B	Beaglebone Black Wireless
CPU Cores	4	2
CPU Clock Frequency	1.2 Ghz	1 Ghz
RAM	1 GB	512 MB
Storage	Micro SD	4 GB;Micro SD
GPIO	40	65
I2C Bus	1	2
SPI Bus	1	2
UART Ports	1	4
Analog Input	—	7 (12 bits)
Power consumption	∼400 mA @ 5V	210-460mA @ 5V
USB Ports	4	1
Ethernet	10/100 BaseT	—
Wifi	802.11b/g/n	802.11 b/g/n
Bluetooth	Bluetooth 4.1 plus BLE	Bluetooth 4.1 plus BLE

BBB provides more interfaces, which is an important factor considering that one of the defined requirements is the capability of adding more sensors. Another factor in favor of the BBB is the built-in PMIC that manages the board power sources.

RPi has more USB interfaces but it requires a memory card to support the board operating system, while the BBB has already 4GB of eMMC on-board flash storage, and still allows the usage of an additional memory card.

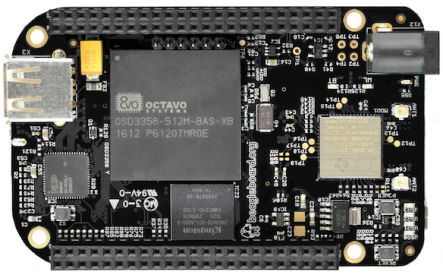

Figure 4.2: Beaglebone Black Wireless [25, 26]

Considering the facts presented above, the Beaglebone Black Wireless was chosen because it has more interfaces that are needed to communicate with the other components.

4.1.2 Propulsion System

Taking into account the requirements defined in Chapter 3, four thrusters were chosen to propel the vehicle. The thrusters selected were the T100 thrusters by BlueRobotics [27] (Figure 4.3) since they are an affordable solution for underwater propulsion and are able to provide the necessary forces to drive the profiler downwards. The use of four thrusters allows to control the attitude of the vehicle and, therefore, perform the desired maneuvers.

Figure 4.3: T100 thrusters [27]

The speed of the brushess motor is controlled by four Afro Slim ESC 20 Amp [28] (one for each motor). This ESC supports up to 20 Ampere of current which allows to drive this thruster (maximum current is 11.5 Ampere) and has a voltage range of 7.4 Volt to 14.8 Volt. This ESC also has a compact format which is good for this application since space inside the hull is limited. To control the speed it uses a PWM signal as its input.

4.1.3 Scientific Data Sensors

To provide relevant data about water properties it was used a temperature sensor and a pressure sensor. This last one allows to calculate the depth in meters.

The used temperature sensor was the Blue Robotics Celsius Fast-Response since this device is already prepared to be attached to an watertight enclosure. It provides data through an I2C and has an accuracy of $\pm$ 0.1^{o} C from -5 to 50^{o}C. The pressure sensor used was the Blue Robotics Bar30. This pressure sensor can measure up to 30 Bar at depths up to 300 m with 0.2 mBar (equivalent to 2 mm) resolution. It also uses I2C to communicate.

4.1.4 Navigation Sensors

The pressure sensor mentioned above is also a navigation sensor since it is used to calculate depth.

Additionally, the vehicle uses an IMU to calculate its orientation and a GPS to know its absolute position when at surface.

The IMU used is the SparkFun 6 DOF Digital Combo Board - ITG3200/ADXL345 [29, 30]. Combining the measurements from the accelerometer and the gyroscope present in this device, we can estimate the orientation of the system. This device communicates over I2C.

The GPS used is the Amaryllo Roots [31]. This device is used to obtain vehicle's position when at surface. This device has an USB interface to connect to the processing unit.

4.1.5 Communication System

The vehicle should be able to receive the mission plan using an wireless connection and communicate its position at the end of the mission.

To receive the mission plan, the processing unit used already has 802.11 b/g/n 2.4GHz WiFi interface that allows to access the system remotely. However, it was used TP-Link TL-WN725N Wireless USB adapter [32] mounted at the top of the profiler since, similarly to the GPS, WiFi does not work properly underwater.

4.1.6 Power System

As defined in section 3.2, the vehicle should have enough power to perform 10 profiles to 200 m, along 24 hours. Considering this requirement, it was necessary to choose a set of batteries that provided enough power. Taking into account that space is limited and weight should be as low as possible, it was used Li-ion batteries, since this type of batteries offer higher energy density.

To choose the number of batteries necessary to provide power, it was considered three moments of operation: idle, where the vehicle is waiting to perform a profile; during descent, where the profiler is performing a profile and transmitting its position at the end of the mission. Considering the power consumption of every component used it was estimated the consumption during these three phases.

During idle time the estimated consumption is 3.42 W. During this period, the Iridium module can be used in low power mode since the position is not being transmitted. During the descent the vehicle has all subsystems turned on except for the Iridium module that is on low power mode. To estimate the consumption of the thrusters we used data from the MARES AUV and assumed a value of 70 W to drive the profiler down. Considering the other systems, the final value estimated for the consumption during descent is about 75 W. During the transmission of the final position, the Iridium module is used to transmit current position causing an increase in power consumption yielding a power consumption of 5.75 W during this period.

To calculate the battery capacity necessary to meet the requirements, it was considered that the profiler moves at a speed of 1 m/s during profiling, spends 24 hours in idle mode and that it

spends half an hour in transmitting its position. Using the values presented above, the estimated capacity needed to fulfill the requirements is 126.95 Whr.

To manage power distribution to the system, as well as charging of the batteries after mission, it was used the Ocean Server BBDC-02R Dual Battery Controller with ATX Power Supply [33]. This module is an Intelligent Battery and Power System (IBPS) capable of providing the necessary power to the system and already has DC-DC converters to provide the necessary voltage levels.

To meet the capacity estimated above were used 2 Ocean Server BA95HC-FL [34] batteries yielding, a total capacity of 190 Whr. These are smart batteries that are compatible with the IBPS used and allow to easily access state data, like time to end and percentage to end.

4.1.7 Internal Sensors

As defined in section 3.2, the vehicle should detect water entrance and overheating inside the pressure case. Therefore, it was used a temperature sensor TMP 36 [35] to measure the temperature inside the pressure housing. This device measures temperature from -40^{o}C to 125^{o}C with an accuracy of $\pm$ 2^{o}C that is good enough to detect overheating.

4.1.8 Interfaces

To communicate with the RS232 devices, two MAX232 [36] were used. This integrated circuit is a dual driver/receiver that allows to convert TTL to RS232 and RS232 to TTL levels. Since the BeagleBone Black has only one USB port, it was used a 4 port USB powered hub [37] to interface with the GPS, camera and the external Wi-fi.

4.2 Hardware Implementation

The prototype was mounted using modules available in the laboratory, like the pressure housing and endcaps, and 3D printed parts like the supports for the thrusters or the nose of the profiler. Figure 4.4 presents the final configuration of the profiler.

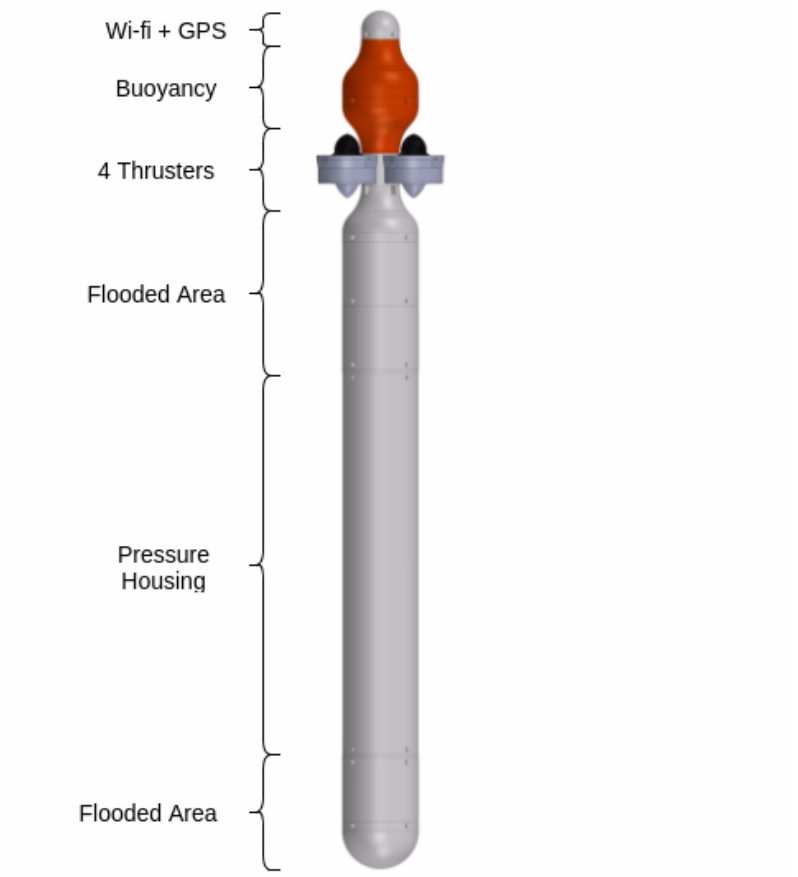

Figure 4.4: Final configuration of the profiler

The electronics are mounted inside the pressure housing fixed to an aluminum frame (figure 4.5). The batteries are mounted at the bottom to lower the center of gravity of the vehicle and are fixed in the tray using 3D printed supports . The IBPS is mounted close to the batteries to reduce the cable length between the batteries and the power controller. Since space inside the pressure housing is limited, the processing unit and the ESCs are mounted on the other side of the the IBPS.

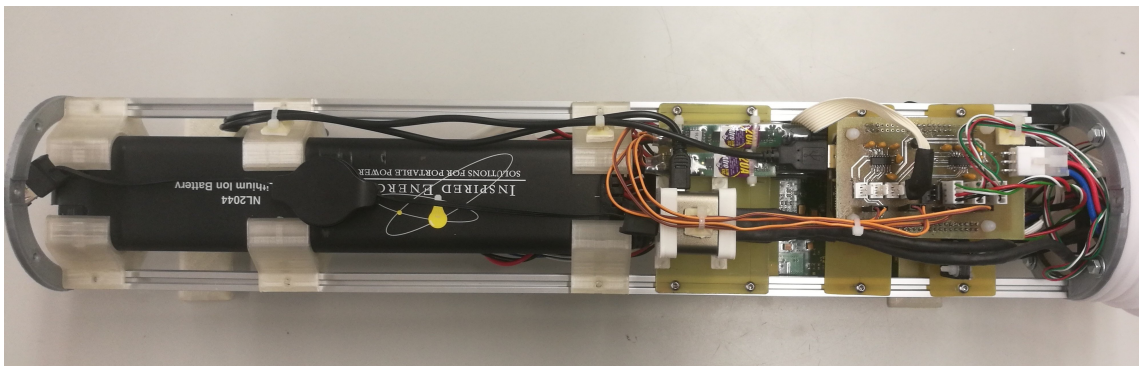

Figure 4.5: Frame and electronics mounted inside the profiler

On top of the processing unit is mounted an interface board (figures 4.6 and 4.7) where the sensors are connected.

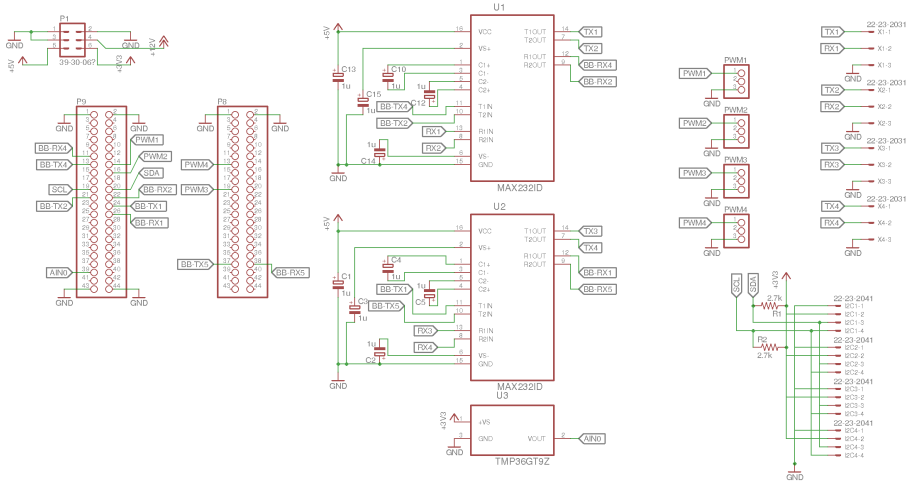

Figure 4.6: Schematic of the interface board

This board was designed to have 4 RS232 connections (and the MAX232 devices to convert voltage levels), 4 connectors to connect to an I2C bus and the analog temperature sensor to measure internal temperature. This board was developed to reduce wiring inside the pressure housing due to space limitation. This way, all the interfaces are connected in the same place.

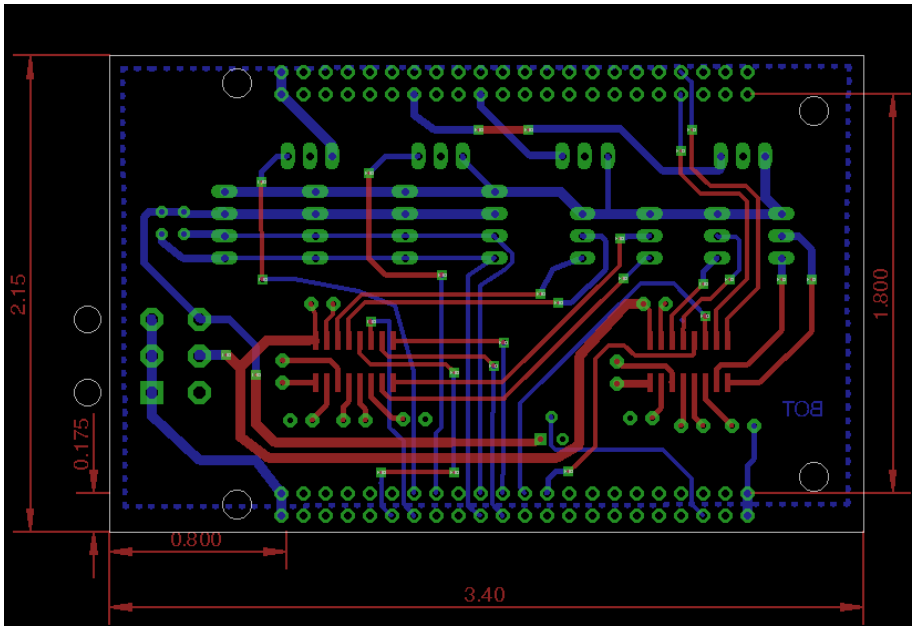

Figure 4.7: Interface board

In the endcaps that seal the pressure housing are mounted the connectors to the outside of the pressure housing. In the upper endcap (figure 4.8) are mounted:

- the pressure and temperature sensor, to reduce cable length inside the pressure housing since the upper endcap is closer to the interface board were the sensors are connected
- two connectors to the thrusters, that need to be connected to the ESCs. These two connectors are branched for the four thrusters due to space limitations in the upper endcap
- the USB cables to the Wifi module and GPS, since these need to be mounted at the top end of the profiler to ensure that are above water level when the vehicle is at surface
- the charging cable that allows to recharge the batteries without needing to disassemble the vehicle.
- a connector that seals the pressure housing. This was used to test the sealing of the pressure housing

In the bottom lid is mounted a USB connector to interface with a camera.

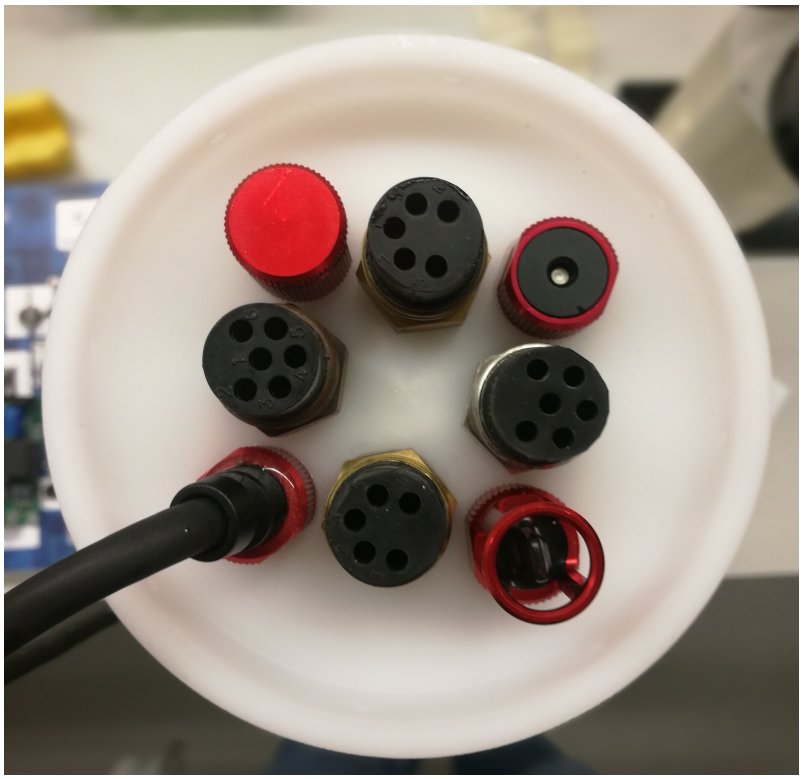

Figure 4.8: Top endcap

4.3 Software Implementation

In this section, the software implementation will be presented that allows to execute the architecture proposed in 3.4. It will also be described the interactions between different subsystems and how the different subsystems communicate with the processing unit.

4.3.1 Device Drivers

To implement the software architecture described in section 3.4 it is necessary to develop a set of device drivers that allow to interface with the different subsystems described in section 3.3. Interfacing with the different hardware devices (as the I2C bus, UART ports, etc) available in the BeagleBone Black uses device files which allow the software to interact with the device driver as if it were a file [38].

For each device it was developed a library that implements a class with a set of methods that allow to interface with the device. It was used the Blacklib C++ library [39]. This library allows to use GPIO pins and interface with devices using I2C, UART and SPI.

4.3.1.1 I2C Devices

In this vehicle there are three devices that use an I2C interface to communicate data (IMU, temperature sensor and pressure sensor).

The IMU is composed by an accelerometer and a gyroscope. Therefore, to read the measurements provided by the IMU it is necessary to read both the accelerometer and the gyroscope. The IMU class (figure 4.9) defined as follows:

```
class IMU{
    private:
        BlackLib::BlackI2C Acel;
        BlackLib::BlackI2C Gyro;
        float agains[3];//accelerometer gains
        int goffset[3];//gyroscope offsets
        float exInt, eyInt, ezInt;  // scaled integral error
        volatile float twoKp;      // 2 * proportional gain (Kp)
        volatile float twoKi;      // 2 * integral gain (Ki)
        volatile float q0, q1, q2, q3; // quaternion of sensor frame relative to auxiliary frame
        volatile float integralFBx,  integralFBy, integralFBz;
        float sampleFreq; // half the sample period expressed in seconds
        float invSqrt(float number);
        void AHRSupdate(float gx, float gy, float gz, float ax, float ay, float az);
        void GyroSetClockSource(void );
        void GyroSetFullScale(void );
        void GyroSet_ITG_and_RawData_Ready(void );
        void GyroCalibrate(int totSamples, int sampleDelayMS);
    public:
        IMU();
        bool readAcel(float *x,float *y, float *z);
        bool readGyro(float *x, float*y, float *z);
        void getValues(float * values);
        void getQ(float * q);
        void getYawPitchRoll(float * ypr);
        void getEuler(float * angles);
        void close();
};
```

Figure 4.9: IMU class

The initialization of the device is made according to the flowchart represented in figure 4.10. This device is initialized by configuring the gyroscope and accelerometer for the desired operation, waiting 70 ms for the gyroscope to initialize and saving the time that is used for angle calculation

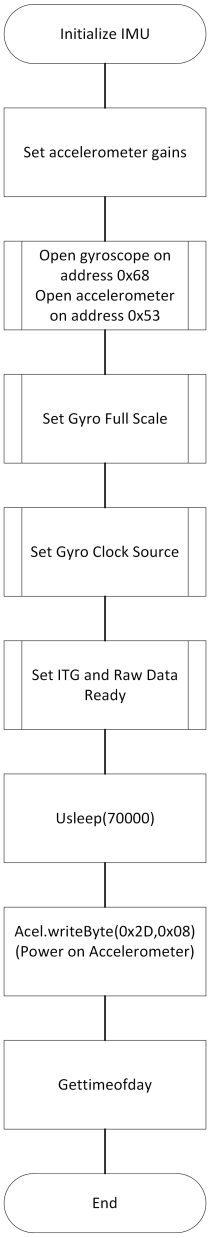

Figure 4.10: Flowchart of the initialization of the IMU

The estimation of the angles using the measurements from the accelerometer and the gyroscope is made using the algorithm implemented by [40]. The implementation of the filter was adapted from the FreeIMU library described in [41]. The angle measurement follows the flowchart represented in figure 4.11.

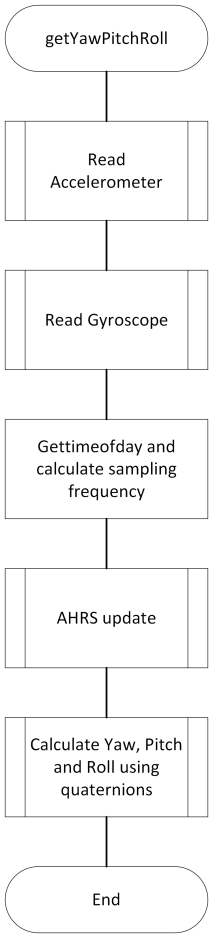

Figure 4.11: Flowchart for reading yaw, pitch and roll from the IMU

The class developed to interface with the temperature sensor is represented in figure 4.12.

```
class TemperaturaI2C
{
    private:
        uint16_t K4;
        uint16_t K3;
        uint16_t K2;
        uint16_t K1;
        uint16_t K0;
        BlackLib::BlackI2C TempSensor;
    public:
        //Temperatura I2C constructor-initializes calibration parameters
        TemperaturaI2C();
        float readTemperature();
        void close();
        ~TemperaturaI2C();
};
```

Figure 4.12: Temperature class

This implementation is based on the Arduino example available at [42]. As shown in figure 4.13, during initialization, the calibration parameters are read and stored in the private variables K0, K1, K2, K3 and K4 that are used to calculate the temperature as described in the datasheet.

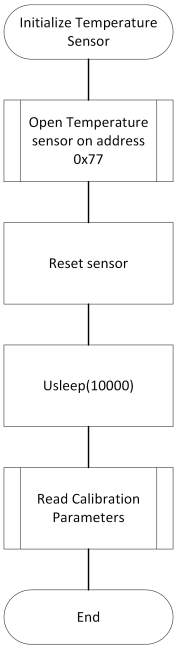

Figure 4.13: Flowchart representing the initialization of the temperature sensor

In the readTemperature method (figure 4.14), the conversion is started and, when available, is used to calculate the temperature as described in the datasheet [43].

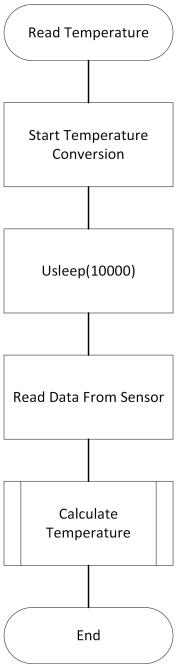

Figure 4.14: Flowchart for reading temperature values from the temperature sensor

The class to obtain values from the pressure sensor is implemented as shown in figure 4.15.

```
class PressureI2C
{
    private:
        uint8_t crcRead;
        uint16_t n_prom[8];
        //uint16_t fluidDensity=1029;//defined as sea water density
        uint16_t fluidDensity=1000;//defined as water density
        BlackLib::BlackI2C PressureSensor;
    public:
        float InitPressure=0;
        PressureI2C();
        unsigned char crc4(uint16_t prom[]); // n_prom defined as 8x unsigned int (n_prom[8])
        float readPressure();
        float calcDepth(float pressure);
        uint16_t setFluidDensity(uint16_t fluidDensity);
        void close();
        ~PressureI2C();
};
```

Figure 4.15: Pressure class

The Pressure class based on the Arduino implementation available at [42]. This class has a method to initialize the pressure sensor, reading the parameters from the sensor that are used for pressure calculation as shown in figure 4.16.

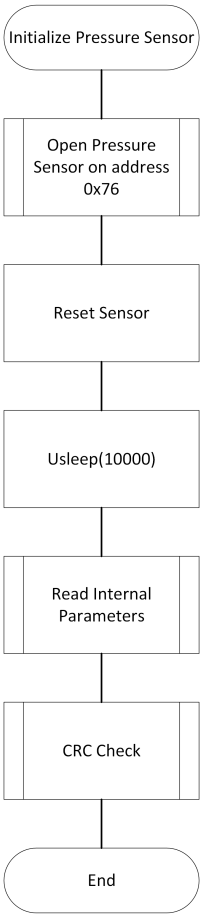

Figure 4.16: Flowchart describing the initialization of the pressure sensor

The pressure reading (figure 4.17) is started and, when finished, is used to calculate the pressure using the method described in the datasheet [44]. It is also implemented a CRC method, as described in the datasheet, to detect possible errors during internal parameters reading.

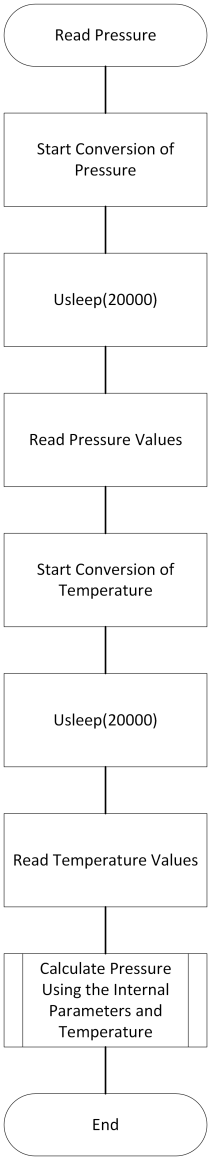

Figure 4.17: Flowchart for reading values from the pressure sensor

4.3.1.2 RS-232

The Intelligent Battery Power System uses an RS-232 interface to communicate battery status. To communicate using RS-232 it is used one of the UART ports available in the BeagleBone Black configured with 19200 baudrate, no parity bit, one stop bit and 8 data bits as indicated in the IBPS software manual [45]. The class developed to interface with the battery manager is defined as shown in figure 4.18.

```
class BatteryManager
{
    private:
        BlackLib::BlackUART Uart4;
        uint8_t checksum(string data);
    public:
        BatteryManager();
        bool info(unsigned int &percentage,unsigned int &timeToEmpty);
        void close();
        ~BatteryManager();
};
```

Figure 4.18: Battery Manager class

This class allows to read the remaining percentage and remaining time in minutes from the batteries connected to the Intelligent Battery Power System. The data transmitted from the IBPS is defined accordingly to the Smart Battery Data Specification [46]. As shown in figure 4.19, when the class constructor is called, the UART port is opened using the configurations mentioned above and it is sent two bytes to start receiving the hexadecimal data from the IBPS.

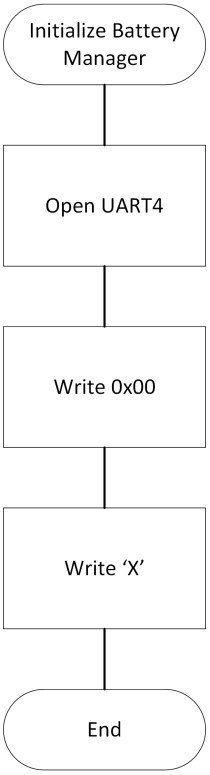

Figure 4.19: Flowchart describing battery manager initialization

To obtain percentage and remaining time, it is read the device file of the UART port. When the system data message is received (message identified with "$S") the percentage and remaining time is saved (4.20).

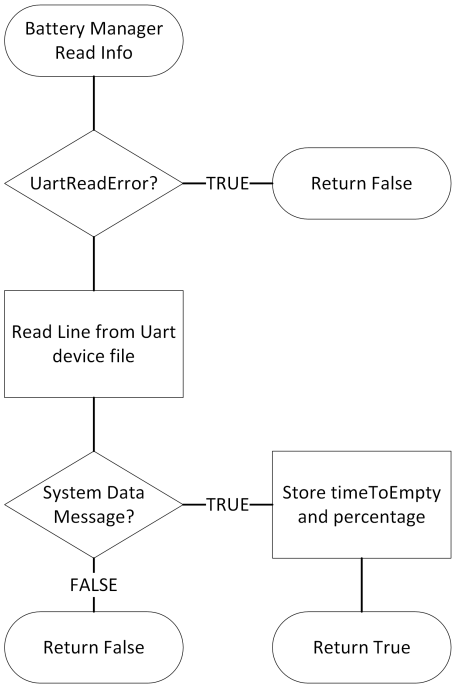

Figure 4.20: Flowchart for reading info from battery manager

4.3.1.3 USB

The USB is used to communicate with the GPS and Wi-fi.

For using the USB Wi-fi module, it was necessary to disable the onboard Wi-fi module.

The GPS outputs data using the NMEA 0183 standard [47]. To interface with the USB, it is used its device file where the data is received from the GPS. The class developed is defined in figure 4.21.

```
class GPS
{
    private:
        fstream fd;
        vector<string> splitStringByComma(string input);
    public:
        float latitude;
        float longitude;
        time_t UTC;//time and date since Epoch
        string speed;//Knots
        string heading;
        GPS();
        void close();
        bool getPosition();
        ~GPS();
};
```

Figure 4.21: GPS class

The GPS constructor opens the respective device file.

To get a position fix is used the RMC message that contains the minimum recommended data wich includes time, status (indicates if fix is valid), latitude and longitude, speed over the ground,

track angle in degrees, date and magnetic variation [47]. As shown in figure 4.22, the data received from the GPS is only stored if the fix is valid.

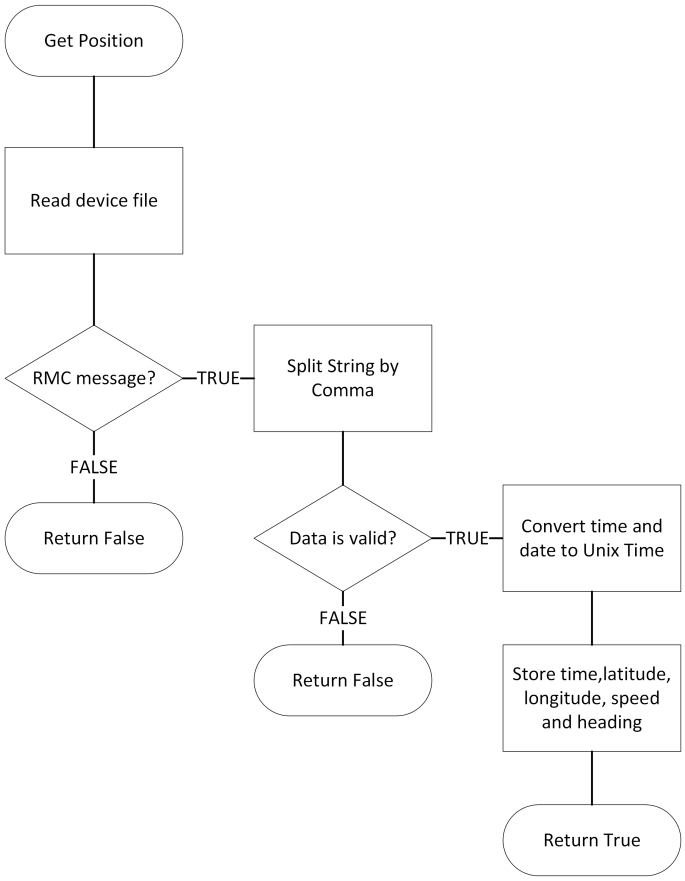

Figure 4.22: Flowchart for reading position from GPS

4.3.1.4 PWM

The thrusters are controlled using PWM signal sent to the ESC's. The Blacklib C++ library provides a class to use PWM on the BeagleBone Black. However, this class does not work on the current version of the kernel of the BBB because the location of the device files is different than in the previous versions. Therefore it was developed a class (figure 4.23) to interface with the available PWM pins based on this example [48] that loads the PWM on Beagle Bone Black.

```
class PWM
{
    private:
            int id1;
            int id2;
            bool state=false;//true if PWM enabled
            std::string ocpPath;
            std::string pwmPath;
            double currentPeriod=0;//nanoseconds
            double currentDutyCycle=0;//nanoseconds
    public:
            //PWM constructor-initializes PWM on Pid1_id2
            PWM(int id1, int id2);
            //function to set period in nanoseconds
            bool setPeriod(unsigned int period);
            //function to set duty cycle in nanoseconds
            bool setDutyCycle(unsigned int dutyCycle);
            //function to enable pwm
            bool enablePWM();
            //function to disable pwm
            bool disablePWM();
            //function to read period from file
            std::string readPeriod();
            //function to read duty cycle value from file
            std::string readDutyCycle();
            //function to get dutycycle
            double getDutyCycle();
            //function to get Period
            double getPeriod();
            std::string searchDirectory(std::string searchIn, std::string searchThis);
            std::string executeCommand(std::string command);
            virtual ~PWM();
};
```

Figure 4.23: PWM Class

The class constructor (figure 4.24) configures the pins for PWM use and saves the path for the device files that allow to enable PWM, change period and duty cycle.

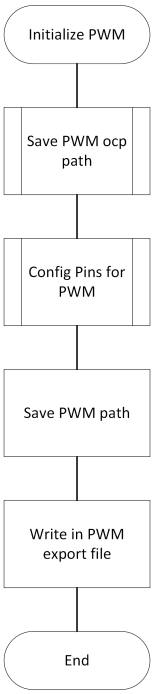

Figure 4.24: Flowchart for initializing PWM

Each PWM pin has three device files that can be changed using this class:

- period to change the period value in nanoseconds;
- duty_cycle to change duty cycle value in nanoseconds;
- enable that if set to one enables the PWM or if set to zero disables the PWM.

The setPeriod, setDutyCycle, enablePWM and disablePWM are methods that allow to write on the proper device file.

4.3.1.5 ADC

The ADC is used to read the voltage value from the TMP36. To read the ADC, it was implemented a class (figure 4.25) that in the constructor gets the path to the device file of the ADC and in the method getValue returns the voltage value in the pin.

```
class ADC{
    private:
        string ADCpath;
    public:
        ADC(string adcname);
        float getValue();//returns value in Volts

};
```

Figure 4.25: ADC class

This class allows to obtain the value read by the ADC and convert it to a voltage value by doing:

$$V = ADCvalue * \frac{1.8}{4095} \tag{4.1}$$

The value 1.8 is the maximum voltage allowed in the ADC pins and 4095 is the resolution of the ADC.

This voltage value is then converted to the temperature value using:

$$Temperature = \frac{V - 0.5}{0.01} \tag{4.2}$$

4.3.1.6 ESC firmware

The used ESC comes preloaded with the SimonK firmware [49]. However, the default parameters used in this firmware does not allow to drive the thrusters in normal and reverse mode.

Therefore, it was flashed the firmware available at [50]. This firmware provided by the thruster manufacturer is a modified version of the above that allows normal and reverse mode and it is configured with the motor parameters.

4.3.1.7 Data sharing between processes

The data is made available to all the processes using shared memory. This was used since it is the fastest inter process communication (IPC) method [51].

Shared memory allows to access the same memory by different processes and making changes in memory transparent to all processes that are attached to it. Since it is available to different processes, it is necessary to implement a synchronization mechanism between processes to prevent race conditions. The synchronization method used is described in section 4.3.1.8.

It is used two blocks of shared memory: data and timestamps (table 4.2). In the memory allocated for data, it is available the data read from the different sensors present in the system while in timestamp is available the time at which the data was acquired.

Table 4.2: Shared memory

Position	Data	Timestamp
0	External Temperature	Timestamp Temperature
1	Pressure	Timestamp Pressure
2	Depth	Timestamp Depth
3	Yaw	Timestamp Yaw
4	Pitch	Timestamp Pitch
5	Roll	Timestamp Roll
6	Internal Temperature	Timestamp Internal Temperature
7	Batteries Average Charge Percentage	Timestamp Batteries Percentage
8	Minutes to Empty Batteries	Timestamp Minutes to Empty
9	Latitude	Timestamp Latitude
10	Longitude	Timestamp Longitude

4.3.1.8 Synchronization between processes

Taking into account that it was used shared memory and that the kernel does not synchronize access to shared memory [51], it is necessary to use a synchronization mechanism.

To synchronize accesses to this resource by different processes it was used Inter Process (IPC) semaphores. Semaphores are a synchronization mechanism that are used to control access to a resource. Every time that a process needs to access the shared memory, it has to lock the semaphore or wait until the resource is available. After accessing the shared memory the process must free the resource.

4.3.2 Mission Plan

To read the mission plan it was developed two classes (Mission and Actions), available in the Mission library (figure 4.26).

The Actions class defines the parameters of each command read from the file and the time of start and time of finish.

The Mission class has a method to read the config.ini file that inserts each command in a vector of Actions. This vector will store the commands read from the file that will be executed by the vehicle.

```
class Actions{
    public:
        std::string type;
        float depth;//used in gotoz, surface and hover
        float pitch;//used in gotoz
        float roll;//used in gotoz
        int timeout;//timeout for maneuvers
        int wait;//duration of wait in seconds
        int duration;//used in hover action
        float ascent_rate;//ascent rate for surface maneuver
        float descent_rate;
        time_t start;
        time_t finish;
        Actions(std::string type, float depth, float pitch, float roll,
                int timeout, int wait, int duration, float ascent_rate, float descent_rate);
};

class Mission{
    public:
        vector<Actions> actionScript;
        float homeLatitude;
        float homeLongitude;
        bool readFile(string file);

};
```

Figure 4.26: Mission and Actions classes

To plan a mission on the vehicle, the user must create a mission file with the name config.ini (as shown in figure 4.27). This file must be started with [mission] and the following commands are available:

- gotoz=<depth> <pitch> <roll> S <descent_rate> T <timeout> — command that allows to profile to a certain depth (in meters) controlling the pitch, roll (in degrees) and descent rate (m/s).

- surface=<depth> S <ascent_rate> T <timeout> — command to make the profiler surface with a defined ascent rate (m/s).

- wait=<seconds> — command to wait for a determined period of seconds. Allows to set intervals between different maneuvers. Since the vehicle has a positive buoyancy, the vehicle waits at surface.

- hover= <depth> D <duration> T <timeout> — command to hover at fixed depth during the number of seconds defined in duration.

For each command (except the wait command that stops the profiler), a timeout must be set to, in case the profiler does not complete the maneuver, this is interrupted and the next command is executed.

```
[mission]
gotoz=100 20 20 S 1 T 500
surface=0 S 0.1 T 1000
wait=1000
hover=50 D 100 T 500
surface=0 S 0.5 T 500
wait=3000
gotoz=120 0 0 S 0.5 T 1000
surface=0 S 1 T 300
```

Figure 4.27: Example of a possible mission configuration

4.3.3 Software Architecture

As stated in section 3.4 the processing unit runs 3 processes that will be described in this section.

4.3.3.1 Mission Control and Supervision

The Mission Control and Supervision is the main process and is responsible for creating the other processes.

At the beginning of execution, this process loads the device tree overlay cape-universal that allows to configure the pins available in the BeagleBone Black. The device tree overlay allows to modify the system during runtime. The device tree is loaded by the Capemgr implemented in the kernel that allows to allocate appropriate resources on kernel [52]. Then, the pins P9.13 and P9.11 are configured as UART pins. The other pins used are configured by default.

After pin configuration, the mission file is passed as an argument when running the program is read and the commands are stored in a vector as described in section 4.3.2.

After the mission plan is read, the shared memory blocks described in section 4.3.1.7 are created as well as the semaphores for data synchronization (section 4.3.1.8). The other processes described in 3.4 are then created.

The ESCs are then initialized by sending a stop signal (PWM with duty cycle of 1500 microseconds) during 2 seconds (as indicated by the manufacturer) and the process is stopped waiting for a GPS valid fix. The GPS fix is used to set the start position of the profiler and to synchronize the date and time of the processing unit with the GPS.

After the start position is defined, it is set a timer of 100 milliseconds. This timer sends a signal to the process to run the control cycle every 100 milliseconds.

The control cycle implements the controllers described in chapter 6 for each maneuver and ensures that every command is executed in time or, in case of timeout, the next command is executed.

In the end of the mission, the commands executed are logged in the mission plan log as described in section 4.3.6, the remaining processes are terminated and the semaphores and shared memory are eliminated.

4.3.3.2 Sensors

The sensors process is responsible for reading data from the sensors available and publish the data acquired in the shared memory available for the other processes.

This process starts by attaching to the shared memory block created by the Mission Control and Supervision process and creates objects to read data from the available sensors.

After this, a timer is set using signals that delivers a signal to the process every 100 miliseconds. Each time a signal is delivered, all sensors must be read using the objects created above. The read data and their respective timestamps are then passed to the shared memory block using the semaphores to lock access to this shared resource.

In the first time that the vehicle obtains a valid position fix, this process sets the time and date of the system using the time and date received in the GPS message to ensure proper synchronization of the vehicle's processing unit. After setting the time, this process sends a signal to the main process to resume execution.

At the end of the execution, the process detaches itself from the shared memory.

4.3.4 Logging Process

As noted in section 3.4, writing the log to a file can be a slow task, since it involves writing to a disk or other slow external device. Therefore, the log process is responsible to write the data read from the sensors in a file without delaying the execution of the program.

To ensure that the data and timestamps of the files are correct, it was implemented a class (figure 4.28) that has as members an ID that identifies the type of data, the data read and a timestamp as a timeval struct. This structure contains two members: the number of seconds in Unix time and the microseconds elapsed, allowing a more detailed log.

```
class DataTime
{
    public:
        char ID;
        float data;
        struct timeval timestamp;
        DataTime();
        DataTime(char ID,float data, struct timeval timestamp);

};
```

Figure 4.28: Data Time class

This process starts by attaching to the shared memory and creating the file where the log will be written. Then it sets a timer that sends a signal to the process each 250 miliseconds. Everytime the signal is received, the shared memory is accessed and the data and timestamps are read. These are stored in a queue that ensures that the order of the data read is correct since a queue is a First In First Out (FIFO) data type, meaning that the first elements added to the queue will be the first to be removed. This guarantees that the data is written on the log in the correct order. Each type of data has its own queue.

The log is written in background, removing the front element of each data queue, following the order described later in section 4.3.5.1.

4.3.5 Log

During the mission, the profiler stores two different log files: a data log that contains data read from all the sensors during the execution of the mission, and a mission log that saves the main mission events: the type of command, the time of start and the time of end.

4.3.5.1 Data Log

The log that contains the data is named DataLog<time>.txt where <time> corresponds to the number of seconds since 00:00 hours, Jan 1, 1970 UTC.

The first line of this file describes the order of the parameters (ID, DATA ,TIME SECONDS, TIME USECONDS). Each line of the file corresponds to the data available from the sensor identified by the ID (table 4.3) and contains the data and time at which the data was read.

Table 4.3: Data log

ID	Data	
1	External Temperature	
2	Pressure	
3	Depth	
4	Yaw	
5	Pitch	
6	Roll	
7	Internal Temperature	
8	Batteries Average Charge Percentage	
9	Minutes to Empty Batteries	
10	Latitude	Longitude

Figure 4.29 presents an excerpt of a data log from one of the tests performed.

```
ID,DATA,TIME SECONDS,TIME USECONDS
1,21.4921,1495706353,646049
2,1030.29,1495706353,646050
3,0.0684705,1495706353,646051
4,-0.000141879,1495706353,646053
5,0.0455052,1495706353,646054
6,-0.0106308,1495706353,646055
7,33.8242,1495706353,646056
8,0,1495706353,646057
9,0,1495706353,646058
10,0,0,1495706353,646060
1,21.4921,1495706353,946189
2,1032.74,1495706353,946191
3,0.0934804,1495706353,946192
4,0.0019129,1495706353,946193
5,0.185223,1495706353,946194
6,-0.0722474,1495706353,946196
7,33.7802,1495706353,946197
8,10,1495706353,946198
9,187,1495706353,946199
10,0,0,1495706353,946200
```

Figure 4.29: Example of a data log

4.3.6 Mission Log

The mission log is available in the file MissionLog<time>.txt where <time> corresponds to the number of seconds at the end of the execution since 00:00 hours, Jan 1, 1970 UTC.

The mission log is written at the end of a complete mission and stores in the file the commands read from the mission configuration file, the time of start of each command and the time of finish of each command. Meanwhile, during mission execution, the command data saved in the Actions vector is updated. Figure 4.30, presents an example of a mission log.

```
ID,Type, Start, Finish
1,wait,1494955832,1494955837
2,gotoz,1494955837,1494955838
3,surface,1494955838,1494955838
4,wait,1494955838,1494955839
5,hover,1494955839,1494955840
6,surface,1494955840,1494955841
7,wait,1494955841,1494955842
8,gotoz,1494955842,1494955843
```

Figure 4.30: Example of a mission log

Chapter 5

Vehicle Model and Simulation

This chapter, it will present a 6 DOF model of an underwater vehicle following the approach presented in [53]. This model will be used to analyze the stability of the system and to develop and test the performance of the controllers.

The model described takes into account the kinematics and dynamics of the vehicle in the 6 degrees of freedom. The motion of an underwater vehicle is described by 6 independent coordinates that are necessary to determine the position and orientation of a rigid body. The first three coordinates describe the position of the vehicle along the x ,y and z axis, while the last three coordinates describe the orientation of the vehicle (ϕ,θ,ψ).

5.1 Coordinate Frames

To properly analyze the system, it is convenient to define two coordinate frames: a moving coordinate frame fixed on the body (body-fixed frame) and an inertial reference frame (earth fixed frame). The notation used is as presented in [53] and is explained in table 5.1. The motion of a 6 DOF vehicles can be described by the following vectors:

$$\eta = [\eta_1^T, \eta_2^T]; \qquad \eta_1^T = [x, y, z]^T \qquad \eta_2^T = [\phi, \theta, \psi]^T \tag{5.1}$$

$$\nu = [\nu_1^T, \nu_2^T]; \qquad \nu_1^T = [u, v, w]^T \qquad \nu_2^T = [p, q, r]^T \tag{5.2}$$

$$\tau = [\tau_1^T, \tau_2^T]; \qquad \tau_1^T = [X, Y, Z]^T \qquad \tau_2^T = [K, M, N]^T \tag{5.3}$$

In 5.1, η defines the position and orientation of the vehicle with coordinates in the earth fixed frame. In 5.2 ν defines the linear and angular velocities referenced to the body fixed frame and in 5.3, τ defines the forces and moments applied to the vehicle.

Table 5.1: Notation used for marine vehicles according to [53]

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

5.2 6 DOF Model of the Vehicle

Again, according to [53], the motion of an underwater vehicle can be described in the body fixed frame using a matrix representation as follows:

$$M\dot{\nu} + C(\nu)\nu + D(\nu)\nu + g(\eta) = \tau \tag{5.4}$$

$$\dot{\eta} = J(\eta)\nu \tag{5.5}$$

where M represents the inertia matrix, $C(\nu)$ is the matrix of Coriolis and centripetal terms, $D(\nu)$ is the damping matrix, $g(\eta)$ is the vector of gravitational forces and moments and τ is the vector of control inputs. Since, M and $C(\nu)$ include added mass terms, these matrices can be written as:

$$M = M_{RB} + M_A \qquad C = C_{RB} + C_A \tag{5.6}$$

The equation 5.5 gives the vehicle flight path relative to the earth fixed frame using the transformation matrix $J(\eta)$ that can be defined as [53]:

$$J(\eta) = \begin{bmatrix} c\psi c\theta & -s\psi c\phi + c\psi s\theta s\phi & s\psi s\phi + c\psi c\phi s\theta & 0 & 0 & 0 \\ s\psi c\theta & c\psi c\phi + s\phi s\theta s\psi & -c\psi s\phi + s\theta s\psi c\phi & 0 & 0 & 0 \\ -s\theta & c\theta s\phi & c\theta c\phi & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & s\phi t\theta & c\phi t\theta \\ 0 & 0 & 0 & 0 & c\phi & -s\phi \\ 0 & 0 & 0 & 0 & s\phi/c\theta & c\psi/c\theta \end{bmatrix} \tag{5.7}$$

Where s $\cdot$ = sin $\cdot$, c $\cdot$ = cos $\cdot$ and t $\cdot$ = tan $\cdot$.

5.2.1 6 DOF Rigid Body Equations of Motion

As seen in [53], the rigid body equations of motion can be represented as follows:

$$\begin{aligned}
m[\dot{u} - vr + wq - x_g(q^2 + r^2) + y_g(pq - \dot{r}) + z_g(pr + \dot{q})] &= X \\
m[\dot{v} - wp + ur - y_g(r^2 + p^2) + z_g(qr - \dot{p}) + x_g(qp + \dot{r})] &= Y \\
m[\dot{w} - uq + vp - z_g(p^2 + q^2) + x_g(rp - \dot{q}) + y_g(rq + \dot{p})] &= Z \\
I_x\dot{p} + (I_z - I_y)qr - (\dot{r} + pq)I_{xz} + (r^2 - q^2)I_{yz} + (pr - \dot{q})I_{xy} & \\
+m[y_g(\dot{w} - uq + vp) - z_g(\dot{v} - wp + ur)] &= K \\
I_y\dot{q} + (I_x - I_z)rp - (\dot{p} + qr)I_{xy} + (p^2 - r^2)I_{zx} + (qp - \dot{r})I_{yz} & \\
+m[z_g(\dot{u} - vr + wq) - x_g(\dot{w} - uq + vp)] &= M \\
I_z\dot{r} + (I_y - I_x)pq - (\dot{q} + rp)I_{yz} + (q^2 - p^2)I_{xy} + (rq - \dot{p})I_{zx} & \\
+m[x_g(\dot{v} - wp + ur) - y_g(\dot{u} - vr + wq)] &= N
\end{aligned} \tag{5.8}$$

In equation 5.8, m represents the mass of the vehicle and (x_g, y_g, z_g) represents the coordinates of the centre of gravity of the rigid body. These equations can be represented in a vectorial form as:

$$M_{RB}\dot{\nu} + C_{RB}(\nu)\nu = \tau \tag{5.9}$$

Where M_{RB} can be defined as:

$$M_{RB} = \begin{bmatrix}
m & 0 & 0 & 0 & mz_g & -my_g \\
0 & m & 0 & -mz_g & 0 & mx_g \\
0 & 0 & m & my_g & -mx_g & 0 \\
0 & -mz_g & my_g & I_x & -I_{xy} & -I_{xz} \\
mz_g & 0 & -mx_g & -I_{yx} & I_y & -I_{yz} \\
-my_g & m_{xg} & 0 & -I_{zx} & -I_{xy} & I_z
\end{bmatrix} \tag{5.10}$$

Defining the origin of the body-fixed frame in the center of gravity of the body and neglecting I_{xy}, I_{xz} and I_{yz} since these values are small compared with the moments of Inertia I_x, I_y and I_z, results in a diagonal matrix M_{RB}.

According to [53], the C_{RB} matrix consists of Coriolis vector term $w \times v$ and the centripetal vector term $w \times (w \times r_g)$. This matrix can be written as:

$$C_{RB} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ -m(y_g q + z_g r) & m(y_g p + w) & m(z_g p - v) \\ m(x_g q - w) & -m(z_g r + x_g p) & m(z_g q + u) \\ m(x_g r + v) & m(y_g r - u) & -m(x_g p + y_g q) \end{bmatrix} \begin{bmatrix} m(y_g q + z_g r) & -m(x_g q - w) & -m(x_g r + v) \\ -m(y_g p + w) & m(z_g r + x_g p) & -m(y_g r - u) \\ -m(z_g p - v) & -m(z_g q + u) & m(x_g p + y_g q) \\ 0 & -I_{yz} q - I_{xz} p + I_z r & I_{yz} r + I_{xy} p - I_y q \\ I_{yz} q + I_{xz} p - I_z r & 0 & -I_{xz} r - I_{xy} q + I_x p \\ -I_{yz} r - I_{xy} p + I_y q & I_{xz} r + I_{xy} q - I_x p & 0 \end{bmatrix} \tag{5.11}$$

5.2.2 Hydrodynamic Forces and Moments

As described in [54], the motion of a rigid body in an empty space does not take into account the presence of hydrodynamics forces and moments caused by the presence of the fluid. This forces are divided into radiation-induced forces, environmental disturbances and restoring forces due to gravity and buoyancy [54].

5.2.2.1 Added Mass

In a fluid environment, it has to be considered its additional inertia when the vehicle moves. The added mass corresponds to the reaction force exerted by the surrounding fluid when accelerating the body [54].

The added inertia matrix (M_A) is defined as:

$$M_A = \begin{bmatrix} X_{\dot{u}} & X_{\dot{v}} & X_{\dot{w}} & X_{\dot{p}} & X_{\dot{q}} & X_{\dot{r}} \\ Y_{\dot{u}} & Y_{\dot{v}} & Y_{\dot{w}} & Y_{\dot{p}} & Y_{\dot{q}} & Y_{\dot{r}} \\ Z_{\dot{u}} & Z_{\dot{v}} & Z_{\dot{w}} & Z_{\dot{p}} & Z_{\dot{q}} & Z_{\dot{r}} \\ K_{\dot{u}} & K_{\dot{v}} & K_{\dot{w}} & K_{\dot{p}} & K_{\dot{q}} & K_{\dot{r}} \\ M_{\dot{u}} & M_{\dot{v}} & M_{\dot{w}} & M_{\dot{p}} & M_{\dot{q}} & M_{\dot{r}} \\ N_{\dot{u}} & N_{\dot{v}} & N_{\dot{w}} & N_{\dot{p}} & N_{\dot{q}} & N_{\dot{r}} \end{bmatrix} \tag{5.12}$$

In M_A, for instance, the hydrodynamic force along x_b due to the linear acceleration in the x_b direction is defined as:

$$X_A := -X_{\dot{u}} \dot{u} \qquad ,where \qquad X_{\dot{u}} := \frac{\partial X}{\partial \dot{u}} \tag{5.13}$$

The coefficients of the matrix M_A can be obtained by applying strip theory [53]. This theory consists in dividing the submerged part of the vehicle into a number of strips. For each strip, its possible to determine the two-dimensional hydrodynamics coefficients and, taking into account the length of the vehicle, obtain the three-dimensional hydrodynamics coefficients. As seen in [53], M_A can be simplified taking advantage of the symmetry plans of the body. Considering that the vehicle has top-bottom and port-starboard symmetry, M_A is equal to:

$$M_A = \begin{bmatrix} m_{11} & 0 & 0 & 0 & m_{15} & 0 \\ 0 & m_{22} & 0 & m_{24} & 0 & 0 \\ 0 & 0 & m_{33} & 0 & 0 & 0 \\ 0 & m_{42} & 0 & m_{44} & 0 & 0 \\ m_{51} & 0 & 0 & 0 & m_{55} & 0 \\ 0 & 0 & 0 & 0 & 0 & m_{66} \end{bmatrix} \tag{5.14}$$

The added mass also makes an added Coriolis and centripetal contribution. The Coriolis and centripetal matrix ($C_A(\nu)$) can be represented as:

$$C_A = \begin{bmatrix} 0 & 0 & 0 & 0 & -a_3 & a_2 \\ 0 & 0 & 0 & a_3 & 0 & -a_1 \\ 0 & 0 & 0 & -a_2 & a_1 & 0 \\ 0 & -a_3 & a_2 & 0 & -b_3 & b_2 \\ a_3 & 0 & -a_1 & b_3 & 0 & -b_1 \\ -a_2 & a_1 & 0 & -b_2 & b_1 & 0 \end{bmatrix} \tag{5.15}$$

where a_1, a_2, a_3, b_1, b_2 and b_3 can be defined as:

$$\begin{aligned} a_1 &= X_{\dot{u}}u + X_{\dot{v}}v + X_{\dot{w}}w + X_{\dot{p}}p + X_{\dot{q}}q + X_{\dot{r}}r \\ a_2 &= X_{\dot{v}}u + Y_{\dot{v}}v + Y_{\dot{w}}w + Y_{\dot{p}}p + Y_{\dot{q}}q + Y_{\dot{r}}r \\ a_3 &= X_{\dot{w}}u + Y_{\dot{w}}v + Z_{\dot{w}}w + Z_{\dot{p}}p + Z_{\dot{q}}q + Z_{\dot{r}}r \\ b_1 &= X_{\dot{p}}u + Y_{\dot{p}}v + Z_{\dot{p}}w + K_{\dot{p}}p + K_{\dot{q}}q + K_{\dot{r}}r \\ b_2 &= X_{\dot{q}}u + Y_{\dot{q}}v + Z_{\dot{q}}w + K_{\dot{q}}p + M_{\dot{q}}q + M_{\dot{r}}r \\ b_3 &= X_{\dot{r}}u + Y_{\dot{r}}v + Z_{\dot{r}}w + K_{\dot{r}}p + M_{\dot{r}}q + N_{\dot{r}}r \end{aligned} \tag{5.16}$$

5.2.2.2 Hydrodynamic Damping

The hydrodynamic damping for marine vehicles is mainly caused by potential damping, skin friction, wave drift damping, vortex shedding damping and viscous damping.

Potential damping and wave drift damping are typically negligible for underwater vehicles [54].

Skin friction is caused by the boundary layers of the vehicle and can affect its low-frequency motion [54]. In addition to linear skin friction, at high frequencies there will be a contribution due to turbulent boundary layers [53].

As seen in [55], the viscous damping force due to vortex shedding can be modeled as:

$$f(U) = -\frac{1}{2}\rho C_D(Rn)A|U|U \tag{5.17}$$

In 5.17, U is the velocity of the vehicle, A is the projected cross-sectional area, $C_D(Rn)$ is the drag-coefficient based on the representative area and ρ is the water density. The drag coefficient $C_D(Rn)$ depends on the Reynold's Number:

$$R_n = \frac{Ul}{\nu} \tag{5.18}$$

where U is the operation speed of the vehicle, l is the characteristic length and ν is the fluid kinematic viscosity, which for seawater at 15°C, gives a value of 1.190×10^{-6}m/s [56].

Hydrodynamic damping in underwater vehicles moving in 6 DOF at high speed is coupled and highly non-linear. To simplify the model of an underwater vehicle, some assumptions are typically made (as seen in [53, 57, 58]). Typically, is assumed that linear and coupled terms are negligible and that terms greater than second order are neglected too, assuming that their effects are small. However, since this vehicle is projected to move at low speeds, the linear terms of drag need to be considered as they are not negligible. Therefore, the linear and quadratic term of drag are considered yielding the damping matrix as follows:

$$D(\nu) = D_l + D_q(\nu) \tag{5.19}$$

where D_l represents the linear drag matrix and D_q the quadratic drag matrix. The linear drag, in this case, can be written as:

$$D_l = -\begin{bmatrix} X_u & 0 & 0 & 0 & X_q & 0 \\ 0 & Y_v & 0 & Y_p & 0 & 0 \\ 0 & 0 & Z_w & 0 & 0 & 0 \\ 0 & K_v & 0 & K_p & 0 & 0 \\ M_u & 0 & 0 & 0 & M_q & 0 \\ 0 & 0 & 0 & 0 & 0 & N_r \end{bmatrix} \tag{5.20}$$

For this vehicle, the quadratic drag matrix is expressed as:

$$D_m(\nu) = -\begin{bmatrix} X_{u|u|}|u| & 0 & 0 & 0 & X_{q|q|}|q| & 0 \\ 0 & Y_{v|v|}|v| & 0 & Y_{p|p|}|p| & 0 & 0 \\ 0 & 0 & Z_{w|w|}|w| & 0 & 0 & 0 \\ 0 & K_{v|v|}|v| & 0 & K_{p|p|}|p| & 0 & 0 \\ M_{u|u|}|u| & 0 & 0 & 0 & M_{q|q|}|q| & 0 \\ 0 & 0 & 0 & 0 & 0 & N_{r|r|}|r| \end{bmatrix} \tag{5.21}$$

5.2.2.3 Restoring Forces and Moments

The gravitational force and the buoyancy are called restoring forces. The gravitational force acts through the centre of gravity $r_g = [x_g, y_g, z_g]$ while the buoyancy acts through the centre of buoyancy $r_b = [x_b, y_b, z_b]$. The restoring forces acting through an underwater vehicle can be described as [53]:

$$g(\eta) = \begin{bmatrix} (W-B)s\theta \\ -(W-B)c\theta s\phi \\ -(W-B)c\theta c\phi \\ -(y_gW - y_bB)c\theta c\phi + (z_gW - z_bB)c\theta s\phi \\ (z_gW - z_bB)s\theta + (x_gW - x_bB)c\theta c\phi \\ -(x_gW - x_bB)c\theta s\phi - (y_gW - y_bB)s\theta \end{bmatrix} \tag{5.22}$$

5.3 Thrusters Mapping

The four independent thrusters provide the forces and moments to propel the vehicle (τ).

The thruster module can deliver force in the heave direction and produce torque in roll and pitch directions. As seen in [59], the forces and moments delivered by the thrusters can be modeled by:

$$\tau = T * f \tag{5.23}$$

where $T \in \mathbb{R}^{n \times n}$ is the thruster configuration matrix and f is the control force.

Considering the above mentioned, the T matrix resulted in:

$$T = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 \\ -l & -l & l & l \\ l & -l & l & -l \\ 0 & 0 & 0 & 0 \end{bmatrix} \tag{5.24}$$

In the matrix, l is the distance that separates the center of the thruster and the center of gravity.

The matrix f can be represented as $f = [M1\ M2\ M3\ M4]$ where M1, M2, M3 and M4 represent the forces produced by each thruster.

5.4 Vehicle Parameters and Coefficient Derivation

5.4.1 Vehicle Reference Frames

Considering the coordinate frame used by [53], typical underwater vehicle are oriented along the x-axis of the body fixed frame.

However, considering that the profiler is a vehicle used mainly for vertical movement, in this work, it is considered that the vehicle is aligned with the z-axis as seen in figure 5.1.

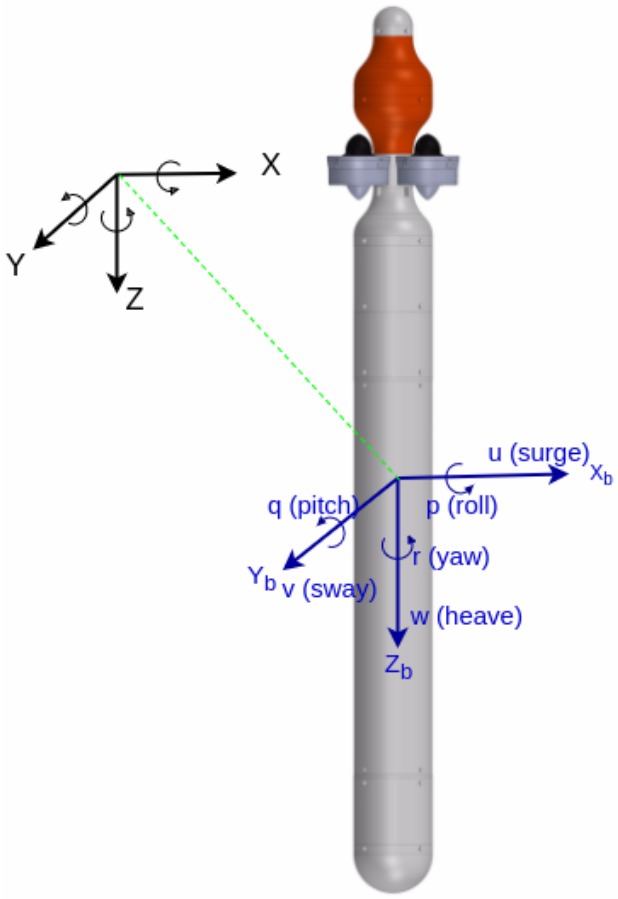

Figure 5.1: Autonomous Underwater Profiler reference frames

The vehicle has an hull radius of 6 cm and a length of 1.35 m.

5.4.2 Vehicle Weight and Buoyancy

One of the requirements defined in chapter 3 was that the vehicle must be lightweight (with a weight less than 30 Kg).

Following this obligation, the mass of the vehicle is 11.3 Kg. However, the adjustments made to the center of buoyancy and center of gravity are made using loads that change the weight of the profiler. Therefore, this weight of the vehicle changes according to the adjustments made to the centers of buoyancy and gravity. For the centers selected in section 5.4.3 the vehicle has a weight of 111 N.

Another requirement defined in chapter 3 was that the vehicle should have a positive buoyancy. This has a direct impact on power consumption because the greater the buoyancy selected, the

greater the force required to make the profiler go down. The vehicle buoyancy was defined as 120 g.

5.4.3 Center of Buoyancy and Center of Gravity

During mission, the center of gravity (CG) and buoyancy (CB) do not change.

The distance between CB and CG has an impact on the power consumption and the limits of the maneuvers allowed by the vehicle. From the matrix $g(\eta)$ presented in section 5.2.2.3, it is possible to conclude that the bigger the distance that separates CB and CG, the bigger the moment necessary to rotate the vehicle and, consequently, the power to rotate the body. This can also be seen as an advantage since it grants greater stability of the vehicle during completely vertical profiles.

To prove the impact of the distance between the CB and CG, two simulations were performed, considering two values for the distance: 3 cm e 5 cm, where the profiler was released from a starting point with an angle in roll of 90º. It is verified in figure 5.3 that the longer the distance between CB and CG, the faster the system responds to this perturbation recovering more quickly to the equilibrium point than in figure 5.2 were it was considered a smaller distance between CB and CG.

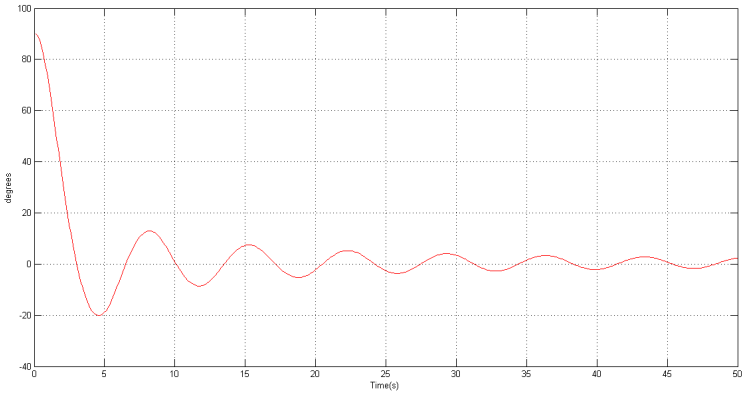

Figure 5.2: Simulation of the release of the profiler from a starting point of 90º degrees in pitch with 3 cm distance between CB and CG

In figure 5.2, it is possible to observe that the profiler takes approximately 3 seconds to reach 95 percent of the final value.

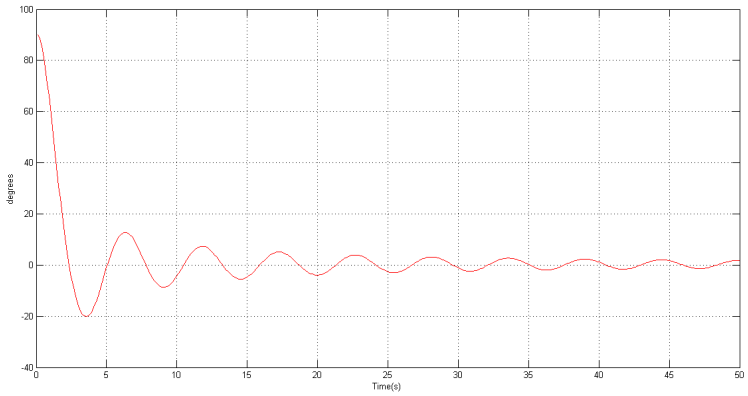

Figure 5.3: Simulation of the release of the profiler from a starting point of 90^{o} degrees in pitch with 5 cm distance between CB and CG

With a bigger distance between CB and CG, in figure 5.3, the vehicle reaches 95 percent of the final value in approximately 2 seconds proving that the bigger the distance between CB and CG, the faster the system rejects to disturbances.

However, for the same distances considered above and using two thrusters with the same forces (0.75 N) makes the profiler go to a position in roll smaller for the case where the distance between CB and CG is bigger (5.4).

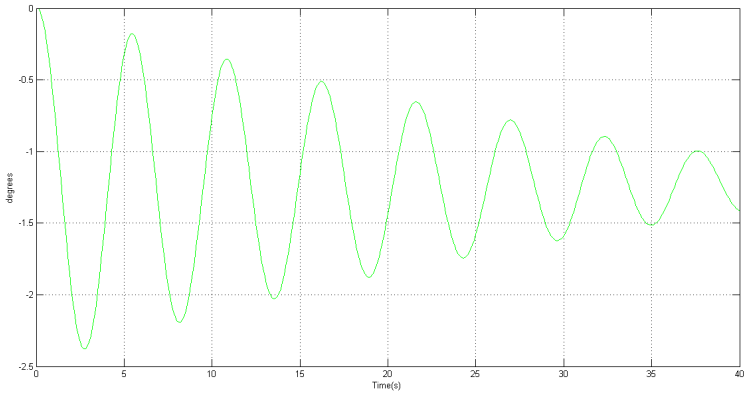

Figure 5.4: Simulation of the position of the profiler using two thrusters at 0.75 N each with 5 cm between CB and CG

Considering a smaller distance between CB and CG (5.2) and applying the same force to the thrusters, the angle in roll is bigger than in figure 5.4. As expected, it is thus confirmed that a smaller distance between CB and CG makes the forces necessary to rotate the vehicle lower.

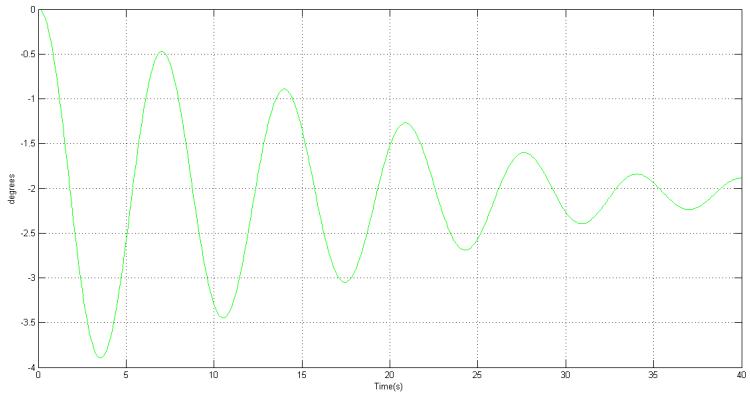

Figure 5.5: Simulation of the position of the profiler using two thrusters at 0.75 N each with 3 cm between CB and CG

Taking this into account, CB and CG should be chosen according to the mission that the profiler will execute. If the profiler will perform a mission where it is necessary to rotate in pitch and roll, the distance between CB and CG should be smaller. When the mission requires completely vertical movement, the distance between CB and CG should be larger to ensure greater stability.

For this work, it was considered a smaller distance between CB and CG to test the motion limits of the vehicle and to ensure that the developed controllers are capable of reject disturbances to the movement of vehicle, since considering a smaller distance between CB and CG makes the profiler more susceptible to perturbations.

The final values to the CB and CG are presented in the following section.

5.4.4 Determination of the Center of Buoyancy

The center of buoyancy of the vehicle was determined using experimental tests. The vehicle is released from a starting angle and the angle evolution is recorded over time. The obtained response is later compared with the simulation results that are used to select the center of buoyancy, which has a closer response to the observed during the tests.

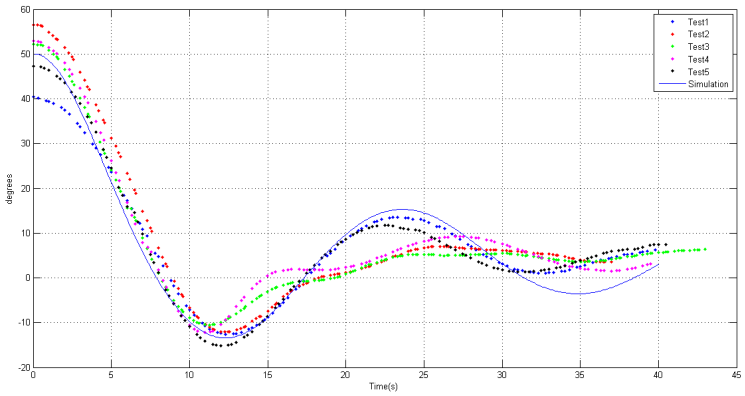

Figure 5.6: Results of the experimental tests and simulation

Figure 5.6 shows the results of the five tests performed and the results of the simulation. The profiler was released from a starting point between 40 and 60 degrees. The response was then used to, using the simulation, estimate the position of the CB and CG by adjusting the position of CB considered in the simulation and the linear drag coefficients. Table 5.2 lists the linear drag coefficients adjustment factors that are used to adjust the coefficients estimated in section 5.4.6.

Table 5.2: Linear drag adjustment factors

Parameter	Adjustment Factor
K_p	0.01
M_q	0.01

The values estimated for the center of buoyancy and center of mass of the vehicle are presented in table 5.3.

Table 5.3: Vehicle's center of buoyancy and center of gravity

Parameter	Center of Buoyancy	Center of Gravity	Units
x	0	-2.25e-4	m
y	0	0	m
z	-3e-3	0	m

5.4.5 Moments of Inertia

As seen in section 5.2.1, only the moments of Inertia I_x, I_y and I_z are considered. These values were estimated considering the vehicle as cylinder with radius 0.06 m and with 1.35 m of length using the formula defined in [60] obtaining the following values (table 5.4):

Table 5.4: Vehicle's moments of Inertia

Parameter	Value	Units
I_x	1.68	kg m^2
I_y	1.68	kg m^2
I_z	0.02	kg m^2

These values emphasizes the directional behavior of the vehicle because, as expected, the inertia in x and y is much bigger than in z.

5.4.6 Hydrodynamic Damping

As explained in section 5.2.2.2, the damping of an underwater vehicle moving in six degrees of freedom is coupled and non-linear.

Since drag coefficients estimation is mainly based on empirical expressions, the drag coefficients yielded by these expressions can be wrong [57] and they should be estimated based on experimental tests. However, the determination of the drag coefficients is beyond the scope of this work and therefore the results of the empirical expressions will be used for simulations of the vehicle as a first approach to the modeling of the vehicle.

Using expression 5.18, for an operating speed of 1 m/s and considering vehicle's length equal to 1.35 m yields R_n=1.008 $\times 10^6$ falling in the transition zone between laminar and turbulent flow. However, the hull of the profiler is not completely smooth which trips the flow around the vehicle into the turbulent regime (as seen in [57]).

The non-linear drag coefficient for drag in the Z direction can be calculated using:

$$Z_{w|w|} = -\frac{1}{2}\rho C_D(Rn)A \tag{5.25}$$

In the expression, ρ is the density of the surrounding fluid, A the vehicle frontal area, and C_D the axial drag coefficient of the vehicle. The axial drag coefficient can be calculated using the empirical formula given by [61] used in [57]:

$$C_D = \frac{c_{ss}\pi A_p}{A_f}\left[1 + 60\left(\frac{d}{l}\right)^3 + 0.0025\left(\frac{l}{d}\right)\right] \tag{5.26}$$

where c_{ss} is Schoenherr's value for flat plate skin friction, $A_p = ld$ is the vehicle plan area, and A_f is the vehicle frontal area. c_{ss} value is given by [62] and is 3.397×10^{-3}.

The method to estimate the other drag coefficients is analogous to strip theory, used to calculate added mass as explained in 5.2.2.1. The nonlinear crossflow drag coefficients are expressed as follows:

$$Y_{v|v|} = X_{u|u|} = -\frac{1}{2}\rho c_{dc}\int_{xt}^{xc} 2R(x)dx \tag{5.27}$$

$$M_{u|u|} = -K_{v|v|} = -\frac{1}{2}\rho c_{dc}\int_{xt}^{xc} 2xR(x)dx \tag{5.28}$$

$$Y_{p|p|} = -X_{q|q|} = -\frac{1}{2}\rho c_{dc} \int_{xt}^{xc} 2x|x|R(x)dx \tag{5.29}$$

$$M_{q|q|} = -K_{p|p|} = -\frac{1}{2}\rho c_{dc} \int_{xt}^{xc} 2x^3 R(x)dx \tag{5.30}$$

Above, ρ is the seawater density, c_{dc} is the drag coefficient of a cylinder and R(x) the hull radius. For the drag coefficient c_{dc}, it was considered the value given by [63] for a cylinder. For drag calculation the vehicle's profile was approximated using cylinders and semi-spheres as seen in figure 5.7.

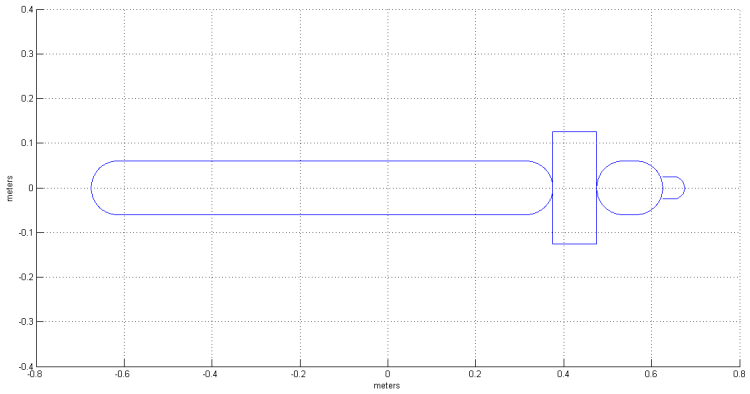

Figure 5.7: Approximated profile of the vehicle used for damping coefficients calculation

The integration limits are defined in table 5.5.

Table 5.5: Integration limits

Parameter	Value	Units
xc	- 0.675	m
xt	0.675	m
r	0.125	m

As [64] states, it is hard to estimate linear drag for underwater vehicles as the approximations available are typically used for surface vessels. As the identification of these parameters is not in the scope of this work, the drag coefficients were estimated for quadratic linear as linear coefficients. These were adjusted based on experimental tests like the one described in section 5.4.4. To ensure that the controllers designed performed correctly, they were tested multiplying the damping matrix by different scalar factors to test the robustness of the controller.

Drag coefficients can be found in table 5.6.

Table 5.6: Drag coefficients

Parameter	Value	Units
$X_{u\|u\|}$	-70.35	kg/m
$Y_{v\|v\|}$	-70.35	kg/m
$Z_{w\|w\|}$	-0.92	kg/m
$K_{p\|p\|}$	-4.95	$kg \cdot m^2/rad^2$
$M_{q\|q\|}$	-4.95	$kg \cdot m^2/rad^2$
$N_{r\|r\|}$	-0.01	$kg \cdot m^2/rad^2$
$M_{u\|u\|}$	0.57	kg
$K_{v\|v\|}$	-0.57	kg
$Y_{p\|p\|}$	0.02	$kg \cdot m/rad^2$
$X_{q\|q\|}$	-0.02	$kg \cdot m/rad^2$

5.4.7 Added Mass

As explained in section 5.2.2.1, the added mass coefficients can be estimated using strip theory.

Approximating the vehicle by a ellipsoid, according to [53] the axial added mass can be given by:

$$Z_{\dot{w}} = -\frac{\alpha_0}{2-\alpha_0} m \tag{5.31}$$

where α_0 is given by:

$$\alpha_0 = \frac{2(1-e^2)}{e^3}\left(\frac{1}{2} ln\left(\frac{1+e}{1-e}\right) - e\right) \tag{5.32}$$

e is given by:

$$e = 1 - \left(\frac{b}{a}\right)^2 \tag{5.33}$$

The added mass of a single cylindrical slice is given by:

$$m_a(x) = \pi \rho R(x)^2 \tag{5.34}$$

The added mass of the vehicle is given by integrating equation 5.34 over the length of the vehicle. Therefore, its coefficients are defined as:

$$Y_{\dot{v}} = X_{\dot{u}} = \int_{xt}^{xc} m_a(x) dx \tag{5.35}$$

$$M_{\dot{u}} = -K_{\dot{v}} = \int_{xt}^{xc} x m_a(x) dx \tag{5.36}$$

$$M_{\dot{q}} = -K_{\dot{p}} = \int_{xt}^{xc} x^2 m_a(x) dx \tag{5.37}$$

$$N_{\dot{r}} = \int_{-r}^{r} x^2 m_a(x) dx \tag{5.38}$$

$$K_{\dot{p}} = Y_{\dot{p}} \tag{5.39}$$

$$X_{\dot{q}} = M_{\dot{u}} \tag{5.40}$$

Added mass coefficients coefficients are present in table 5.7.

Table 5.7: Added mass coefficients

Parameter	Value	Units
$X_{\dot{u}}$	-16.3487	kg
$Y_{\dot{v}}$	-16.3487	kg
$Z_{\dot{w}}$	0.3236	kg
$K_{\dot{p}}$	-2.3711	$kg \cdot m^2/rad$
$M_{\dot{q}}$	-2.3711	$kg \cdot m^2/rad$
$N_{\dot{r}}$	-0.1	$kg \cdot m^2/rad$
$M_{\dot{u}}$	-0.3052	$kg \cdot m$
$K_{\dot{v}}$	0.3052	$kg \cdot m$
$Y_{\dot{p}}$	0.3052	$kg \cdot m/rad$
$X_{\dot{q}}$	-0.3052	$kg \cdot m/rad$

5.4.8 Thrusters Parameters

The manufacturer of the thrusters provides its performance charts (figure 5.8) which are used to, using Matlab curve fitting tool, find a relation between PWM and the force produced by the thrusters.

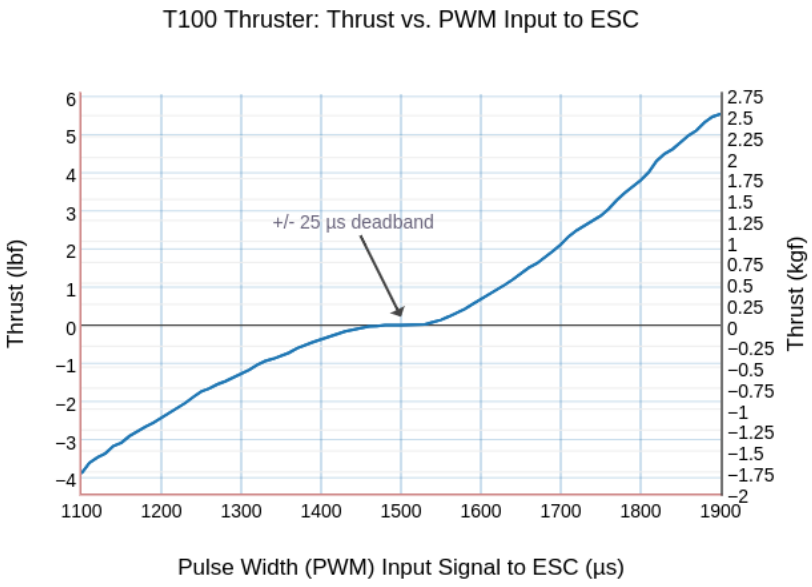

Figure 5.8: Performance chart of T100 thruster [27]

For the positive side of the curve, curve fitting (see figure 5.9) obtained the following function that relates PWM width with Force in Newtons:

$$F(x) = -0.2977x^2 + 21.96x + 1520 \tag{5.41}$$

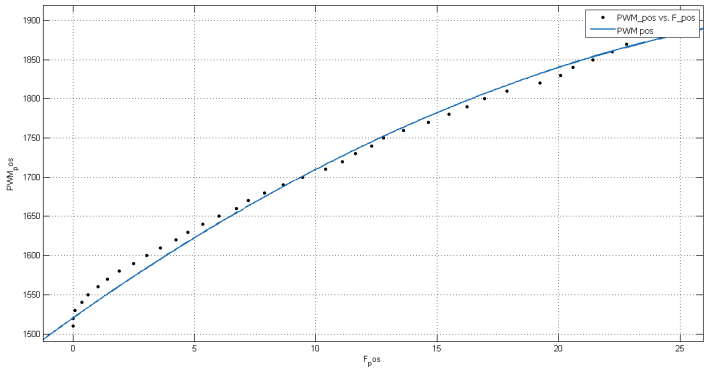

Figure 5.9: PWM positive side

For the negative side of the curve, curve fitting (see figure 5.10) yielded the following function:

$$F(x) = 0.9196x^2 + 37x + 1480 \tag{5.42}$$

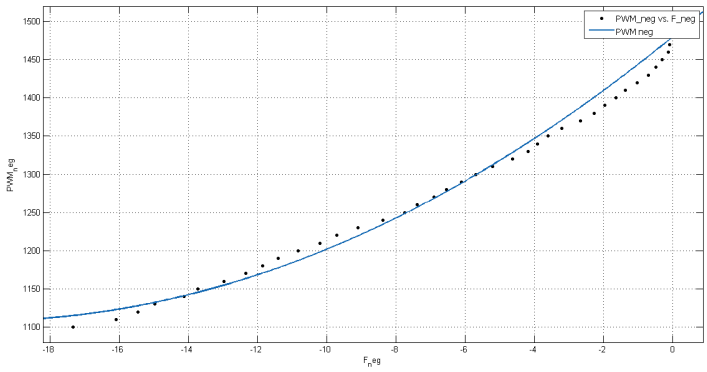

Figure 5.10: PWM negative side

5.5 Simulation

Using equations 5.4 and 5.5, it is possible to determine its acceleration in the body-fixed frame. To obtain its position and speeds it is necessary to resort to numerical integration.

For the simulation it was used Euler's method that applies the following iterative formula:

$$x_{n+1} = x_n + f(x_n, u_n) \cdot \Delta t \tag{5.43}$$

,where Δt s the time step. In this simulation, the time step considered was 0.1 s since this is the period of the control cycle used on the vehicle.

This method was implemented in Matlab using the MSS toolbox [65]. The simulation requires two inputs: initial conditions, that represent the vehicle's starting position, orientation and speeds and control inputs that are the external forces applied to the vehicle.

5.6 Model Validation

To verify the developed model, some simulations were executed. With this simulations, it is intended to ensure that the model behaves as expected and, therefore, simulates the vehicle's motion correctly.

The first test of the model is to observe the behavior of the vehicle when it begins at a nonzero depth. Since the vehicle has a positive buoyancy, it is expected that the vehicle rises to the surface as seen in figure 5.11.

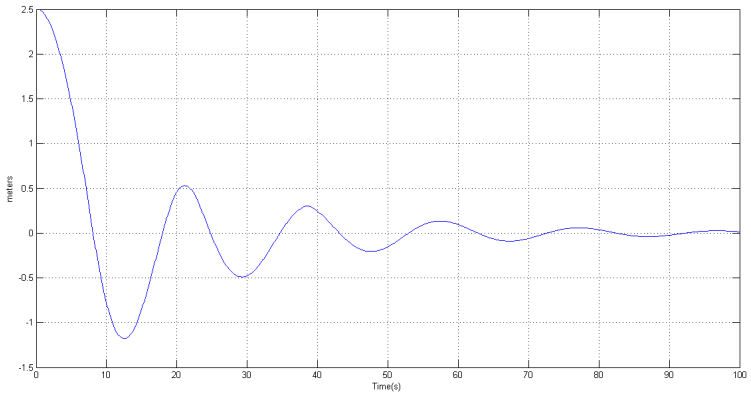

Figure 5.11: Simulation of the position in z starting from 2.5 meters

Another test to be considered is to observe the behavior of the profiler when acting on all 4 thrusters available with a force greater than the positive buoyancy. In this test, it is expected to see the profiler dive (figure 5.12).

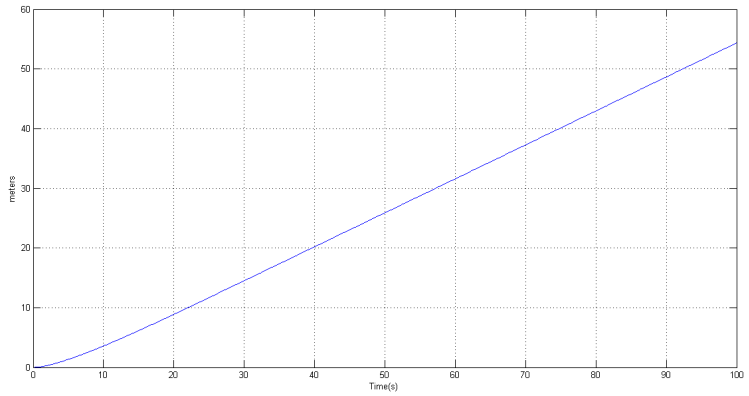

Figure 5.12: Simulation of the position in z when acting on the 4 thrusters with a force greater than the positive buoyancy

The remaining test consists in acting on only 2 thrusters from the 4 available. In this test, it is expected to see the profiler pitch/roll angles change. At the same time, the position in z should increase since the force provided by the two thrusters is larger than the buoyancy. The results are presented in figure 5.13

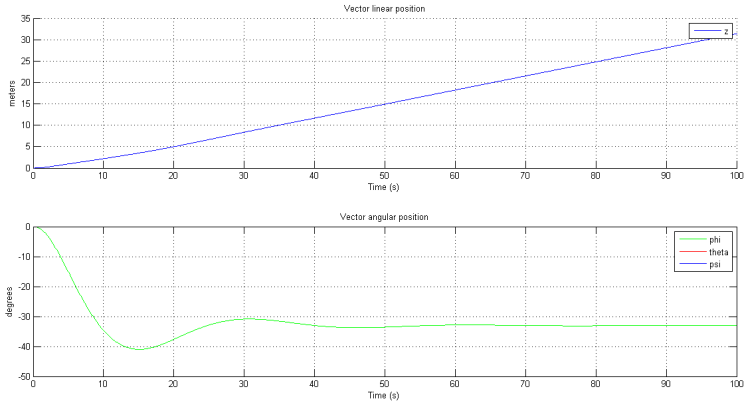

Figure 5.13: Simulation of the position in z, pitch and roll when acting on the 2 thrusters with a force greater than the positive buoyancy

Taking into account that all the results from the simulations were as expected, we conclude that the model for the vehicle is valid.

Chapter 6

Controllers

In this chapter it will be discussed and explained the speed and position controllers implemented in the vehicle.

It was implemented a speed controller that controls velocities in z, *pitch* and *roll* and, using the speed controller, a position controller that allows to control the profiler's position in z, *pitch* and *roll*.

For each maneuver described in section 4.3.2, the controllers were adjusted taking into account its purpose and the speeds that are typically involved in the described maneuvers.

6.1 Speed Controller

In order to perform the maneuvers defined in section 4.3.2, it is necessary to control the speed of the profiler as this is one of the parameters of the mission plan.

The model described in chapter 5 allows to obtain a model that describes the behavior of the system, despite being affected by modeling uncertainties and neglected terms. To ensure the correct behavior of the system despite these, the controllers that will be presented were also tested considering different parameters than those presented in chapter 5, to ensure its robustness.

The speed controller controls the velocities in z, pitch and roll. It was designed using Lyapunov theory, which allows to overcome the limitation of linear controllers that only are guaranteed to work around the selected operating points [66].

6.1.1 Lyapunov Theory

Lyapunov theory allows to conclude about the stability of a system and designing control laws as seen in [67, 66, 68].

It is based on the fact that if the total energy of a system is continuously dissipated, then the system will stabilize to an equilibrium point [69].

Lyapunov theory defines that if "*there exists a scalar function V of the state x with continuous first derivatives such that* $V(x)$ *is positive definite,* $\dot{V}(x)$ *is negative definite and* $V(x) \to \infty$ *as* $x \to \infty$*, then the equilibrium point at the origin is globally asymptotically stable*" [69].

A scalar function $f(x)$ is globally positive definite if $f(x) > 0, \forall x \neq 0$ and globally positive semi-definite if $f(x) \geq 0, \forall x \neq 0$.

Analogously, a scalar function $f(x)$ is globally positive negative if $f(x) < 0, \forall x \neq 0$ and globally negative semi-definite if $f(x) \leq 0, \forall x \neq 0$.

6.1.2 Nonlinear Control

The profiler is controlled using four thrusters. The control variable is the force to be provided by each thruster and is represented by f in expression 5.23. It is considered that the thrusters can instantly apply force to the vehicle, although this is not exact in reality. This assumption is made to simplify the model and is justified, since the time constants associated with the actuation are smaller than the ones associated with vehicle motion [68].

In order to simplify the determination of the speed controller, the order of the model was reduced eliminating lines and columns of the matrices that have low influence on the motion of the vehicle.

With this controller, it is aimed to control the velocities in z (w), roll(p) and pitch (q). Therefore, an error vector is defined as $e = v_r - v_{ref}$ as:

$$e = \begin{bmatrix} w - w_{ref} \\ p - p_{ref} \\ q - q_{ref} \end{bmatrix}$$

Then the Lyapunov candidate function V is defined as:

$$V = \frac{1}{2} e^T e \tag{6.1}$$

In which the time derivative:

$$\dot{V} = e^T \dot{e} = e^T (\dot{v}_r - \dot{v}_{ref}) \tag{6.2}$$

To guarantee the stability of the system, $\dot{V}$ must be negative definite which imposes:

$$e^T (\dot{v}_r - \dot{v}_{ref}) < 0 \Rightarrow (\dot{v}_r - \dot{v}_{ref}) < -k_e e \tag{6.3}$$

where $k_e \in \mathbb{R}, k_e > 0$.

To satisfy 6.3, we cannot vary the error variable instantaneously. Therefore, we define a new error variable α as:

$$\alpha = \dot{v}_r - \dot{v}_{ref} + k_e e \tag{6.4}$$

That imposes the time derivative Lyapunov function to be:

$$\dot{V} = e^T (\alpha - k_e e) \tag{6.5}$$

Substituting $\dot{v}_r$ in equation 6.4 for the expression for $\dot{v}$ given in 5.4 yields:

$$\alpha = M^{-1}[(C(v_r) + D(v_r)) \cdot v_r + g(\eta) + M(\dot{v}_{ref} - k_e e) + \tau] \tag{6.6}$$

Considering that τ can be actuated using f, to guarantee that $\dot{V}$ is negative definite τ is:

$$\tau = (C(v_r) + D(v_r)) \cdot v_r + g(\eta) + M(\dot{v}_{ref} - k_e e) \tag{6.7}$$

Finally, the forces f in each thruster are calculated as:

$$f = T^{+}\tau \tag{6.8}$$

It is concluded that the time derivative of the Lyapunov Function (6.5) results in:

$$\dot{V} = -k_e e^T e < 0, \forall e \neq 0 \tag{6.9}$$

The value k_e depends on actuators characteristics and this value is defined to prevent to reach saturation during long intervals. The controller gains are adjusted using practical techniques because, for nonlinear systems, it is hard to compute response characteristics.

6.2 Position Controller

The position controller controls the position on the 3 DOF: heave, pitch and roll. The position controller feeds the speed controller by passing the speed references to it (see figure 6.1).

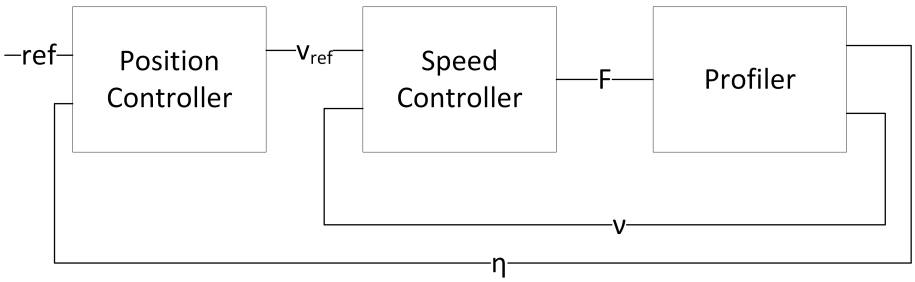

Figure 6.1: Block diagram of the controllers

Starting by defining the errors in z, pitch and roll as:

$$e_z = z_{ref} - z \tag{6.10}$$

$$e_\phi = \phi_{ref} - \phi \tag{6.11}$$

$$e_\theta = \theta_{ref} - \theta \tag{6.12}$$

The references for the speed controller are then defined as:

$$w = K_{pz} e_z + K_{iz} \int e_z dt + K_{dz} \frac{de_z}{dt} \tag{6.13}$$

$$p = K_{p\phi} e_\phi + K_{i\phi} \int e_\phi dt \tag{6.14}$$

$$q = K_{p\theta} e_\theta + K_{i\theta} \int e_\theta dt \tag{6.15}$$

where $K_{pz}, K_{iz}, K_{dz}, K_{p\phi}, K_{i\phi}, K_{p\theta}, K_{i\theta} \in \mathbb{R}^+$ are proportional, integral and derivative gains, respectively.

The integral part of the controller is used to eliminate the steady state error to the reference. For the controller in z, it was also considered a derivative term to prevent overshoot, since during the descent, the integral part of the controller accumulates error that makes the profiler overshoot.

For each maneuver, the controller gains were determined considering velocities saturation. Since the dynamics of each maneuver are different, the gains considered were determined for each maneuver.

6.2.1 Position Reference

Changing abruptly the position reference of the controllers makes the vehicle accelerate abruptly, causing significant consumption peaks since the vehicle needs to accelerate quickly to meet the speed reference.

To ensure that the position reference for the controllers is not made abruptly, the position reference is generated using a ramp with a slope equal to the desired velocity for the profiler (figure 6.2). This makes the profiler go to the desired speed and position more smoothly.

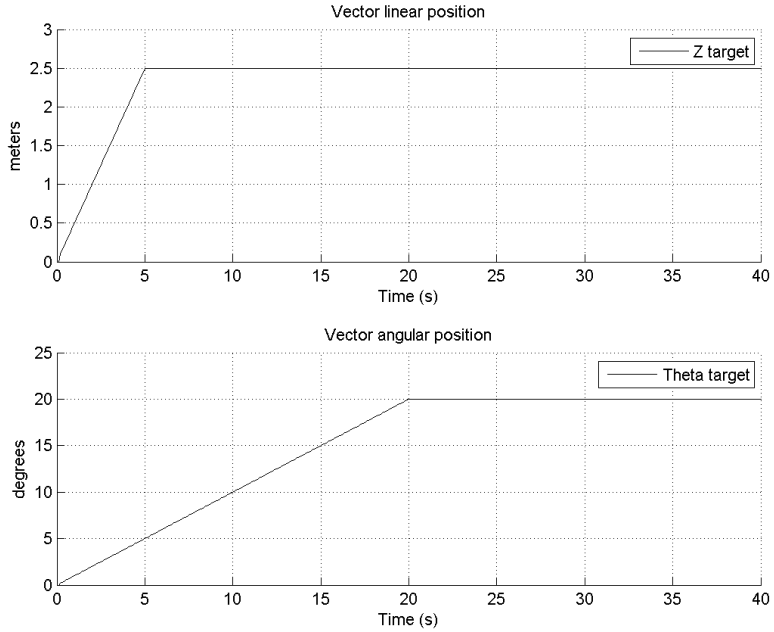

Figure 6.2: Position reference with velocity in z of 0.5 m/s and in pitch of 0.05 rad/s

6.3 Controller Gains

As stated above, the presented model has uncertainties since it is hard to precisely determine the hydrodynamic coefficients.

Having this in mind, the controllers were tuned based on experimental tests to achieve the desired performance for the system since the obtained model is not exact.

To determine the controller gains, it was followed a empirical approach starting with only proportional gains. The initial values considered for the proportional gains were small to achieve a smooth response. After the proportional gains were determined, it was added an integral gain to grant no steady state error and a derivative gain to decrease overshoot and oscillations. The gains considered are presented below in section 6.4.

6.4 Controllers Simulation

In this section it will be presented the simulation results for each controller designed. The controllers presented bellow were simulated resorting to the model described in chapter 5.

6.4.1 Goto Z

Figures 6.3 and 6.4 present the simulations for the gotoz controller. It is presented two simulations for two different cases: a descent with 0 degrees in pitch and roll angles and a simulation with 20 degrees in pitch and roll.

For the first case, the profiler was instructed to descent to 3.5 meters with a speed of 0.3 m/s.

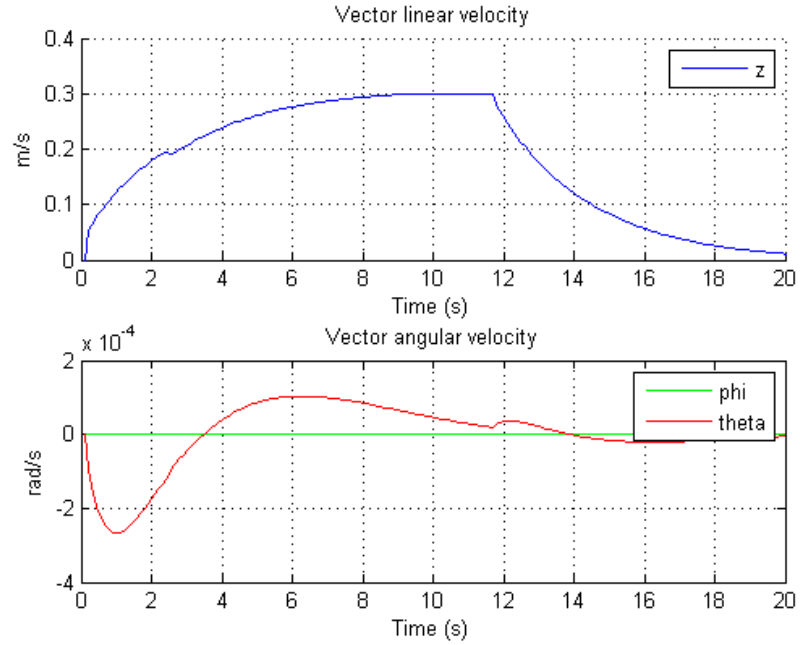

Figure 6.3: Speed during gotoz maneuver with 0 in pitch and roll

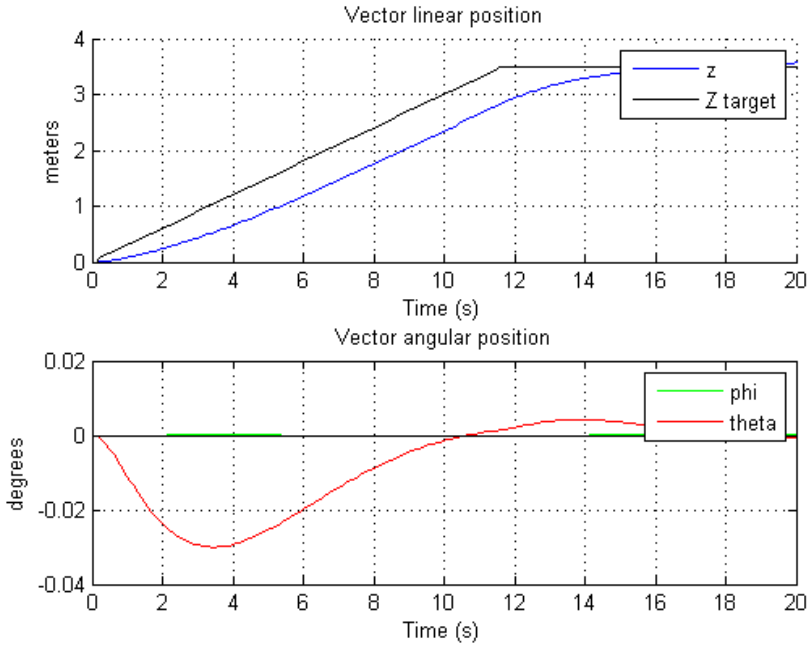

Figure 6.4: Position during gotoz maneuver with 0 in pitch and roll

For the second case, the profiler was instructed to descent with the same criteria of the first

case, but this time with 20 degrees in both pitch and roll. The results are presented in figures 6.5 and 6.6.

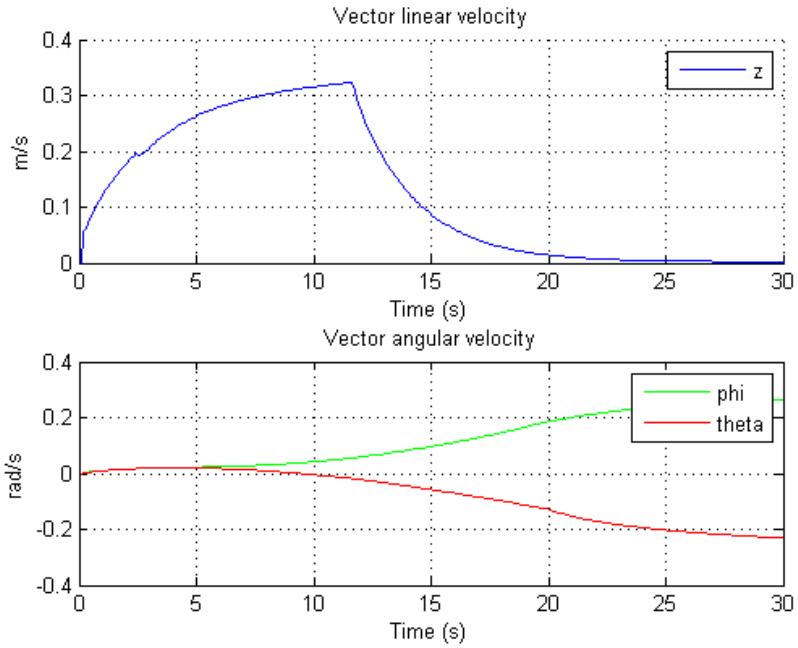

Figure 6.5: Speed during gotoz maneuver with 0 in pitch and roll

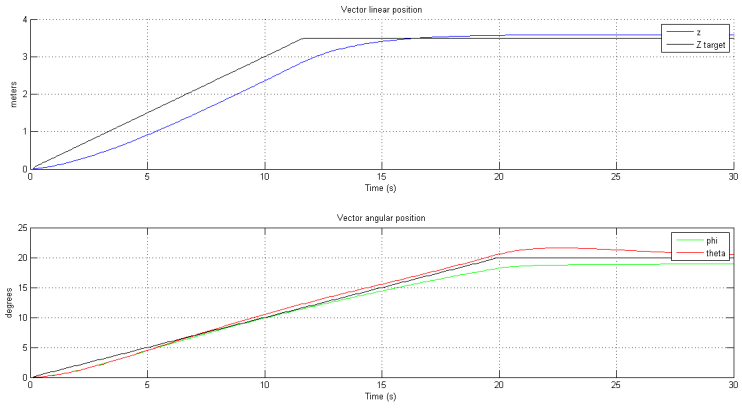

Figure 6.6: Position during gotoz maneuver with 0 in pitch and roll

The gotoz controller gains are presented in table 6.1.

Table 6.1: Gotoz controller gains

Parameter	Value
K_{pz}	0.4
K_{iz}	0.0005
K_{dz}	0.01
$K_{p\phi}$	0.6
$K_{i\phi}$	0.02
$K_{p\theta}$	0.6
$K_{i\theta}$	0.02

6.4.2 Surface

For simulating the hover maneuver, it was considered that the profiler should surface from a depth of 3.5 meters with an ascent speed of 0.2 m/s. The results are presented in figures 6.7 and 6.8:

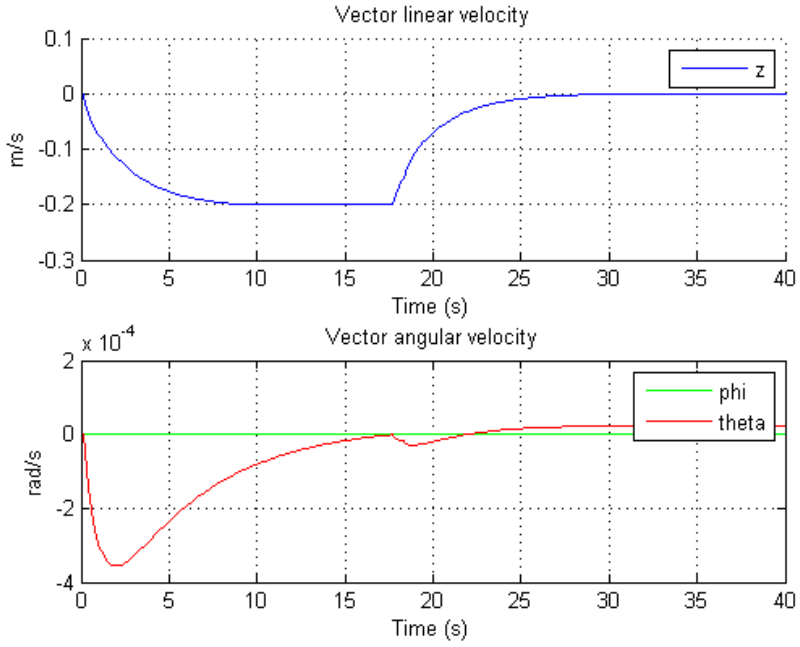

Figure 6.7: Speed during surface maneuver

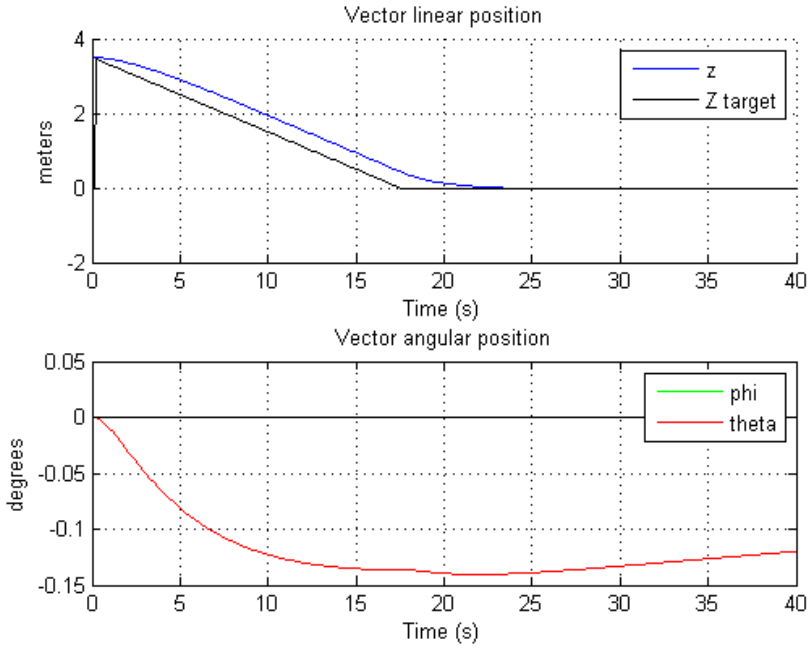

Figure 6.8: Depth during surface maneuver

The gains considered for the surface controller are (table 6.2):

Table 6.2: Surface controller gains

Parameter	Value
K_{pz}	0.4
K_{iz}	0.0005
K_{dz}	0.015
$K_{p\phi}$	0.2
$K_{i\phi}$	0.0002
$K_{p\theta}$	0.2
$K_{i\theta}$	0.0002

6.4.3 Hover

For simulating the hover maneuver, in this example, it was considered that the profiler should hover at 2.5 meters with a descent speed of 0.2 m/s. The results are presented bellow (figures 6.9 and 6.10):

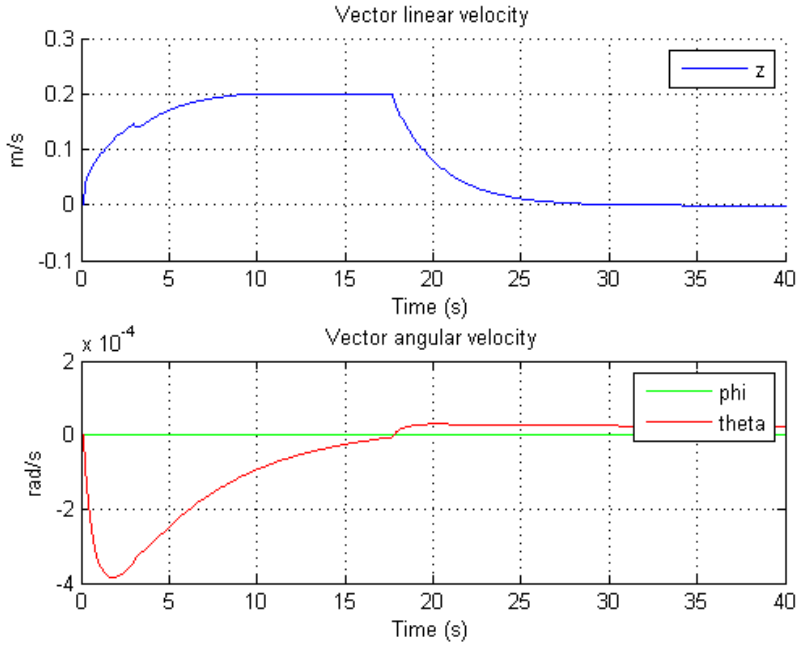

Figure 6.9: Speed during hover maneuver

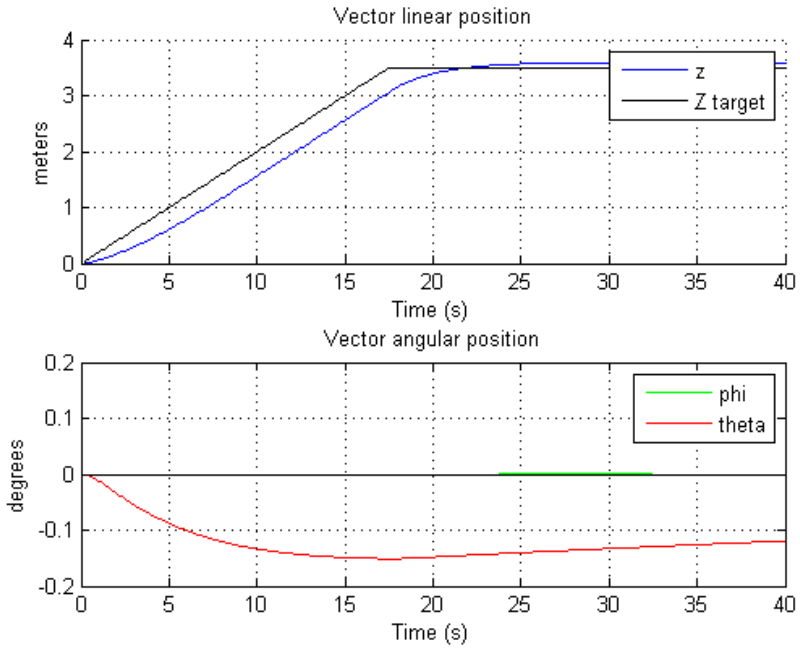

Figure 6.10: Depth during hover maneuver

The gains considered for the controllers in this maneuver are (table 6.3):

Table 6.3: Hover controller gains

Parameter	Value
K_{pz}	0.4
K_{iz}	0.0005
K_{dz}	0.015
$K_{p\phi}$	0.2
$K_{i\phi}$	0.0002
$K_{p\theta}$	0.2
$K_{i\theta}$	0.0002

Chapter 7

Experimental Results

In this chapter it will be presented and discussed the results obtained from experimental tests done to validate the developed work.

As stated in chapter 4, the data is logged to the log file with a period of 250 ms and it is available in a file named DataLog<time>.txt. This file was retrieved from the profiler using the Wi-fi connection and was then processed to obtain the figures shown in this chapter.

The tests presented in this chapter were performed in two places: the water tank at FEUP and the water tank at ISEP. The first has a depth of 1.7 meters making the depth of the movement made by the profiler very limited. The second has a depth of 5 meters.

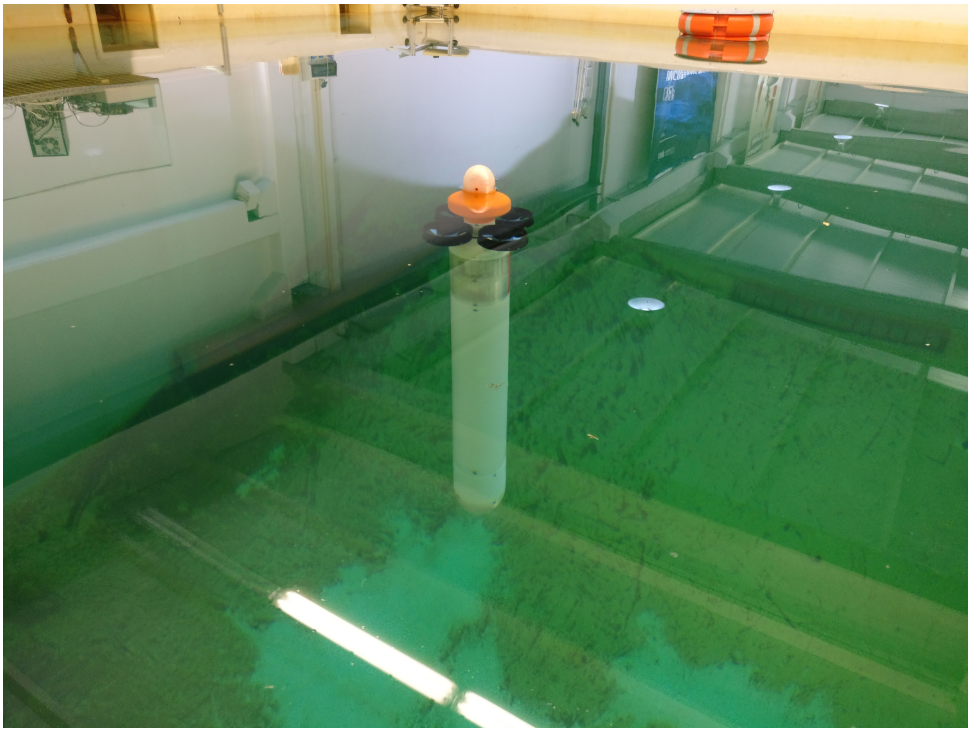

Figure 7.1: Profiler at surface in the test tank at ISEP

The tests made in the tank were divided in two subcategories: validation tests, to ensure the normal working underwater of the vehicle and guarantee that the data acquired by the profiler was valid and controller tests which consist in testing and experimental validation of the controllers already described in chapter 6.

For the tests presented in this chapter, the data from the external temperature sensor will only be presented once, since the water temperature variation is negligible due to the depth of the tank. Internal temperature variation is also negligible and will also be presented once. The data from GPS is not available since the tests were performed inside a building.

7.1 Validation Tests

The Profiler was tested to ensure that it would work underwater without water entrance inside the pressure housing. In these set of tests, the data acquired by the profiler was also tested to guarantee that the measurements taken by the vehicle were correct.

7.1.1 Sealing

Before launching the profiler in the tank it was necessary to ensure that the pressure housing is sealed from water entrance. Even though the endcap has two o-rings as a safety measure, this test was performed with one o-ring to confirm the correct sealing of the pressure housing. To guarantee that the pressure housing was sealed, it was used a vacuum pump to remove the air and it was measured the pressure inside the housing. The pressure housing was then left sealed for 30 minutes without an increase in the pressure inside, meaning that the pressure housing was correctly sealed.

7.1.2 Stationary

The vehicle was the dropped in the tank where it was left stationary for 90 seconds recording data. This test produced the following results:

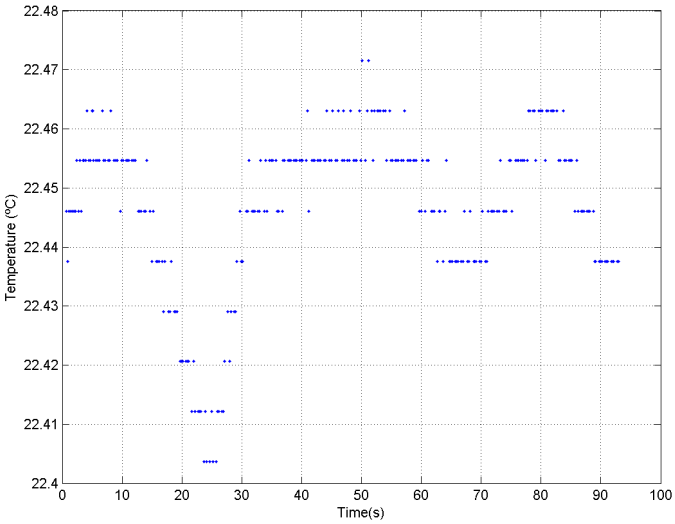

Figure 7.2: External temperature with the profiler stopped

In figure 7.2, it is possible to observe the external temperature measured by the profiler. As expected, the measured temperature does not vary significantly. In all the performed tests, the external temperature does not vary greatly, since the depth of the dives is insufficient to observe a significant variation. The mean value measured by the sensor during this period is 22.45 ºC and the standard deviation is 0.014ºC.

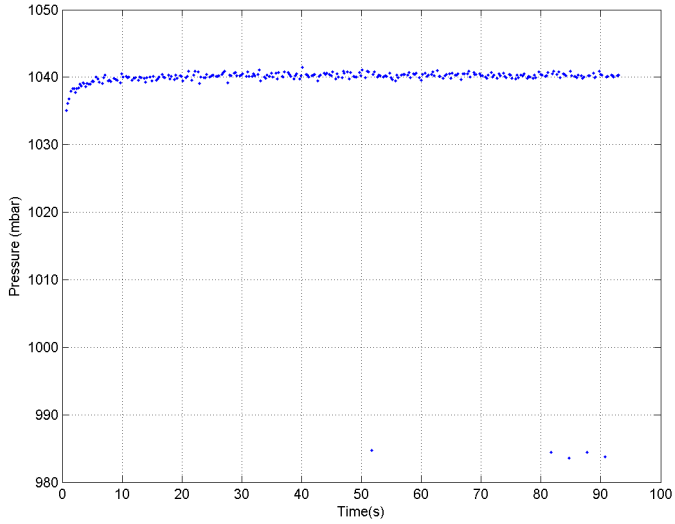

Figure 7.3: Pressure with the profiler stopped at surface

Figure 7.3 shows the pressure variation over time. It is possible to observe that the measured pressure is close to the standard atmosphere (1013.25 mbar) as expected. The mean value of the

pressure during the measured period is 1039.1 mbar and the standard deviation is 7.49 mbar.

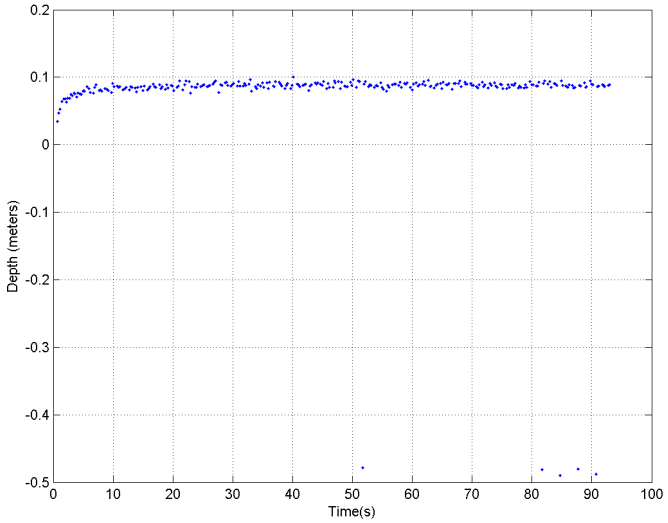

Figure 7.4: Depth with the profiler stopped at surface

In figure 7.4 it is possible to observe the profiler depth. As the profiler is stopped at surface, the depth does not vary. It is possible to observe that the depth is not zero because the pressure sensor takes around 2 seconds until the measurements stabilize.

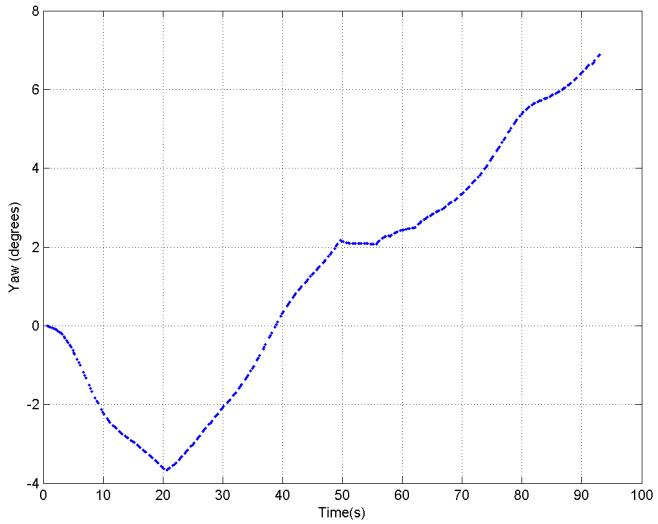

Figure 7.5: Yaw with the profiler stopped

As seen in figure 7.5, the yaw measured by the IMU drifts over the time due to the drift of the gyroscope and to the fact that the accelerometer cannot measure this type of motion. Therefore, the yaw measurements are not used for heading.

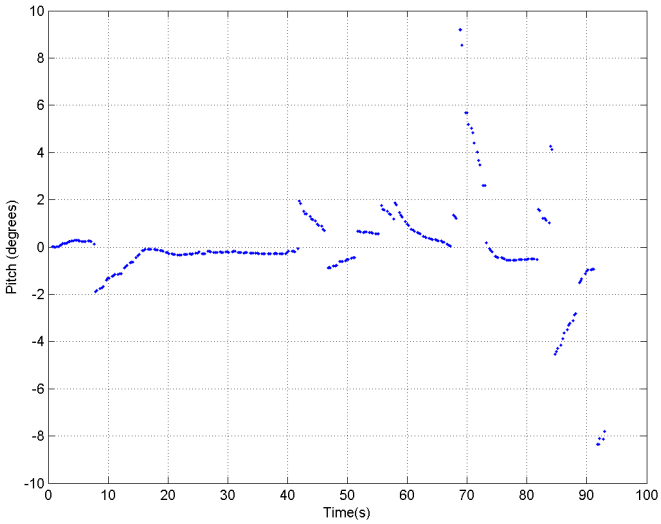

Figure 7.6: Pitch with the profiler stopped

In figure 7.6, the pitch angle of the profiler when stopped is represented. It is possible to observe that the pitch angle is close to zero.

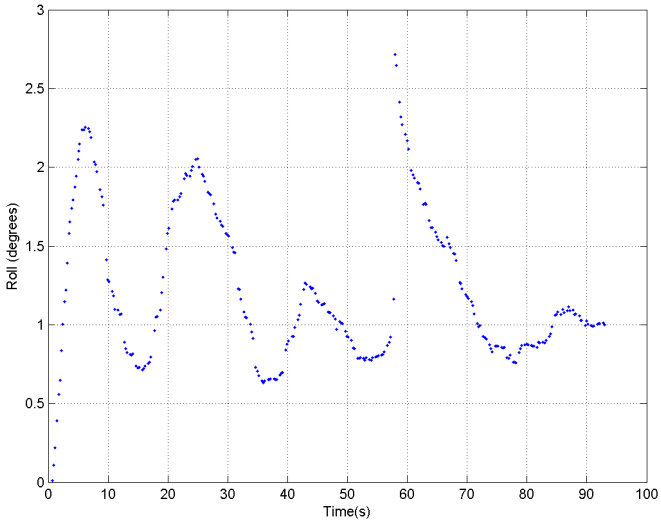

Figure 7.7: Roll with the profiler stopped

Figure 7.7, shows the roll angle of the profiler when stopped. Due to the misalignment of the CB and CG, the angle is not zero. It is possible to observe that it stabilizes around 1 degree.

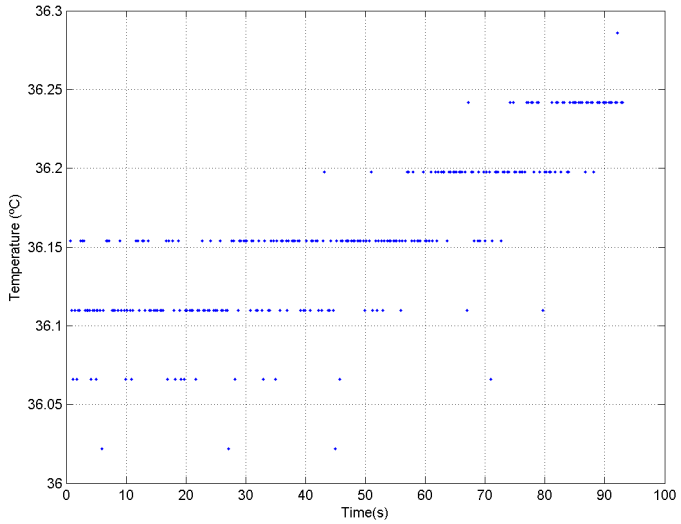

Figure 7.8: Internal temperature of the profiler

Since these tests were performed after the profiler internal temperature have stabilized, it is possible to see in figure 7.8 that this is approximately 36°C. To observe the variation of the internal temperature of the profiler after turning it on, it was recorded its temperature during five minutes. The results are presented in figure 7.9 where it is possible to observe that it increases after starting the system.

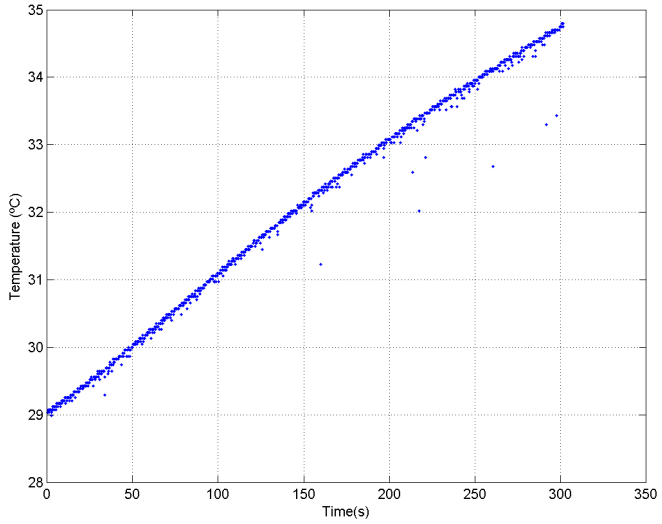

Figure 7.9: Internal temperature of the profiler

Figures 7.10, show data related to the batteries status, more precisely, the remaining percentage of the batteries of the profiler and the remaining time to empty the batteries. As the profiler is

stopped in this test and power consumption is low, the variation between consecutive measures observed on both figures is small.

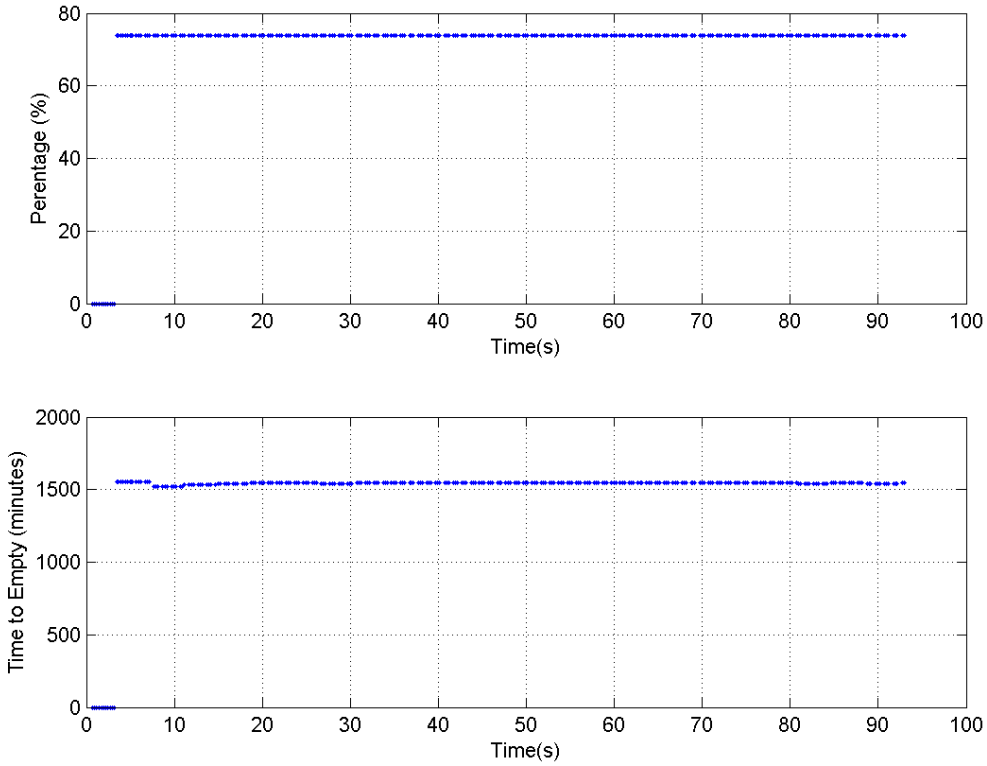

Figure 7.10: Remaining percentage and time to empty of the batteries of the profiler

7.1.3 Dive

In this test, the vehicle was pushed forcing the profiler to dive while registering the data. It is expected to see the depth increase until 0.4 meters (maximum depth that the profiler reaches in the pool) and then, due to the positive buoyancy, surface and oscillate on the surface (7.11). The objective of this test is to guarantee that the pressure sensor is functioning correctly and is able to detect depth variations.

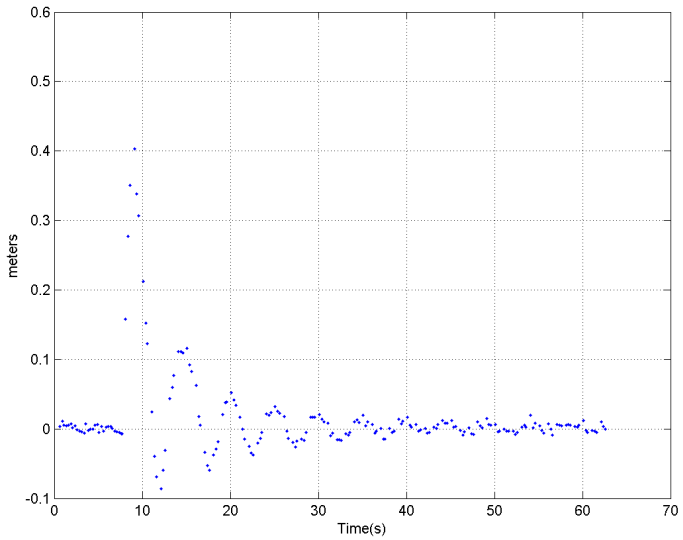

Figure 7.11: Depth variation when pushing the profiler

7.1.4 Angular Test

In this test the vehicle pitch angle was varied to approximately to 50 degrees. These results were used for determining the center of buoyancy, as described in 5. In this test (figures 7.12 and 7.13, it is expected to see the angles in pitch and roll vary.

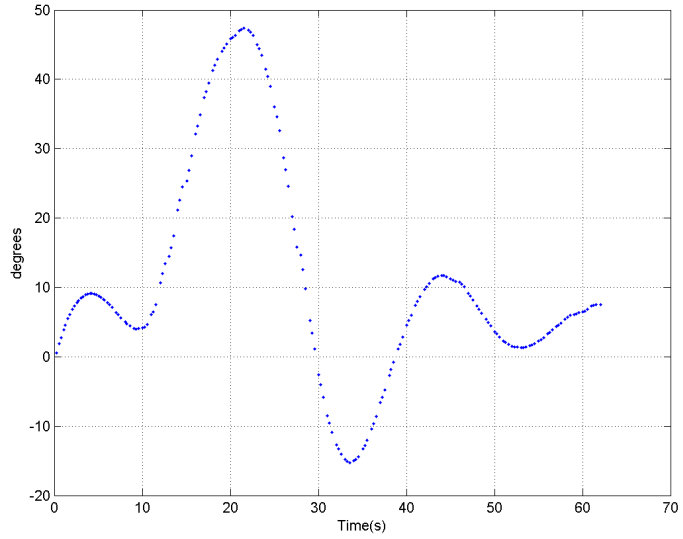

Figure 7.12: Pitch variation when inclining the vehicle in Pitch

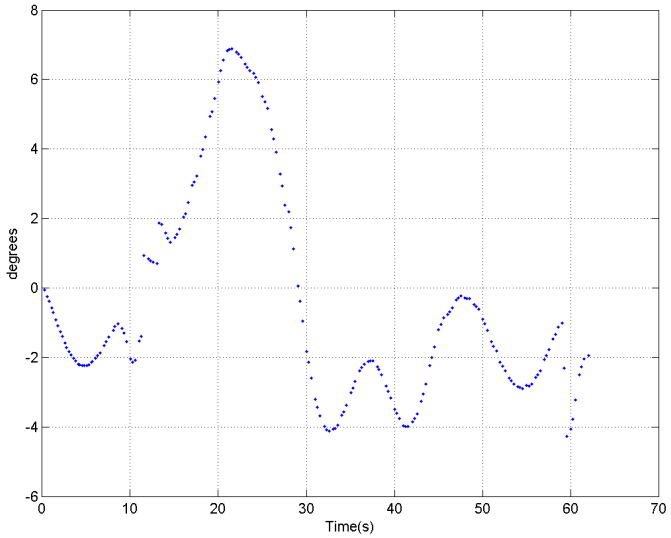

Figure 7.13: Roll variation when inclining the vehicle in Pitch

7.2 Controller Tests

In this section it will be presented the tests performed to the controllers presented in 6. In all following figures, it is also represented the target depth of the profiler at any moment to verify that the profiler controls its speed, following the reference during the maneuver.

All the tested maneuvers are preceded by a wait command of 5 seconds to allow angle measurements to stabilize. Moreover, the profiler stops 2 seconds to initialize the ESC's resulting in 7 seconds before the profiler starts the maneuver.

7.2.1 Go To z Test

To test the go to z maneuver, two different tests were performed: one where the profiler dives to a defined depth completely vertical (ie, pitch and roll are both 0) and another where the dive is made with an angle in pitch and roll different than zero.

The maneuver is considered completed when the profiler is within 10 cm of the target and the error of the angles is lower than 2 degrees.

For the first test, the profiler was instructed to dive to 0.4 meters with a speed of 0.1 m/s. The results are presented in figure 7.14:

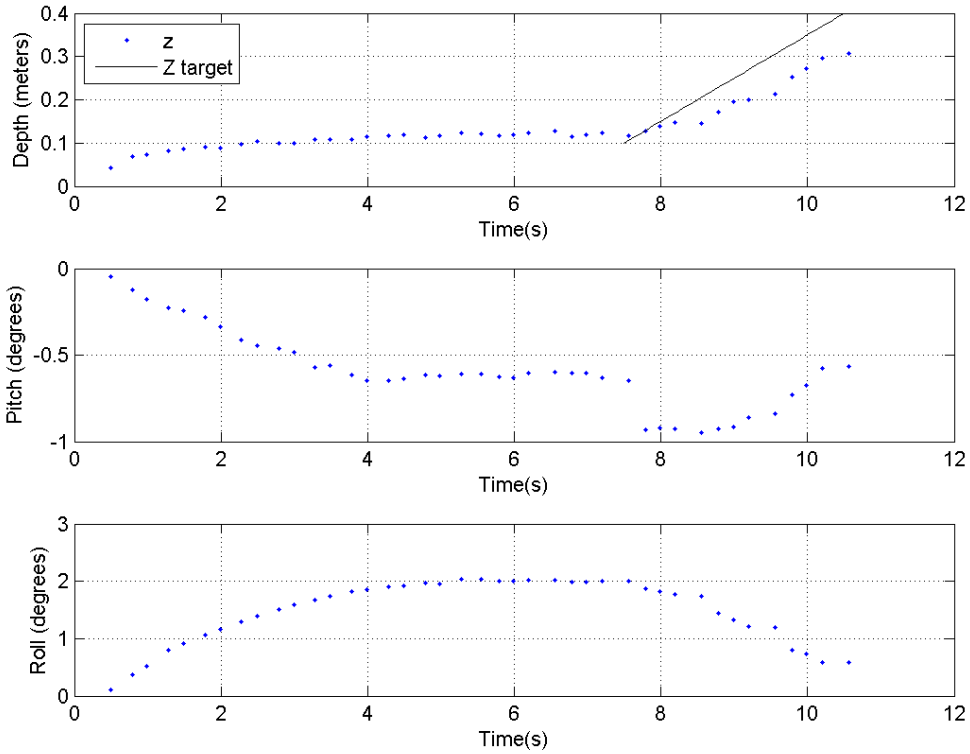

Figure 7.14: Depth, pitch and roll of the profiler when diving up to 0.4 m

In figure 7.14, it is possible to observe that the profiler achieves the desired depth with a controlled speed.

As expected, the angles in pitch and roll are small.

For the second test, the profiler was commanded to dive up to 0.4 m depth with an angle in pitch of -30 degrees and an angle in roll of 30 degrees. This test was performed in the smaller tank. The results are presented in figure 7.15:

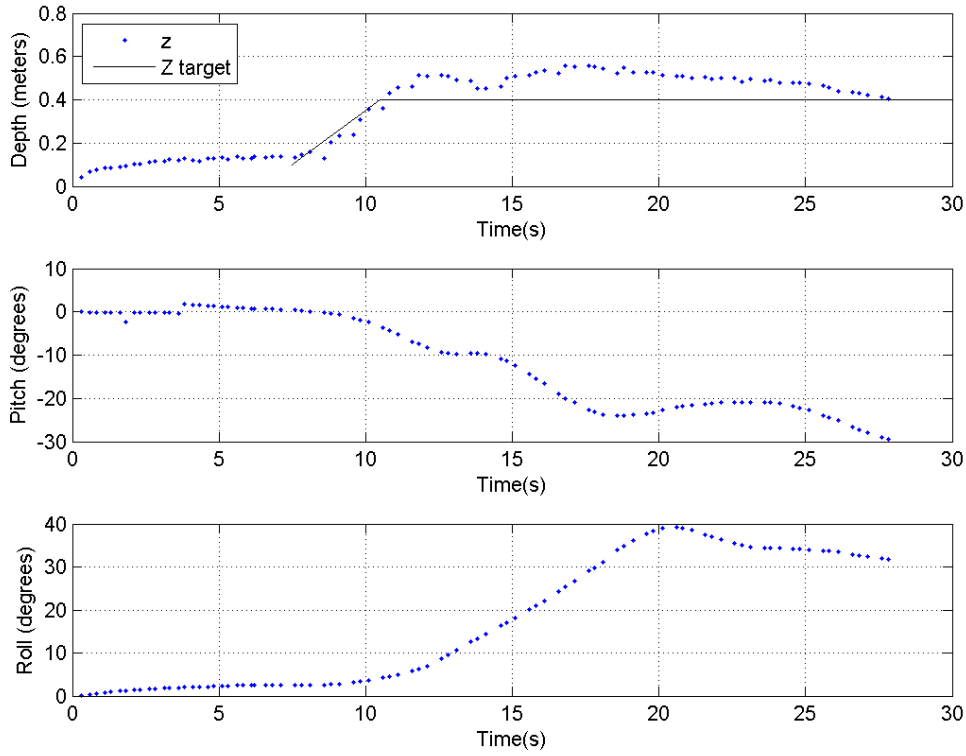

Figure 7.15: Depth, pitch and roll of the profiler when diving up to 0.4 m

In figure 7.15, it is possible to observe that the profiler exceeds its depth target. This happens because the forces needed to provide a rotation of the vehicle, also make the profiler go down. It is possible to notice that once the angles in pitch and roll are reached, the profiler goes to the desired position as needed.

After the tests performed in the smaller tank, tests were performed in the bigger tank to observe the profiler behavior to greater depths. The profiler was instructed to dive up to 3.5 meters. The results are presented bellow:

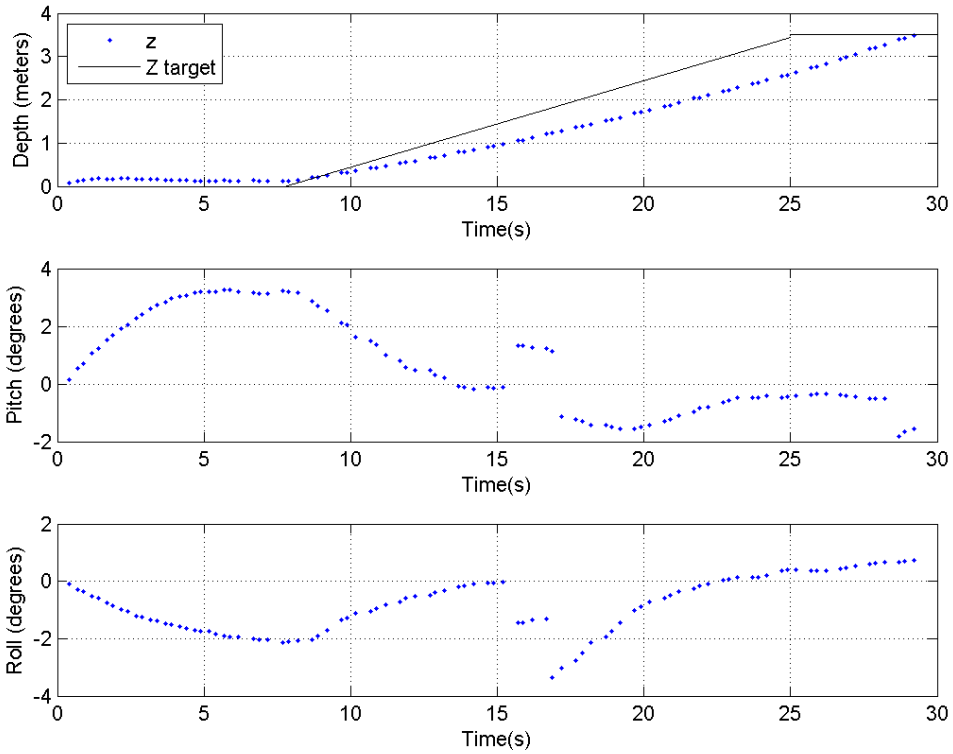

Figure 7.16: Depth, pitch and roll during gotoz to 3.5 meters

Similiarly to the results obtained in the smaller tank, the profiler achieves its target depth (figure 7.16) with a controlled speed while, at the same time, controlling its pitch and roll angles maintaining these close to zero.

7.2.2 Surface

Since the surface maneuver is to perform a controlled rise of the profiler to the surface, for this test, it was necessary to perform a gotoz before the surface. Therefore, in this test the profiler was instructed to reach the depth of 1.5 meters and then control its ascent with a speed of 0.2 m/s to 0.1 meters. The results are presented in figure 7.17 and show that the profiler rises with a controlled speed.

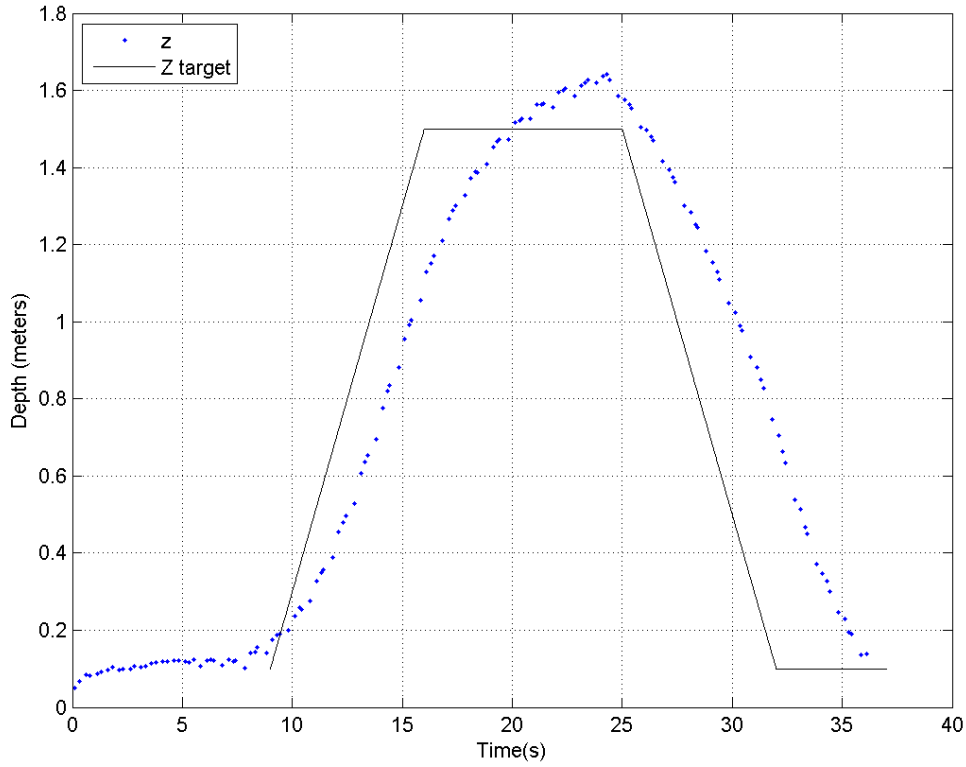

Figure 7.17: Depth during gotoz and surface

7.2.3 Hover

For the hover test, the profiler was instructed to hover at 0.3 meters of depth with a speed of 0.1 m/s during 60 seconds. The results are presented in figure 7.18.

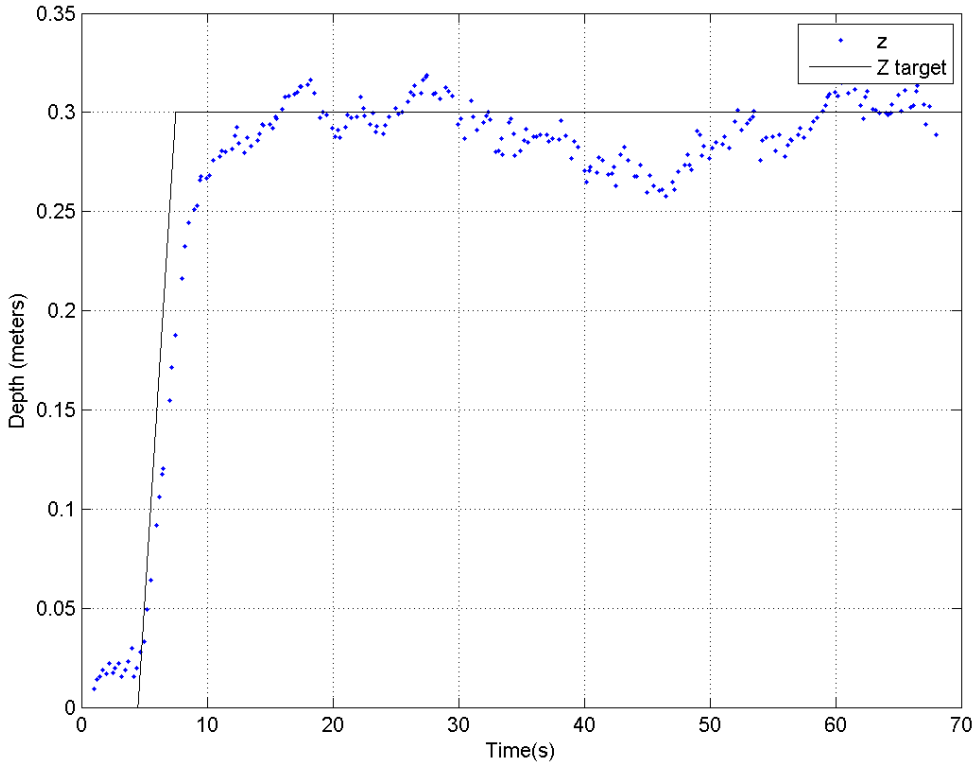

Figure 7.18: Results of the experimental tests for hover control

In figure 7.18, it is possible to observe that the profiler dives up to approximately 0.3 and hovers in that zone for 60 seconds, as defined.

To ensure the robustness of the controller, the profiler was subjected to external disturbances in figure 7.19. In this test, after achieving the hovering position, the profiler was pushed; after recovering, it was pulled to the surface and then pushed again.

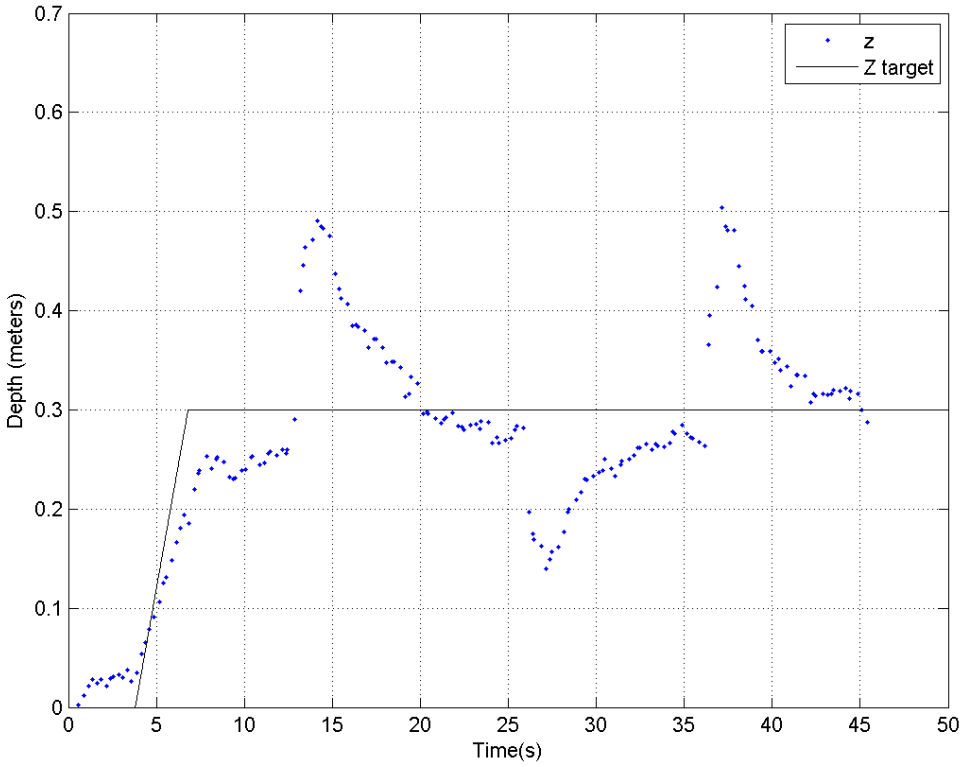

Figure 7.19: Results of the experimental tests for hover control

The vehicle goes to the target depth with the velocity controlled by the position and velocity controller. Due to the ramp reference explained in chapter 6, the velocity is controlled with the defined value. When the target depth is achieved, the controller stays close to the position. Once a perturbation occurs, since the reference of position is no longer a ramp, it varies abruptly making the controller react with a bigger velocity that decreases as the error to the target decreases explaining the different behavior during the diving and the disturbance reaction.

After these tests at the smaller tank, a test was performed in the bigger tank considering a greater descent. In this test, the profiler was instructed, after waiting 5 seconds, to dive to 2.5 meters and hover at this depth for 60 seconds. The results are presented bellow:

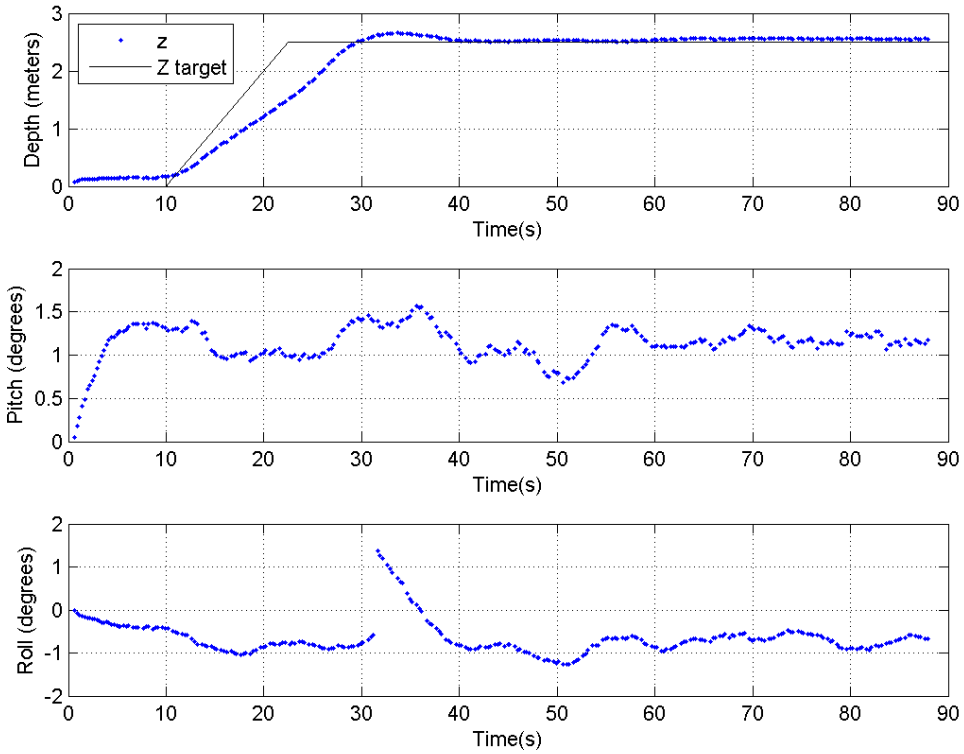

Figure 7.20: Depth during hover

In figure 7.20, it is possible to observe that the profiler hovers at the desired depth for 60 seconds. In figure 7.20, it is possible to observe that the profiler reaches 2.5 meters with a small overshoot that is compensated. In pitch and roll, it is possible to observe that the profiler angles are small and close to zero.

7.3 Multiple Maneuvers

Proven the correct operation of the controllers in section 7.2, in this section it will be presented an example of mission plans tested in the tank at ISEP as well as its results.

7.3.1 Hover at Different Depths and Surface

In this test, the vehicle was instructed to hover at three different depths: 1.5, 2.5 and 3.5 meters followed by a surface with an ascent rate of 0.5 m/s. The results are presented in figure 7.21.

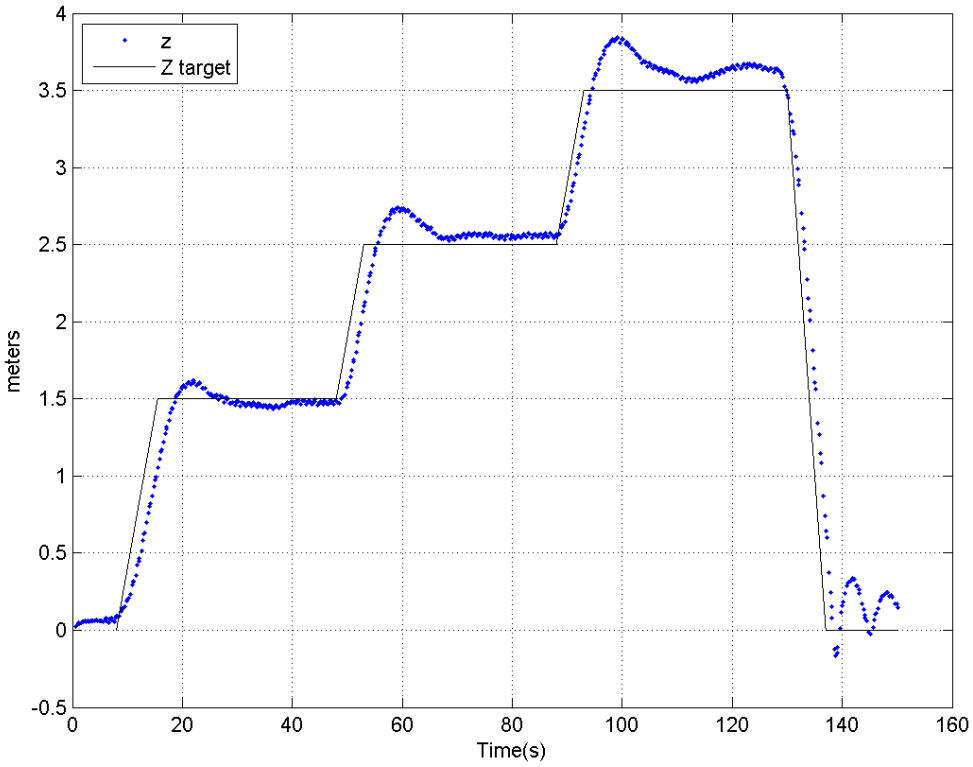

Figure 7.21: Hover at three different depths in one mission

It is possible to see that the vehicle hovers at the desired depths and then, after completing the maneuvers, rises with a controlled speed to the surface.

Chapter 8

Conclusions and Future Work

This chapter will present the main contributions of this work as well as the conclusions drawn from it. Additionally, it will present some suggestions on future work that would improve the prototype operation, allow to achieve a better performance and add new features.

8.1 Main Contributions

This dissertation focused on the development of a new autonomous underwater vehicle to move mainly in the vertical column of water. At the same time, the vehicle should allow to control its heading to add extra functionality to the operation of a typical profiler, that usually only moves in the vertical plan and drifts with the currents.

To develop the profiler it was used a systems engineering approach following three different phases in an iterative process: design, implementation and validation that allow to achieve a functional prototype in the end if this work.

The main objective of this work was to develop a prototype that satisfy the requirements defined in chapter 3 and to implement controllers that allow this system to move vertically and provide limited horizontal movement.

The development of the prototype focused on the physical implementation of the vehicle, that allows to fulfill the prototype requirements defined to allow to perform profiles up to 200 m with a portable and modular vehicle, and on the software implementation, that allows the vehicle to autonomously operate, executing a mission plan and gathering data from its set of sensors. The modularity of the vehicle allows to easily add new payloads as the vehicle is prepared to receive new sensors using the places available in the endcaps. This can be integrated in the vehicle's software by implementing a device driver and incorporating the measurements in the shared memory and the log.

To design the controllers for vehicle movement, it was implemented a model that describes the profiler dynamics and kinematics in 6 DOF. This model was used to test the performance of the controllers before deployment in the tests tank.

Taking into account the results presented in chapter 7, it is possible to conclude that the main objective of this dissertation was achieved since, in the end of this work, the developed prototype was capable of accomplishing most of the defined requirements.

8.2 Future Work

As seen above, the developed system, despite completing most of the requirements defined, still needs further testing since it was only tested in a controlled environment.

The first suggestion to improve the work done is to add more sensors, like a camera in the bottom of the profiler to allow image acquirement from the sea bottom, an altimeter to measure the altitude in relation to the sea bottom and an Iridium module to communicate the profiler position in the end of the mission. These features were initially thought but were not implemented since they were not considered a priority due to the time limitation of the dissertation. However, considering the modularity of the system developed, all these subsystems are ready for a swift integration.

As referred in chapter 5, the parameters used in the simulation should be determined based on experimental tests. One of the improvements that could be done is to determine these based on tests and identification techniques to get a more accurate model of the vehicle. Using a more accurate model it allows to have a simulation closer to the real world that would be used to improve the performance of the controllers developed.

Another limitation of the profiler is the rotation in yaw caused by the misalignment between the CB and CG and the rotation of the thrusters blades. To overcome this limitation it is possible to add another DOF in yaw. This could be implemented by replacing the used IMU for a 9 DOF IMU. A 9 DOF IMU would allow to measure angles in yaw without the drift seen in the used 6 DOF IMU because this type, besides having a gyroscope and an accelerometer, also has a magnetometer that allows to have drift free measures in yaw. Having yaw angle, its rotation could be controlled adding a subsystem to control it.

Overcoming this limitation would grant full control of the heading of the profiler and therefore would allow to implement new maneuvers. Some of these were already considered, like perform profiles in a defined direction or correct the displacement introduced by currents.

Moreover, the configuration file that is used to plan a mission could be also used to configure the profiler operation by defining the payloads used in the operation as well as other configurable parameters like the logging period.

Another improvement that could be done is to design an user interface that would be used for mission planing, starting a mission and analyze the data retrieved from the profiler.

References

[1] Matthew Dunbabin and Lino Marques. Robotics for environmental monitoring. *IEEE Robotics & Automation*, page 16, 2012.

[2] Russell B. Wynn, Veerle A.I. Huvenne, Timothy P. Le Bas, Bramley J. Murton, Douglas P. Connelly, Brian J. Bett, Henry A. Ruhl, Kirsty J. Morris, Jeffrey Peakall, Daniel R. Parsons, Esther J. Sumner, Stephen E. Darby, Robert M. Dorrell, and James E. Hunt. Autonomous underwater vehicles (AUVs): Their past, present and future contributions to the advancement of marine geoscience. *Marine Geology*, 352:451 – 468, 2014. 50th Anniversary Special Issue.

[3] T. B. Sanford, J. H. Dunlap, J. A. Carlson, D. C. Webb, and J. B. Girton. Autonomous velocity and density profiler: Em-apex. In *Proceedings of the IEEE/OES Eighth Working Conference on Current Measurement Technology, 2005.*, pages 152–156, June 2005.

[4] B. Ward and T. Fristedt. Air-sea interaction profiler: Autonomous upper ocean measurements. In *2008 IEEE/OES US/EU-Baltic International Symposium*, pages 1–8, May 2008.

[5] Dean Roemmich, Gregory C. Johnson, Stephen Riser, Russ Davis, John Gilson, W. Brechner Owens, Silvia L. Garzoli, Claudia Schmid, and Mark Ignaszewski. The argo program: Observing the global ocean with profiling floats. *Oceanography*, 22, June 2009.

[6] V. H. Fernandes, A. A. Neto, and D. D. Rodrigues. Pipeline inspection with AUV. In *2015 IEEE/OES Acoustics in Underwater Geosciences Symposium (RIO Acoustics)*, pages 1–5, July 2015.

[7] E. Desa, R. Madhan, N. Dabholkar, S. Prabhudesai, G. Navelkar, A. Mascarenhas, S. Afzulpurkar, M. Phaldesai, and P. K. Maurya. In situ profiling of eastern arabian sea coastal waters using a new autonomous vertical profiler. *IEEE Journal of Oceanic Engineering*, 38(1):43–54, Jan 2013.

[8] A. Schwithal and C. Roman. Development of a new lagrangian float for studying coastal marine ecosystems. In *OCEANS 2009 - EUROPE*, pages 1–6, May 2009.

[9] E. Petzrick, J. Truman, and H. Fargher. Profiling from 6,000 meter with the apex-deep float. In *2013 OCEANS - San Diego*, pages 1–3, Sept 2013.

[10] P. K. Maurya, E. De Sa, A. C. Dubey, N. Dabholkar, and A. Pascoal. Autonomous hovering profiler. In *2016 IEEE/OES Autonomous Underwater Vehicles (AUV)*, pages 268–272, Nov 2016.

[11] M. Antonio, S. Afzulpurkar, G. Navelkar, R. Madhan, P. Maurya, E. Desa, S. Prabhudesai, N. Dabholkar, V. Lamani, V. Manoharan, N. Naik, A. Mahalunkar, O. Naik, T. J. Thottam,

Dinesh Kumar P. K, and B. A. de Araujo. An integrated approach to study mud banks of alleppey kerala using the autonomous vertical profiler (AVP). In *2015 IEEE Underwater Technology (UT)*, pages 1–5, Feb 2015.

[12] Xiaole Xu and Canjun Yang. A pneumatic-driven airdropped portable underwater profiler for rapid deployment. In *OCEANS 2015 - MTS/IEEE Washington*, pages 1–5, Oct 2015.

[13] R. E. Davis, L. A. Regier, J. Dufour, and D. C. Webb. The autonomous lagrangian circulation explorer (alace). *Journal of Atmospheric and Oceanic Technology*, 9(3):264–285, 1992.

[14] Yushi Zhu, Xiaole Xu, Jun Wang, Canjun Yang, Qing Li, and Minjian Cai. A hybrid underwater profiler used for persistent monitoring. In *OCEANS 2015 - MTS/IEEE Washington*, pages 1–5, Oct 2015.

[15] V. Viswanathan and T. Taher. Buoyancy driven autonomous profiling float for shallow waters. In *OCEANS 2016 MTS/IEEE Monterey*, pages 1–6, Sept 2016.

[16] R. Madhan, G. Navelkar, Elgar Desa, Sanjeev Afzulpurkar, Shivanand Prabhudesai, Nitin Dabholkar, Antonio Mascarenhas, and Pramod Maurya. *Safety Aspects for Underwater Vehicles*, pages 10–16. Springer Berlin Heidelberg, Berlin, Heidelberg, 2010.

[17] J. Jouffroy, Q.-Y-Zhou, and O. Zielinski. On active current selection for lagrangian profilers. *Modeling, Identification and Control: A Norwegian Research Bulletin*, 34(1):1–10, 2013.

[18] N. A. Cruz and A. C. Matos. The MARES AUV, a modular autonomous robot for environment sampling. In *OCEANS 2008*, pages 1–6, Sept 2008.

[19] A. T. Morrison, J. D. Billings, and K. W. Doherty. The McLane moored profiler: an autonomous platform for oceanographic measurements. In *OCEANS 2000 MTS/IEEE Conference and Exhibition. Conference Proceedings (Cat. No.00CH37158)*, volume 1, pages 353–358 vol.1, 2000.

[20] J. Singleton, R. Bachmayer, and B. de Young. Development of a new autonomous underwater moored mobile profiler. In *2014 Oceans - St. John's*, pages 1–6, Sept 2014.

[21] A. Mascarenhas, A. Sanjeev, P. K. Maurya, L. Fernandes, R. Madhan, E. de Sa, N. Dabholkar, G. Navelkar, L. Naik, V. G. Shetye, N. B. Shetty, S. Prabhudesai, S. Nagvekar, and V. D. Seabed resident event driven profiling system (srep). concept, design and tests. In *2016 IEEE/OES Autonomous Underwater Vehicles (AUV)*, pages 273–277, Nov 2016.

[22] Teledyne Marine. Teledyne products | APEX Argo, 2017. Online; Acessed 29 January 2017.

[23] A. Amory and E. Maehle. Sembio - a small energy-efficient swarm auv. In *OCEANS 2016 MTS/IEEE Monterey*, pages 1–7, Sept 2016.

[24] Raspberry Pi Foundation. Raspberry pi, October 2016. Version 1.0.

[25] Gerald Coley. Beaglebone black system reference manual, May 2014. Revision C.1.

[26] Beagleboard. Beaglebone black wireless. "http://beagleboard.org/black-wireless", 2017. Online; Acessed 29 May 2017.

[27] BlueRobotics. T100 thrusters. Online; Acessed 12 April 2017, April 2017. http://docs.bluerobotics.com/thrusters/.

[28] Afro ESC. Afro esc 20a user manual, April 2017.

[29] InvenSense. Itg-3200 product specification. Datasheet, October 2010. Rev. 1.4.

[30] Analog Devices. Adxl345 digital accelerometer. Datasheet, February 2017. Rev.E.

[31] Amaryllo. Amaryllo roots - usb wired gps. Online; Acessed 12 April 2017, April 2017. http://www.amaryllo.com/almooj/car-products/amaryllo-roots-usb-wired-gps-receiver.html.

[32] TP-Link. Tp-link's 150mbps wireless n nano usb adapter tl-wn725n. Datasheet, 2017.

[33] Ocean Server. Bbdc-02r quick start guide, September 2008. Rev 1.2.

[34] Ocean Server. Ba-95hc-fl - intelligent battery and power system specifications, November 2009. Revision 2.0.

[35] Analog Devices. Low voltage temperature sensors tmp 36. Datasheet, August 2008. Rev. E.

[36] Texas Instruments. Max232x dual eia-232 drivers/receivers. Datasheet, November 2014.

[37] Conceptronic. 4-ports cube usb 2.0 hub. Datasheet, May 2017.

[38] Matt Welsh, Mathhias Kalle Dalheimer, and Lar Kaufman. *Running Linux*. O'Reilly & Associates, Inc., 1999.

[39] Blacklib. Blacklib beaglebone c++ library. "http://blacklib.yigityuce.com/index.html", 2017. Online; Acessed 5 May 2017.

[40] Robert Mahony, Tarek Hamel, and Jean-Michel Pflimlin. Nonlinear complementary filters on the special orthogonal group. *IEEE Transactions on automatic control*, 53(5):1203–1218, 2008.

[41] Fabio Varesano. Freeimu: An open hardware framework for orientation and motion sensing. *arXiv preprint arXiv:1303.4949*, 2013.

[42] BlueRobotics. Bluerobotics tsys01 library. "https://github.com/bluerobotics/BlueRobotics_TSYS01_Library", 2017. Online; Acessed 5 May 2017.

[43] TE Connectivity. Tsys01 digital temperature sensor. Datasheet, September 2015.

[44] TE Connectivity. Ms5837-30ba ultra small gel filled pressure sensor. Datasheet, March 2012.

[45] Ocean Server. Intelligent Battery and Power System^{TM} Firmware and Software User's Guide, July 2009.

[46] D Friel. Smart battery data specification, December 1998.

[47] National Marine Electronics Association et al. *NMEA 0183–Standard for interfacing marine electronic devices*. NMEA, 2002.

[48] RobertCNelson. cape-unversal-pwm example. "https://github.com/beagleboard/bb.org-overlays/blob/master/examples/cape-unversal-pwm.txt", 2017. Online; Accessed 20 May 2017.

[49] SimonK. Simonk firmware. "https://github.com/sim-/tgy", 2017. Online; Acessed 7 May 2017.

[50] BlueRobotics. Bluerobotics esc firmware. "https://github.com/bluerobotics/tgy/", 2017. Online; Acessed 7 May 2017.

[51] Mark Mitchell, Jeffrey Oldham, and Alex Samuel. *Advanced linux programming*. New Riders Publishing, 2001.

[52] elinux. Capemgr. ("http://elinux.org/Capemgr"), 2017. Online; Accessed 20 May 2017.

[53] Thor I Fossen. *Guidance and control of ocean vehicles*. Chichester ; New York : Wiley, 1994. Includes bibliographical references (p. [455]-474) and index.

[54] Bruno Siciliano and Oussama Khatib, editors. *Springer Handbook of Robotics*. Springer-Verlag Berlin Heidelberg, 2008.

[55] Thor I. Fossen. *Handbook of Marine Craft Hydrodynamics and Motion Control*. Wiley, 2011.

[56] John Nicholas Newman. *Marine hydrodynamics*. MIT press, 1977.

[57] Timothy Timothy Jason Prestero. *Verification of a six-degree of freedom simulation model for the REMUS autonomous underwater vehicle*. PhD thesis, Massachusetts institute of technology, 2001.

[58] B. Ferreira, M. Pinto, A. Matos, and N. Cruz. Hydrodynamic modeling and motion limits of AUV MARES. In *2009 35th Annual Conference of IEEE Industrial Electronics*, pages 2241–2246, Nov 2009.

[59] Thor I Fossen, Tor Arne Johansen, and Tristan Perez. *A survey of control allocation methods for underwater vehicles*. INTECH Open Access Publisher, 2009.

[60] Karl-Heinrich Grote and Erik K. Antonsson, editors. *Springer Handbook of Mechanical Engineering*. Springer-Verlag Berlin Heidelberg, 2009.

[61] Michael S. Triantafyllou. Maneuvering and control of surface and underwater vehicles. Lecture Notes for MIT Ocean Engineering Course 13.49, 1996.

[62] Edward V Lewis. Principles of naval architecture second revision. *Jersey: SNAME*, 1988.

[63] Frank M. White. *Fluid Mechanics*. McGraw-Hill Education, 2011.

[64] Ole Alexander Eidsvik. Identification of hydrodynamic parameters for remotely operated vehicles. Master's thesis, NTNU, 2015.

[65] T. I. Fossen and T. Perez. Marine systems simulator (mss). "http://www.marinecontrol.org", 2004.

[66] A. P. Aguiar and J. P. Hespanha. Trajectory-tracking and path-following of underactuated autonomous vehicles with parametric modeling uncertainty. *IEEE Transactions on Automatic Control*, 52(8):1362–1379, Aug 2007.

[67] Lionel Lapierre and Didik Soetanto. Nonlinear path-following control of an AUV. *Ocean Engineering*, 34(11–12):1734 – 1744, 2007.

[68] Bruno Ferreira, Miguel Pinto, Aníbal Matos, and Nuno Cruz. Control of the mares autonomous underwater vehicle. In *OCEANS 2009, MTS/IEEE Biloxi-Marine Technology for Our Future: Global and Local Challenges*, pages 1–10. IEEE, 2009.

[69] Jean-Jacques E Slotine, Weiping Li, et al. *Applied nonlinear control*. prentice-Hall Englewood Cliffs, NJ, 1991.