EXAMENSARBETE INOM MASKINTEKNIK,
GRUNDNIVÅ, 15 HP

STOCKHOLM, *SVERIGE 2021*

RoBuoy

Dynamic Positioning of an Autonomous Buoy using GNSS
Dynamisk Positionering av en autonom boj med GNSS

ERIK ANDERBERG

MARTIN OLANDERS

KTH
SKOLAN FÖR INDUSTRIELL TEKNIK OCH MANAGEMENT

RoBuoy

Dynamic Positioning of an Autonomous Buoy using GNSS
Dynamisk Positionering av en Autonom Boj med GNSS

ERIK ANDERBERG
MARTIN OLANDERS

Bachelor's Thesis at ITM
Supervisor: Nihad Subasic
Examiner: Nihad Subasic

TRITA-ITM-EX 2021:20

Abstract

Buoys anchored to the seabed are often used for marking courses in competitive sailing and other water sports, but they may need to be relocated several times per day. To avoid the time and fuel consuming labour of raising and moving the anchors, a prototype of an autonomous buoy using electric motors to maintain its position was built and tested.

The prototype buoy was built as a catamaran pontoon boat with one motor in each hull. To navigate it used a Global Navigation Satellite Systems receiver and a compass as sensors. Based on information from the sensors, a microcontroller regulated the buoy's heading and velocity using proportional, integral and derivative control.

The prototype was tested and evaluated in terms of design suitability, control system performance and dynamic positional precision. Except for leaking propeller axle seals the general design of the buoy was found suitable, as was the PID control system. However, while the GNSS position was sufficiently accurate when stationary, it would not register movement smaller than approximately 30 m. Consequently the buoy was only able to stay within 19.8 m of the target location on average. The performance may be improved by either using a different GNSS receiver, or upgrading to Real Time Kinematics GNSS.

Keywords: Autonomous, Buoy, GPS, GNSS, Robot, Mechatronics, Dynamic positioning

Referat

RoBuoy

Bojar förankrade till havsbottnen används ofta för att märka ut banan i kappsegling eller andra vattensporter, men kan behöva flyttas flera gånger per dag. För att undvika det tids- och bränslekrävande arbetet av att lyfta och flytta ankarna byggdes och testades en prototyp av en autonom boj som håller sin position med hjälp av elmotorer.

Prototypbojen byggdes som en pontonbåt i katamaranutförande med en motor i varje skrov. För att navigera använde den en mottagare för Global Navigation Satellite Systems och en kompass som sensorer. Baserat på information från sensorerna styrde en mikrokontroller bojens kurs och hastighet med proportionella, integrerande och deriverande regulatorer.

Prototypen testades och utvärderades med avseende på konstruktionens lämplighet, reglersystemets prestanda och dynamisk positioneringsprecision. Förutom läckande propelleraxelstätningar ansågs den generella designen var lämplig, likaså PID-reglersystemet. Även om den stillastående GNSS-positionen var tillräckligt exakt, så registrerades inte rörelser mindre än 30 m. Följaktligen kunde bojen bara hålla sig inom 19.8 m från målpositionen i genomsnitt. Prestandan skulle kunna förbättras med en annan GNSS mottagare eller genom att uppgradera till Real Time Kinematics GNSS.

Nyckelord: Autonom, Boj, GPS, GNSS, Robot, Mekatronik, Dynamisk positionering

Acknowledgements

Many people have helped us in the process of realising this project. First of all we are very grateful for the support and feedback from the other students doing their bachelor thesis projects in mechatronics at KTH in 2021. We also thank the master students Patrik Tiainen and Justus Conradi for inspiring discussions and helpful advice. Thanks to Jan Stamer for his advice and for letting us use the Production Engineering department workshop. We would also like to extend our gratitude to the sailing club Lidingö Jolleseglare for letting us use one of their boats when conducting initial testing, and to Nazar Netterström for lending us his drone to film the buoy. Staffan Qvarnström has helped us order components, and the assistant Malin Lundvall has been helpful in finding components and accessing workshops. So has assistant Amir Avdic, who also provided great support by sharing his experience as well as discussing ideas and problems. Finally we thank our supervisor Nihad Subasic for his informative lectures and supervision.

Contents

List of Abbreviations

3D	Three dimension
CB	Center of Buoyancy
CG	Center of Gravity
GNSS	Global Navigation Satellite System
GPS	Global Positioning System
I^2C	Inter-Integrated Circuit
PID	Proportional, Integral, Derivative
PP	Polypropylene
PWM	Pulse Width Modulation
RTK	Real Time Kinematics

List of Figures

List of Tables

Chapter 1

Introduction

This thesis is part of a degree project in Mechatronics at KTH, Royal Institute of Technology. The project includes electronics, mechanical parts and programming. Here the background, purpose, scope and method of the project are introduced.

1.1 Background

Buoys anchored to the seabed are often used in competitive sailing and other water sports to mark the course. They are commonly placed on the day of the event and retrieved at the end of it. If the course needs to be changed, due to for example a change in the wind direction, each buoy has to be manually retrieved and moved to its new position.

In deep waters the retrieval of the anchor to clear or reposition the course is hard and time consuming work which may delay the competition. As this often is done using petrol powered boats, it also has negative effects on the environment.

Having robotic, anchorless, buoys which can maintain a position and reposition themselves automatically would improve the working conditions of course setters, increase efficient race time and reduce the environmental impact of water sports.

1.2 Purpose

The purpose of this thesis is to develop an autonomous buoy that maintains a given position at sea, answering the following questions:

- What kind of design is suitable for this type of buoy?
- How can a control system be designed to ensure that the buoy navigates to and maintains a position?
- What level of dynamic positioning precision can be achieved?

1.3 Scope

As a bachelor thesis project, the construction and design of the RoBuoy prototype was limited by time and budget constraints. Therefore it does not consider some otherwise relevant factors such as:

- Seaworthiness - testing will only be done in calm waters.
- Endurance - enough endurance to conduct dynamic positioning tests will be considered sufficient.
- User interface - no user interface for setting the position of the buoy will be created.

1.4 Method

First, material from similar projects and information on relevant topics was gathered for design ideas. Then the hull was built using pipes, wood and 3D-printed components. Next, the propulsion and navigation systems were constructed within the hull, and the construction was waterproofed. A program for the buoy to navigate to and keep its position was written and uploaded to the microcontroller running it, whereafter testing was done at sea to evaluate its performance

Chapter 2

Theoretical Background

This project requires some theoretical understanding of design of watercraft, navigation and control theory, which are outlined here.

2.1 Design of Small Watercraft

An object submerged in water is affected by a buoyancy force (F_B) proportional to the mass of the displaced water. The stability of the object is determined by the relationship between the gravitational force (F_G) and the buoyancy force, or between the center of gravity (CG) and the center of buoyancy (CB). In an equilibrium state CG and CB on the same vertical line. When the object is disturbed CB will move, resulting in a moment (M) on the object as shown in Figure 2.1. For a stable object this moment will turn the object back towards the equilibrium state [1].

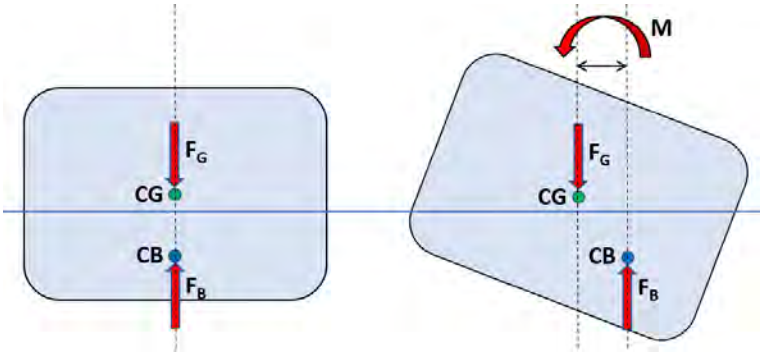


Figure 2.1: Hydrostatic stability and righting moment. Created using Microsoft Powerpoint.

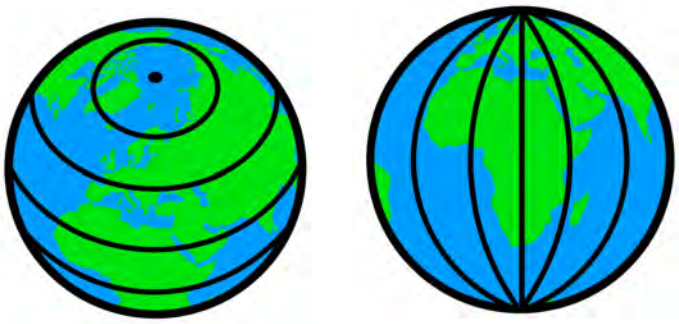
Apart from stability, two other important features of an autonomous surface vessel are the ability to keep the electronics dry, and to keep its heading straight while travelling through water. This can be accomplished by designing the hull as a catamaran pontoon boat with a waterproof box for the electronics [2]. If driven by

a propeller, the vessel will be subject to a sideways force known as the propeller effect, in a direction depending on which way the propeller turns [3].

A method for constructing a waterproof hull is to use polypropylene (PP) tubes with custom made ends, sealing joints and holes for wires with silicone sealant. Using silicone tape to seal propeller axles has proved to work for a short duration, but is not ideal for prolonged operation [4].

2.2 Navigation

Navigation means determining one's position, course and speed. To define the position a global coordinate system of latitudes (horizontal lines) and longitudes (vertical lines) is used. The Northern and Southern hemispheres have 90 latitudes each, making the total of 180 latitudes equivalent to the 180 degrees from the South Pole to the North Pole. Similarly, the Eastern and Western hemispheres have 180 longitudes each, which together are equivalent to Earth's 360 degrees, as shown in Figure 2.2. To make it more precise each degree is divided into 60 minutes, and every minute is divided into tenths. However, the degrees may also directly be divided in decimals. At sea level one minute is equal to one nautical mile (M), or 1852 meters. A position with an error of less than two meters can then be written using degrees and minutes with three decimals on the following form: 59°20.892'N 18°04.128'E, which in decimal form would be 59.34820°N, 18.06880°E [3].



(a) Latitudes of Earth. (b) Longitudes of Earth.

Figure 2.2: Latitudes and longitudes. Created using Pages.

2.2.1 Compass

The heading of a vessel is the direction it is pointing. It is measured as the angle between the direction to the north pole and where the vessel is pointing, H in Figure 2.3, often using a compass. The bearing to a position is the angle between the direction to the north pole and a line between the vessel and said position, B in Figure 2.3. Finally the direction D to a position is measured in the vessel's own

reference system as the angle, 0° to 180° to starbord or port, between its heading and bearing [3].

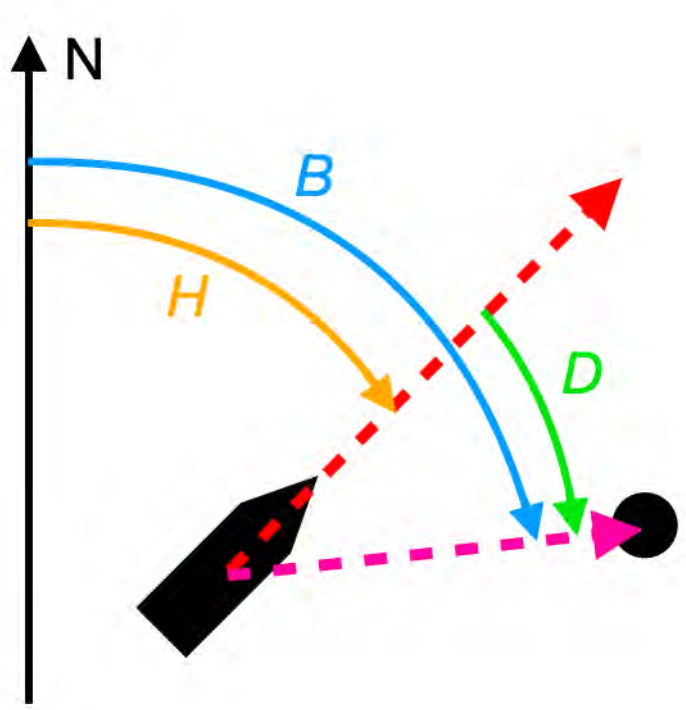


Figure 2.3: Heading (H), bearing (B) and direction (D). N represents north. Created using Pages.

A magnetic compass however, is affected by two main errors, magnetic declination and deviation. Magnetic declination comes from the positional difference between the geographic and the magnetic poles, caused by the movement of the magnetic poles over time. This means the magnetic declination depends on one's position, and can be compensated for using known data on the declination in that area [3]. The declination in Stockholm, Sweden, was 6.85° in April 2021 [5]. Deviation is caused by magnetic or electromagnetic objects, such as engines or electrical circuits, interfering with the compass. The deviation therefore depends on a vessel's heading, and can be compensated for using small compensatory magnets or a table with that vessel's known deviation for each heading [3].

2.2.2 Global Navigation Satellite Systems

Global Navigation Satellite System, GNSS, is a term for systems using satellites to determine a position on Earth. Examples include the American Global Positioning System, GPS, Russian Global Navigation Satellite System, GLONASS, European

Galileo and Chinese BeiDou. Although all these systems are somewhat different from each other, their general principle can be described as follows [6].

A system of satellites, the positions of each being known, broadcast unique coded signals. As these travel at the speed of light, a receivers distance from a satellite is easily calculated if they are both synchronised in time. At a given point in time the signal from a satellite forms a sphere centered around it. To calculate the position of a receiver, the intersection of three such spheres is required. The satellites are synchronised in time to each other but a receiver is often not, only having a simple clock set approximately to the same time as the satellites. To find the delay of the receiver clock a fourth signal is required [6].

When the position of the receiver is known, it is easily translated into latitude, longitude and height. As GNSS only determines a position, a single receiver is incapable of determining a heading [6].

The precision of GNSS may be improved using Real Time Kinematics (RTK) GNSS. This utilises a base station which sends corrections to the mobile receiver, greatly improving its accuracy [7]. Several countries are now covered by network-RTK, meaning the user does not need to set up such a base station, but rather can rely on a pre-existing coverage of correction signals [8]. Using RTK GNSS, the accuracy of a receiver can be increased to a few centimeters [7] which is the precision for RTK-fixed coordinates, or, in mobile applications, at least one meter which is the precision for floating RTK coordinates [9].

2.3 Dynamic Positioning

A vessel on water that utilises dynamic positioning is maintaining its position and heading or staying on a predefined track by only using thrusters. It is common for control systems of dynamic positioning vessels to use GNSS as a position reference system, and also sensors such as gyros and motion reference units. For control, proportional, integral and derivative (PID) controllers are commonly used, together with advanced models of the vessels behaviour [10].

2.4 Control Theory

The purpose of a controller is to make a system behave in a particular way despite disturbances. This is commonly achieved using PID regulated feedback control. In feedback control, the resulting output value of the system is input to a regulator and compared to the desired value. Then the system is given an input depending on the difference between these, the error. With a PID controller this input signal depends proportionally on the current error, the historic error as well as the predicted error [11].

If the error between the desired and current output is denoted e, the input to the system u and time t, the following equation describes the PID regulator.

$$u(t) = K_P e(t) + K_I \int_{t_0}^{t} e(\tau)d\tau + K_D \frac{de(t)}{dt} \tag{2.1}$$

Here, the constants K_P, K_I and K_D are the gains for each part of the controller. The proportional constant K_P acts on the present error, the integration constant K_I acts on the integral of historic errors and the derivation constant K_D the predicted error via its derivative [11].

2.4.1 Evaluation of Controls Systems

One way of evaluating a control system is to observe its response to a sudden change of the desired value, the step response. If the new value is not reached, that means there is a static error. To measure the speed of the system the settling time or rise time can be calculated. Settling time is defined as the time it takes for the system to reach and stay within 5% of the new desired value while rise time is defined as the time it takes for the system to go from 10 % to within 90 % of the new desired value. Furthermore, the overshoot of a step response indicates in percent how far the response surpasses the new desired value [11].

2.4.2 Differential Drive Robots

A differential drive robot is a robot with two driven wheels and one caster wheel for stability. The driven wheels share axle but are independently controlled. Thus, both the direction and velocity of the robot depend on the angular velocities of the two wheels. To make it move to a location a PID controller can be used to control the heading of the robot, and a simple proportional controller dependent on the present distance to the target to control the velocity [12].

When a differential drive robot is controlled that way, the controller output is a reference velocity and reference angular velocity. These may be denoted v_{ref} respectively ω_{ref}. To be useful to the motors, this information is transformed into reference angular velocities for the wheels on each side, denoted $\omega_{R,ref}$ and $\omega_{L,ref}$ for the right and left side. That is done using a transformation matrix

$$M = \frac{1}{r}\begin{bmatrix} 1 & \frac{l}{2} \\ 1 & -\frac{l}{2} \end{bmatrix} \tag{2.2}$$

in which r is the wheel radius and l is the distance between the wheels. This transformation matrix satisfies the equation

$$\begin{bmatrix} \omega_{R,ref} \\ \omega_{L,ref} \end{bmatrix} = M \begin{bmatrix} v_{ref} \\ \omega_{ref} \end{bmatrix} \quad \text{[13].} \tag{2.3}$$

Chapter 3

Demonstrator

To answer the questions in 1.2 Purpose, a demonstrator was constructed. First its physical design, then electric circuit and finally software are presented here.

3.1 Design

The general design of the buoy is visible in Figure 3.1. The hull was designed as a pontoon boat for stability and maneuverability reasons as mentioned in 2.1 Design of Small Watercraft, with a length of 680 mm and a width of 410 mm.

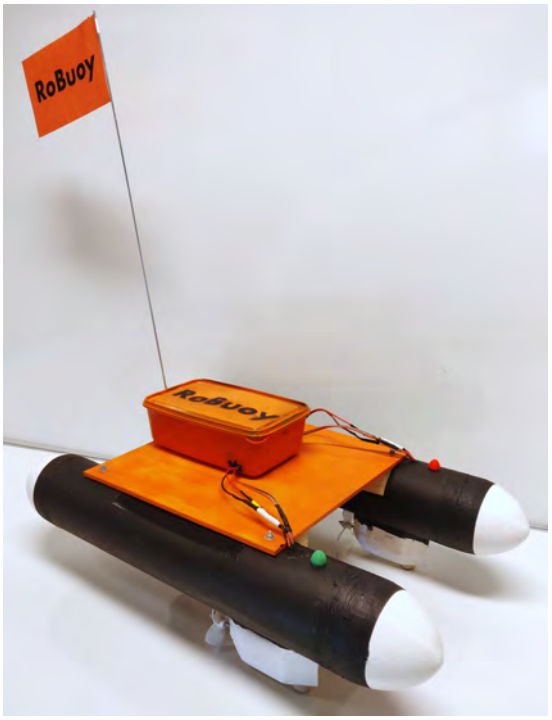


Figure 3.1: Overview of the design of the RoBuoy. Taken by authors.

The pontoons were built using PP pipes of 110 mm diameter with 3D-printed end cones, and were screwed to a wooden board. On the board a plastic box was attached as a waterproof container for the electronics. The board and the box were painted bright orange, and a flag was added at the rear of the electronics box for improved visibility.

The motors were placed in 3D-printed gondolas attached under the pontoons. The propellers were placed behind the gondolas to protect them in case the buoy ran aground. The gondolas were placed as centrally as possible as that was believed to result in good maneuverability, but had to be moved slightly forwards as the wires otherwise would interfere with the placement of the batteries. As the batteries were heavy, they were placed centrally inside the pontoons for stability. To access the batteries for charging and to inspect the interior of the hulls, a hole was cut in the upper part of each hull. To inspect the motor gondolas for water intrusion, holes were drilled in them which subsequently were plugged by screws. Photographs of the buoy from the front, side and top are available in Appendix A.

3.1.1 Waterproofing and Compartmentalisation

The hull had several points susceptible to water intrusion. These included the joints between the PP-pipes and 3D-printed components, access holes drilled and cut in the hull, as well as the propeller axle's protrusion through the hull. Most of these places were stationary and needed to be permanently sealed, however, the propeller axles needed to be able to turn and the large access holes needed to remain openable.

Waterproofing of all stationary parts was achieved using silicone sealant, which was applied to all joints and holes. Compared to the waterproofing of propeller axles mentioned in 2.1, an improved design was used. The propeller axles passed through two walls to reach the interior of the motor gondola as shown in Figure 3.2. The holes in both walls were lined with silicone sealant making them almost waterproof. To further decrease the risk of water leaking in, the space between the two walls was filled with industrial grease.

Figure 3.2: Gondola with motor and propeller axle. Taken by authors.

As the large access holes were above the water surface, they did not need to be

resistant to constant submersion, but only spray and occasional waves covering them. Therefore they were waterproofed using duct tape.

To minimise the risk of sinking if leakage would occur, each hull was divided in watertight compartments. The holes for electrical wires to the motor gondolas were sealed with silicone sealant, meaning that if the axle seal failed, the rest of the hull would not be filled with water. Inside the PP-tubes, a bulkhead was added, minimising the risk of sinking if water penetrated the hull.

3.1.2 Propellers

The propellers used were 3D-printed in a style similar to those found on small boats. The model for 3D-printing was created using OpenSCAD with a design based on *Parametric ROV thruster propeller* [14]. That design was modified in size to fit this project. Furthermore, the hole for the propeller axle was edited to be rectangular, ensuring proper torque transfer.

The outer ends of the propeller axles were threaded and filed flat to fit in the rectangular holes of the propellers. The propellers were bolted to the axle whereafter the nuts were glued in place. The propeller design and its connection to the axle without glue is visible in Figure 3.3.

Figure 3.3: A propeller and its connection to the axle. Taken by authors.

In order to minimise the sideways drifting from the propeller effect mentioned in 2.1 Design of Small Watercraft, the two propellers were printed as mirrored versions of each other. This way they would turn in opposite directions while providing thrust in the same direction, so that the the sideways forces cancel each other out.

3.2 Electronics

This section describes the functions and specifications of the electronic components used for navigation, communication and maneuvering of the buoy. The circuit

connections are shown in Figure 3.4. The electronics are centered around a microcontroller, receiving information from a compass and GNSS module. It controls two motors via H-bridges and communicates wirelessly through a Bluetooth module.

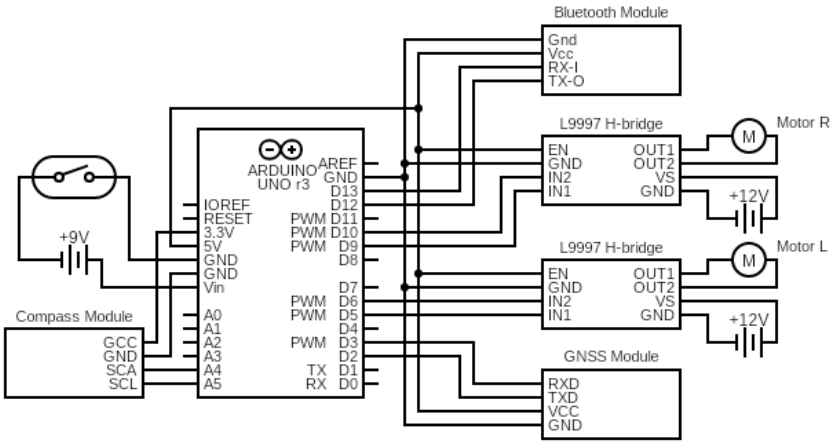


Figure 3.4: Electrical circuit. Created using Circuit Diagram.

All components except the motors and their batteries were attached inside the plastic box, as shown in Figure 3.5. Two holes were drilled and sealed for the wires to the motors and batteries, as was a smaller hole for a power switch easily accessible from the outside.

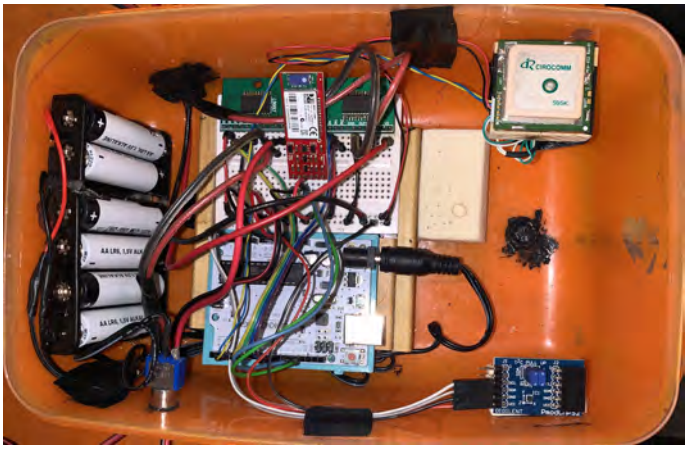


Figure 3.5: Electronics Overview. Taken by authors.

3.2.1 Microcontroller

To control all other components a Playknowlogy Uno Rev. 3 microcontroller, shown in Figure 3.6, was used, which is based on the Arduino Uno design and thus could be programmed using the Arduino standard software [15]. It was powered by a battery-pack of six 1.5 V AA batteries, supplying a total of 9 V to the microcontroller through a 5.5x2.1 mm contact.

(a) Playknowlogy Uno Rev. 3 microcontroller [15].

(b) Battery-pack. Taken by authors.

Figure 3.6: The microcontroller and battery pack used.

3.2.2 Sensors

For GNSS positioning a Parallax PMB-648 GPS Module, shown on the left in Figure 3.7 was used. It had a Cirocomm 595K patch antenna and a 1 Hz navigation update rate with a 5 m specified precision [16].

A Digilent Pmod CMPS2 3-axis anisotropic magneto-resistive sensor, visible to the right in the figure below, was used as a compass. It uses Inter-Integrated Circuit (I^2C) communication and should give a heading with 1° resolution with the manufacturer-supplied example code. [17].

(a) The PMB-648 GPS Module. Taken by authors.

(b) The Pmod CMPS2 compass module [17].

Figure 3.7: The two sensor modules.

3.2.3 Propulsion System

For propulsion two DME34BA 12 V electric motors from Japan Servo, visible in Figure 3.8, were used. Propeller axles, protruding out through the motor gondolas, were attached to the motor axles using custom made aluminum joints.

Figure 3.8: One of the two motors used for propulsion. Taken by authors.

The motors were powered by one Ultracell UL2.4-12 V battery each, visible to the left in Figure 3.9, placed centrally in each of the two hulls. They were connected to their respective motor via L9997 H-bridges, which can be seen to the right in Figure 3.9. The H-bridges were controlled by the microcontroller using Pulse Width Modulation (PWM).

(a) One of the batteries used to power the motors [18].

(b) An L9997 H-bridge. Taken by authors.

Figure 3.9: One of the batteries and H-bridges used in the propulsion system.

3.2.4 Bluetooth Module

For wireless communication with a computer, a SparkFun Bluetooth Mate Gold module was used, visible in Figure 3.10. It is designed to work with Arduino and uses serial communication to send data over distances up to 100 m [19].

Figure 3.10: The SparkFun Bluetooth Mate Gold module used. Taken by authors.

3.3 Software

The microcontroller software program structure consisted of a main program with a setup part initialising communication with components, and a loop with function-calls to get sensor values, perform calculations and set motor voltage. A flowchart outlining the algorithm is visible in Figure 3.11.

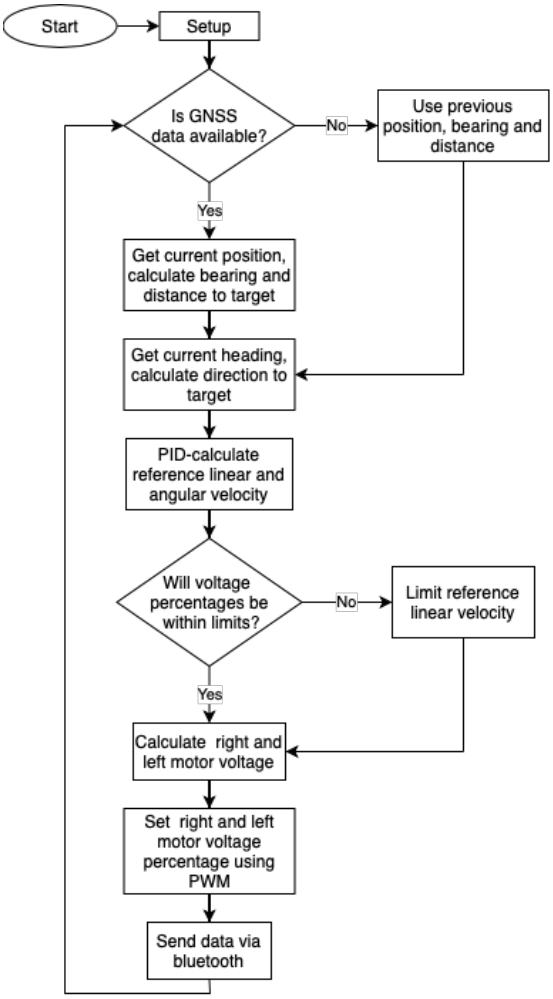


Figure 3.11: Flowchart of the program for the buoy. Created using draw.io.

First, the current position of the buoy, if available, was retrieved from the GNSS module. That information was then used to calculate the distance and bearing to a predefined target position. Thereafter, the current heading of the buoy was retrieved from the compass module. The current heading and bearing to the target position were used to calculate the direction to the target position.

The distance and direction to the target position were then used as inputs to PID controllers controlling the buoy's linear and angular velocity. Subsequently the controller outputs were combined to calculate the voltage percentage for each motor, after which the motors via the H-bridges were set to the corresponding voltage using PWM. The code for the main program can be found in Appendix B.1.

3.3.1 GNSS

The SoftwareSerial library for Arduino was used to create a connection with the GNSS-module. Then the TinyGPS library was used to communicate with it. If a position fix was retrieved, the bearing and distance to the target position were calculated using functions supplied in the library. All this was done in the function getdata_GNSS, available in Appendix B.2

3.3.2 Compass

As the compass used I^2C serial communication, the Arduino Wire library was used to support communication with it. An example code was supplied by the manufacturer of the compass module, however, this code was broken and had to be modified in the following ways:

- Unsigned integers sent from the module were treated as signed integers, resulting in overflow for large sensor values.
- Inefficient calculation of compass heading based on data from sensors was updated.
- Continuous calibration steps which became unnecessary after fixing the problems above were removed.

The modified version used in the demonstrator returned the heading of the compass in degrees, and is available in Appendix B.3. It measures the magnetic field twice with opposite polarity, thus being able to eliminate the offset created by the device itself. Having done that, the horizontal components of the magnetic field are used to calculate the heading of the compass in relation to its x-axis. Finally, this heading was adjusted for declination by adding the declination in the Stockholm area, 6.85°, as testing was done there. Thus the heading of the compass was calculated, however, as the compass module was mounted reversed in the buoy, 180° were removed to calculate the heading of the buoy.

3.3.3 PID Controllers

The direction and distance to the target position were used as inputs to separate PID controllers controlling the buoy's linear and angular velocity, similar to differential drive robots described in 2.4.2 Differential Drive Robots. The controllers were created with the PID_v1 library for Arduino and tuned manually. The output of the distance controller was a reference velocity, v_{ref}, and the output of the heading controller was a reference angular velocity, ω_{ref}.

Defining direction as 0° to 180° starboard or port as presented in 2.2.1 Compass makes it difficult to implement in a mathematical controller. Therefore direction was redefined as the angle between the opposite of the buoy's heading and the bearing to the target position, D in Figure 3.12. Then the heading controller was set to keep that value at 180°, meaning the buoy turned right when the target was to the right of it, and left when the target was to the left of it. The code for the controllers was placed in the main program, available in Appendix B.1.

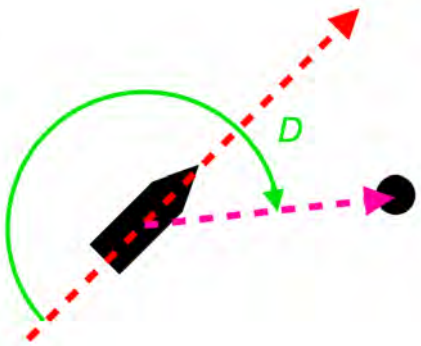


Figure 3.12: Direction (D) to target position as defined in the code. Created using Pages.

3.3.4 Power Allocation

The power for each motor, or more specifically the voltage percentage, U_L and U_R for the left respectively right motor, was calculated in the function getU, available in Appendix B.4. They were calculated using the outputs of the PID controllers, v_{ref} and ω_{ref}, as well as the minimum voltage percentage for the motors to turn, 27% called U_{min}, as

$$\begin{bmatrix} U_R \\ U_L \end{bmatrix} = M \begin{bmatrix} v_{ref} \\ \omega_{ref} \end{bmatrix} + \begin{bmatrix} U_{min} \\ U_{min} \end{bmatrix} \tag{3.1}$$

using the matrix

$$M = k \begin{bmatrix} 1 & \frac{l}{2} \\ 1 & -\frac{l}{2} \end{bmatrix} \quad . \tag{3.2}$$

Here, l is the distance between the propeller axles, 0.30 m. This is similar to the control of differential drive robots. However, as the dynamics of the buoy's linear and angular velocity depending on motor voltage was not studied, an equally precise model could not be used. Therefore the constant $k = 1$ s/m was added to M, making the voltage percentages dimensionless.

The output of both controllers was limited to ensure neither of their outputs alone would result in exceeding the available $\pm 100\%$ of voltage. As they might do so combined, v_{ref} was limited so that the difference between U_R and U_L always was $l\omega_{ref}$ to ensure prioritisation of correct heading.

3.3.5 Setting Motor Voltage

To control the voltage for the motors a function setMotor, available in Appendix B.5, was used. It takes two parameters, for motor choice (right or left) and voltage percentage (-100 to 100, negative for reverse). It sets the motor voltage by sending a PWM-signal corresponding to that percentage to the H-bridges from the pin for the desired motor and direction.

3.3.6 Communication

To communicate with the buoy while it was running, the Bluetooth module was used, utilising the SoftwareSerial library for Arduino and the Communications Toolbox for MATLAB. The module continuously sent data via Bluetooth to a laptop computer, which was running a MATLAB script to retrieve, save and analyse the data. The data being transmitted included current navigational information, motor voltage percentage and time elapsed since the program started. Bluetooth communication was part of the main program, thus the code for it is found in Appendix B.1. The MATLAB program is found in Appendix C.1.

Chapter 4

Experiments

The experiments performed to answer the three questions of the project are described in this chapter.

4.1 Buoy Design

The following experiments were performed to find out whether the design of the design of the buoy was suitable.

1. The motor gondolas were placed in water with the motors running for an extended period of time to test their water integrity.

2. The buoy was placed in water to test whether it floated and remained upright despite small disturbances.

3. The buoy was placed in water for an extended period of time while the motors were running. Then all access points (openings for batteries in hulls and water inspection screws in motor gondolas) were opened and electrical components tested to find if any water had penetrated the hull.

4.2 Control System Design

As the control system consisted of separate PID controllers for heading and distance, these were tested independently to as great extent as possible.

First the heading controller was tested by measuring its step response. That was done by disabling the distance controller to make the reference linear velocity v_{ref} be 30% of maximum velocity, and overwriting the calculated bearing with a set value. That value was then changed by 90° after 15 seconds. The heading of the buoy was sent via Bluetooth to a computer plotting it in MATLAB as function of time. This was evaluated five times for the buoy turning in either direction.

As decreasing the distance to a target position requires going in the correct direction, the heading of the buoy must be updated to evaluate the distance con-

troller. Thus the distance controller was evaluated using the same experiment as for dynamic positioning precision.

4.3 Dynamic Positioning Precision

To test the the prototype's level of precision in dynamic positioning it was given the coordinates for a target point, which it then tried to reach for five minutes. While attempting to reach the target position it continuously sent data including its perceived position, distance to target, bearing and heading. When the time limit was reached it got a new target location on land to make it return to shore. The navigational data was saved for analysis. To evaluate its performance the distance to target over time and the perceived position compared to the target position were studied.

Chapter 5

Results

Here the results of the experiments are presented.

5.1 Results of Buoy Design Experiments

The results of the three experiments performed to evaluate the design are presented in the table below.

Table 5.1: Table of results when testing the design of the buoy.

Test	**Result**
1	Water penetrated neither the propeller axle seal nor other parts of the gondola.
2	The buoy floated and remained upright.
3	Water was found in the motor gondolas, but not the main hull. All components worked well in these conditions.

5.2 Results of Control System Experiments

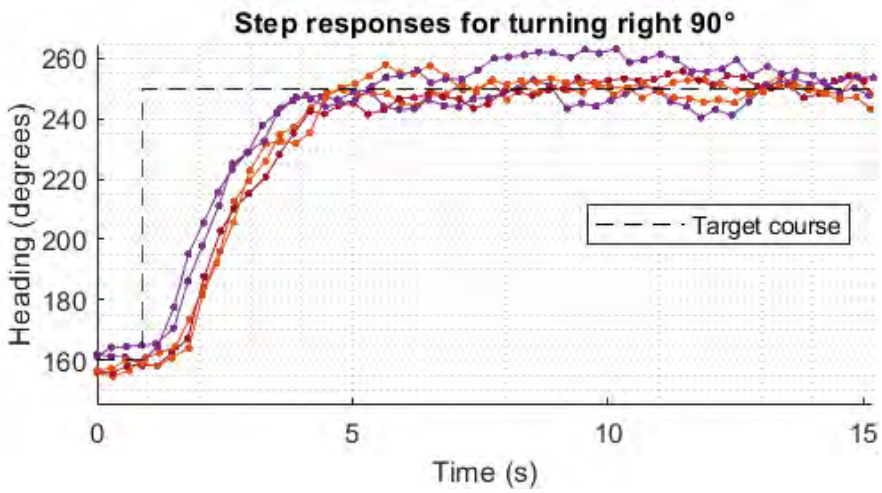
With the PID parameters set according to Table 5.2 the step responses of the buoy's heading when moving with a constant reference velocity was plotted as in Figure 5.1, with average rise time and overshoot presented in Table 5.3. When the buoy had settled on a course, its reported heading fluctuated by about 5°.

Table 5.2: Table of parameters in the course PID controller.

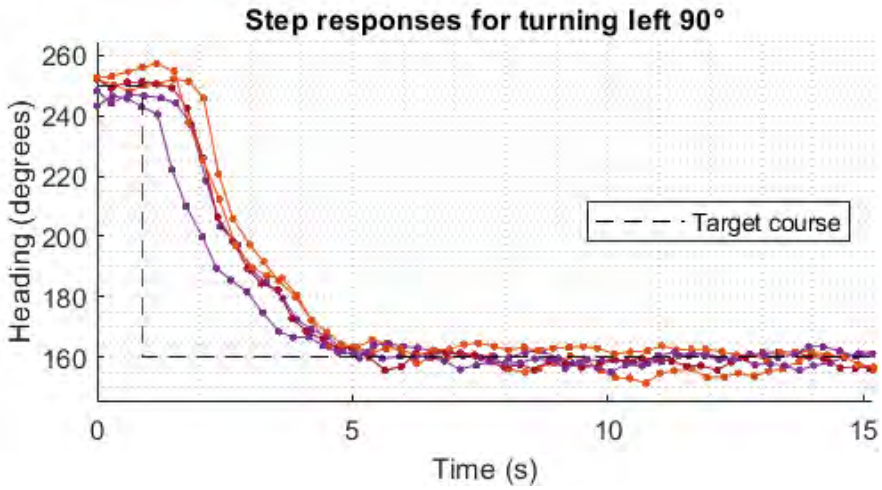
$\mathbf{K_P}$	$\mathbf{K_I}$	$\mathbf{K_D}$
7	0.9	3.2

Table 5.3: Step response results from 90° turns.

	Left	**Right**
Average overshoot	4.7%	7.3%
Average rise time (s)	2.4	2.3



(a) Step response for turning 90° right.



(b) Step response for turning 90° left.

Figure 5.1: Step responses of the buoy's heading when turning. Created with MATLAB.

5.3 Results of Dynamic Positioning Experiments

Using the PID parameters for the course controller mentioned above, and the value $K_p = 2$ for the distance controller, others set to zero, the buoy reported its position as in Figure 5.2 where each dot represents its perceived position.

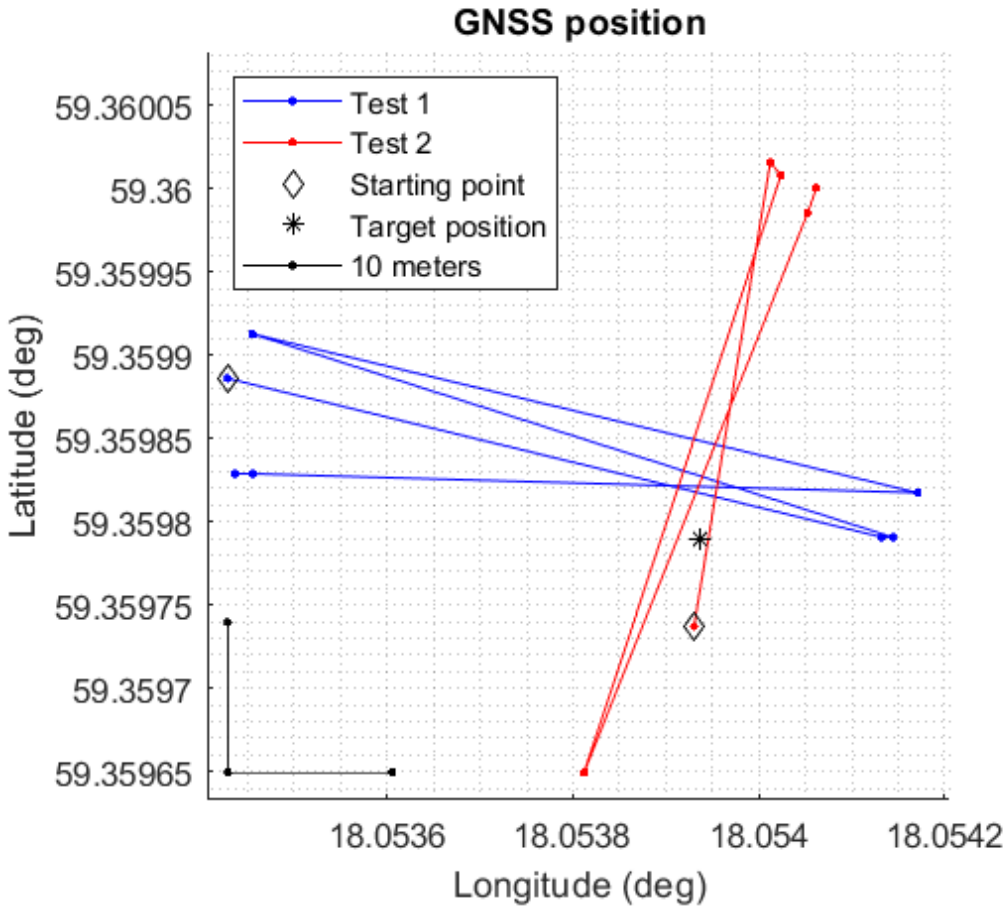


Figure 5.2: Reported GNSS position over time compared to target position. Created with MATLAB.

The number of data points from of each test run differs due to sudden disruptions of the Bluetooth connection, sometimes resulting in a loss of data at the start or end of each test run. When observing the buoy it moved in straight lines, approximately 30 m long, back and forth over the target position.

The reported distance to the target position over time is plotted in Figure 5.3. During the two tests the buoy was able to stay within 31 m of the target position, and the reported average distance was 19.8 m.

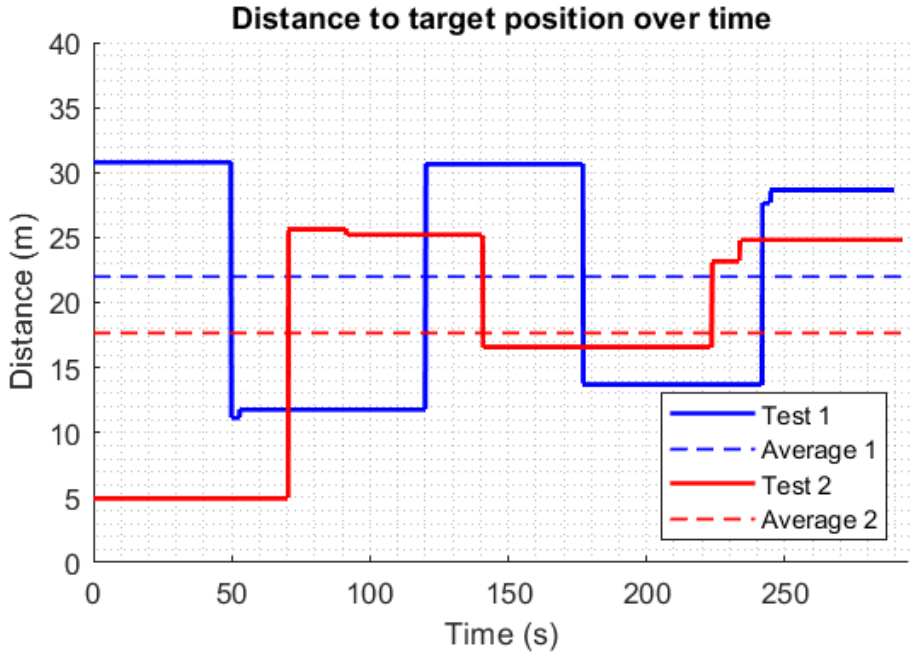


Figure 5.3: Reported distance to target over time. Created with MATLAB.

Chapter 6

Discussion and Conclusions

The evaluation of and conclusions drawn from the results are presented in this chapter.

6.1 Buoy Design

The suitability of the design of the buoy can be evaluated not only from the results of the design experiments but also from its performance in subsequent experiments. The buoy was not only stable and floated, it was also very maneuverable as it could turn 90° with a rise time of less than 2.5 s and easily travelled across the water. As all of this performance was satisfactory, it may be concluded that the general design of the prototype is suitable for this kind of buoy.

However, although improved compared to the source material, the propeller axle seals failed to keep water out of the motor gondolas. The motors continued to work despite this, but the water intrusion probably decreased both their lifespan and efficiency. Therefore the design of the axle seals is not suitable and needs to be improved. Furthermore, in higher speeds the prototype tended to tilt slightly forward, likely because of the increasing forces on the motor gondolas from water resistance. This could probably be solved by modifying the shape of the gondolas to make the force from the water resistance also result in a upwards lift force.

6.2 Control System Design

The step response of the buoy when subjected to a 90° change of target heading was considered to be sufficiently fast, especially when compared to the 1 Hz navigation update of the GNSS module. As the heading of the buoy fluctuated by about 5° even after settling, the settling time could not be calculated for the maneuver. However, as the overshoot in most cases was limited, the rise time is a reasonable measurement of the speed of the system.

The step response is only a way of measuring the general performance of the buoy as it is not expected to perform exactly that maneuver more often than others.

Based on the step responses it can be concluded that the buoy neither has a static error nor trouble with instability when adjusting its heading. There are slight differences in overshoot and rise time between right and left turns, both probably due to the few measurements. The rise times are quite similar and the larger overshoot when turning right is probably much due to the abnormal overshoot of the purple line. The fluctuation around the desired course after settling is considered acceptable, as the differences are relatively small and the buoy maintains a correct average course.

Although the intention was to conduct experiments on calm waters, small waves were often present. Therefore the small disturbances after settling may be due to waves, meaning the step response is not only the result of the 90° change of target course, but also the response to the waves. The effect of the waves might have been augmented by the fact that the compass module was sensitive to tilting, and would report an inaccurate heading if not horizontal. Another source of error might be that deviation was not accounted for when programming the compass, meaning the measured heading of the buoy was not entirely correct. Although a possible area of improvement, the error from this is probably limited as the buoy behaves as expected with respect to course in subsequent tests.

The performance of the distance controller was hard to evaluate as the precision of the position measurements were too inaccurate to use for this control system. When close to the target position the buoy overshoots it, as the reported position is not updated until it has passed it. Either the 1 Hz navigation update rate or the 5 m accuracy specified by the manufacturer of the GNSS module is false, or not achievable when conducting the experiments. As the buoy after having overshot the target position rapidly turns towards it, the performance of the distance controller was not really evaluated. Had it not behaved that way, there would be a greater risk of instability when keeping a position, but this way the stability of the distance controller alone becomes irrelevant. Since it sometime takes over a minute for the GNSS reciever to report a new position, it seems unlikely that the buoy overshoots the target because it travels too fast, but rather because a certain distance is necessary to be traversed for the receiver to report a new position.

Using separate controllers for heading and distance proved to be a satisfactory solution for this buoy as it easily maneuvered to a position and stayed there as well as it could. However, as mentioned above, its performance when close to the target position could not be evaluated.

6.3 Dynamic Positioning Precision

Due to the unpredictability of the Bluetooth connection the number of measurements from each test was effectively limited. However, visual observations indicated that the patterns shown in the results are representative for the entire test runs. The level of precision in dynamic positioning achieved by the buoy was that it on average could remain within 19.8 m of the target position, sometimes driving al-

most 35 m away before turning back. When considering its intended application, marking courses in water sports, that precision is deemed unsatisfactory. The main reason for this inaccuracy is probably the GNSS receiver, as discussed above.

As the buoy needed to adjust its heading to maintain its position, it would not be fully capable of dynamic positioning, where both heading and position are kept permanent. That was however not considered an issue, as the heading of a buoy normally is irrelevant. Rather, it would possibly be an advantage that the buoy only moves forwards and not sideways, as its resistance in the water is lower in that direction.

6.4 Recommendations and Future Work

The physical design of the buoy can mainly be improved by better waterproofing of the propeller axles, for the reasons mentioned above in 6.1.

As the buoy did not achieve a desirable level of precision when keeping a position, this would need to be improved. Since the greatest limiting factor was the weak performance of the GNSS module, an upgrade to a better module is necessary. The question still stands whether another traditional GNSS module would be able to offer improved performance, or if an RTK GNSS module would be necessary. Another option for improved positional performance would be to combine the present GNSS module with another system for determining how far the buoy has traveled. That could be either using dead reckoning based on a model of the buoy's speed depending on reference linear velocity, v_{ref}, or adding an accelerometer to integrate the acceleration of the buoy twice to find how it has moved since the latest position fix. The former system would not be able to compensate for drifting due to current or wind, and the latter is likely to be affected by waves.

Since the project was limited to testing in calm waters only, the endurance of the buoy was not investigated and a user interface was not created, all of these areas could be expanded on. If the buoy should handle rougher seas, the waterproofing of the electronics box and battery access holes would probably need to be improved. The endurance of the buoy was sufficient for the testing conducted but may need some improvement, depending on intended application. Finally a user interface to control the position of several buoys would be necessary for them to be easily used in water sports. To make the most of this potential, another form of communication would probably be suitable, as the 100 m range of the Bluetooth module may be too short and unreliable.

Bibliography

[1] B. Barrass and D. Derrett. *Ship Stability for Masters and Mates*. 6:th ed. Elsevier, 2011.

[2] H. Armstrong, J. Bryan, K. Muir and J. Rothe. "Autonomous Surface Marker Buoy". In: *30th Florida Conference on Recent Advances in Robotics May 11-12, 2017*. 2017.

[3] L. Carlsson. *Navigation för fartygsbefäl klass VIII*. 2:nd ed. Jure förlag AB, 2011.

[4] V. Hanefors and S. Rahmanian. "Sharkbait - a self-stabilising underwater drone". Bachelor's thesis. KTH, Royal Institute of Technology, 2019.

[5] National Centers for Environmental Information. *Magnetic Field Calculators - Magnetic Declination Estimated Value*. Apr. 21, 2021. URL: https://www.ngdc.noaa.gov/geomag/calculators/magcalc.shtml#declination.

[6] B. Hofmann-Wellenhof, H. Lichtenegger and E. Wasle. *GNSS – Global Navigation Satellite Systems GPS, GLONASS, Galileo, and more*. SpringerWienNewYork, 2008.

[7] The Swedish Mapping, Cadastral and Land Registration Authority. *RTK*. May 6, 2021. URL: https://www.lantmateriet.se/en/maps-and-geographic-information/gps-geodesi-och-swepos/GPS-och-satellitpositionering/Metoder-for-GNSS-matning/RTK/.

[8] The Swedish Mapping, Cadastral and Land Registration Authority. *Network RTK*. May 6, 2021. URL: https://www.lantmateriet.se/en/maps-and-geographic-information/gps-geodesi-och-swepos/GPS-och-satellitpositionering/Metoder-for-GNSS-matning/Natverks-RTK/.

[9] K. M. Ng et al. "Performance Evaluation of the RTK-GNSS Navigating under Different Landscape". In: *2018 18th International Conference on Control, Automation and Systems (ICCAS)*. 2018, pp. 1424–1428.

[10] A. J. Sørensen. "A survey of dynamic positioning control systems". In: *Annual Reviews in Control* 35.1 (2011), pp. 123–136. ISSN: 1367-5788. DOI: https://doi.org/10.1016/j.arcontrol.2011.03.008.

[11] T. Glad and L. Ljung. *Reglerteknik Grundläggande Teori*. 4:15:th ed. Studentlitteratur, 2006.

[12] S. Armah, S. Yi and T. Abu-Lebdeh. "Implementation of autonomous navigation algorithms on two wheeled ground mobile robot". In: *American Journal of Engineering and Applied Sciences* 7 (Apr. 2014), pp. 149–164. DOI: 10.3844/ajeassp.2014.149.164.

[13] I. Anvari. "Non-holonomic Differential Drive Mobile Robot Control Design : Critical Dynamics and Coupling Constraints". Master's thesis. Arizona State University, 2013.

[14] Dollar Bay SOAR. *Parametric ROV thruster propeller*. May 21, 2014. URL: https://www.thingiverse.com/thing:338127/files.

[15] Kjell & Company. *Playknowlogy Uno Rev. 3 Arduino-kompatibelt utvecklingskort*. May 9, 2021. URL: https://www.kjell.com/se/produkter/el-verktyg/utvecklingskit/arduino/utvecklingskort/playknowlogy-uno-rev.-3-arduino-kompatibelt-utvecklingskort-p88860.

[16] Parallax Inc. *PMB-648 GPS Module KickStart (#28500)*. Mar. 17, 2016. URL: https://learn.parallax.com/KickStart.

[17] Diligent. *Pmod CMPS2 Reference Manual*. Apr. 22, 2021. URL: https://reference.digilentinc.com/reference/pmod/pmodcmps2/reference-manual.

[18] Elfa. *UL2.4-12 - Laddningsbart batteri, Blysyra, 12V, 2.4Ah, Bladstift, 4.8/6.3 mm, Ultracell*. Apr. 25, 2021. URL: https://www.elfa.se/sv/laddningsbart-batteri-blysyra-12v-4ah-bladstift-mm-ultracell-ul2-12/p/12454016?q=batteri+12V&pos=6&origPos=77&origPageSize=50&track=true.

[19] SparkFun. *SparkFun Bluetooth Mate Gold*. May 6, 2021. URL: https://www.sparkfun.com/products/12580?_ga=2.92862942.2111475236.1620300804-1710119242.1611178219.

Appendices

Appendix A

Photographs

Figure A.1: RoBuoy, view from above. Taken by authors.

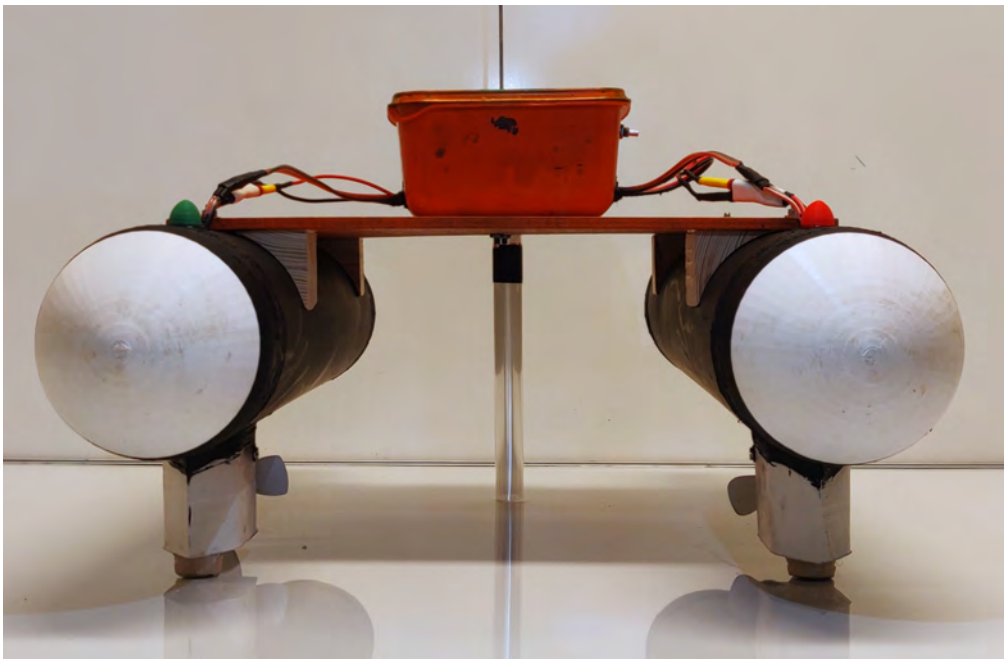


Figure A.2: RoBuoy, front view. Taken by authors.

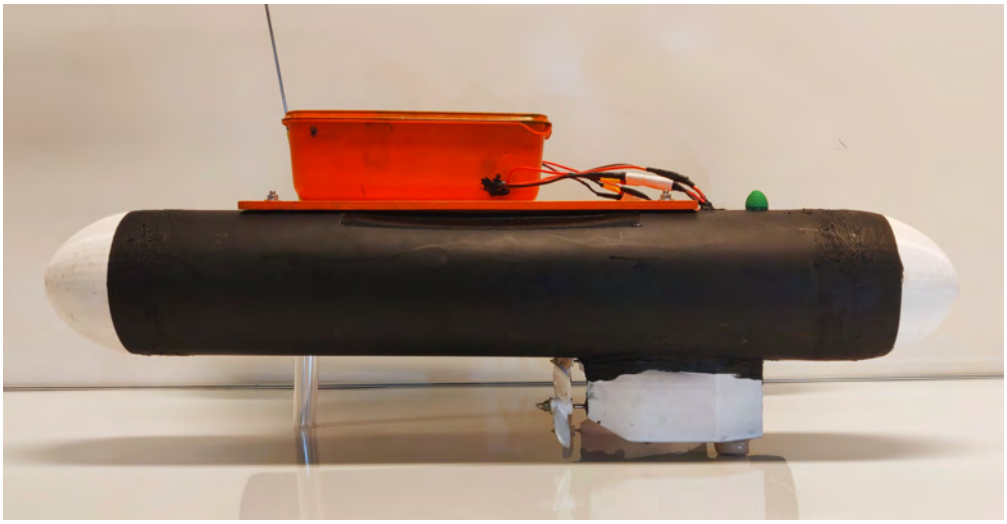


Figure A.3: RoBuoy, side view. Taken by authors.

Appendix B

Arduino Code

B.1 Main Program

```
/*  Main code for RoBuoy, a prototype for an autonomous
     buoy built as a Bachelor's Degree Project at KTH.

    Authors:  Erik Anderberg, Martin Olanders
    Date:     210509
    Course:   MF133X Bachelor's Degree Thesis Project
    in Mechatronics

*/


// Pin variables
const int motorPinRF = 10;  // Right, Forward (Purple)
const int motorPinRR = 9;   // Right, Reverse (Blue)
const int motorPinLF = 5;   // Left, Forward  (Yellow)
const int motorPinLR = 6;   // Left, Reverse  (Green)

// Bluetooth connection
#include <SoftwareSerial.h> // Including library for
// serial connection
int bluetoothTx = 12;  // TX-O pin of bluetooth mate, Arduino D12
int bluetoothRx = 13;  // RX-I pin of bluetooth mate, Arduino D13
SoftwareSerial bluetooth(bluetoothTx, bluetoothRx);

// Navigation variables
float lat_ref = 59.383766; // Target position
float lon_ref = 18.179828;
float lat, lon; // Latitude and longitude
```

```
double distance, bearing, heading, dir;


// PID controller variables and setup
#include <PID_v1.h> // Including library for PID
double v_ref, w_ref; // Linear and angular reference velocities
double U_R, U_L;     //Voltage percentage, right, left
double minU = 3.2 / 12; // Lowest voltage percentage for motors
// to turn
double setc = 180, setd = 0; // Target values for PID controllers
double limheading = 666 - minU * 666; // Limits for PID output
double limdist = 100 - minU * 100;
double Kpc = 7; // Constants for headingPID controller
double Kic = 0.9;
double Kdc = 3.2;
double Kpd = 5; // Constants for distPID controller
double Kid = 0;
double Kdd = 0;
PID headingPID(&dir, &w_ref, &setc, Kpc, Kic, Kdc, DIRECT);
PID distPID(&distance, &v_ref, &setd, Kpd, Kid, Kdd, DIRECT);


void setup() {
  Serial.begin(9600); // Serial Monitor
  Serial.println("Starting");
  CMPS2_init(); // Initialize compass
  delay(10);


  pinMode(motorPinRF, OUTPUT);
  pinMode(motorPinRR, OUTPUT);
  pinMode(motorPinLF, OUTPUT);
  pinMode(motorPinLR, OUTPUT);

  headingPID.SetMode(AUTOMATIC);
  headingPID.SetOutputLimits(-limheading, limheading);
  distPID.SetMode(AUTOMATIC);
  distPID.SetOutputLimits(-limdist, limdist);

  bluetooth.begin(115200);  // The Bluetooth Mate defaults to
  //115200bps
  bluetooth.print("$");  // Print three times individually
  bluetooth.print("$");
  bluetooth.print("$");  // Enter command mode
```

```
  delay(100);  // Short delay, wait for the Mate to send back CMD
  bluetooth.println("U,9600,N");  // Temporarily Change the
  // baudrate to 9600, no parity
  bluetooth.begin(9600);  // Start bluetooth serial at 9600


  delay(10000);//delay to connect bluetooth
  Serial.println("Setup done");
}

void loop() {

// time-dependent target position, menas the buoy returns to
// shore after some time
  if (millis() < 325000) {
    lat_ref = 59.359790;
    lon_ref = 18.053936;
  }
  else if (millis() < 4000000) {
    lat_ref = 59.360008;
    lon_ref = 18.053924;
  }

  getdata_GNSS(); // Calculate lat, lon, bearing and distance

  heading = CMPS2_getheading(); // Get heading from compass

  // Calculate direction to target position
  dir = bearing - heading + 180; // Add 180 to make it pick the
  //shortest turning direction
  //make sure direction is between 0 and 360
  if (dir > 360) {
    dir = dir - 360;
  }
  else if (dir < 0) {
    dir = dir + 360;
  }

  headingPID.Compute();// Compute PID control values
  distPID.Compute();

  getU(); // Calculate motor voltage percentages

  //do not move if extremely far from target, probably GNSS error
```

```
  if (distance < -15000) {
    U_R = 0;
    U_L = 0;
  }

  setMotor('R', U_R); // Set motor voltage
  setMotor('L', U_L);

  // Send data through bluetooth
  bluetooth.print(lat, 7);
  bluetooth.print('\t');
  bluetooth.print(lon, 7);
  bluetooth.print('\t');
  bluetooth.print(distance);
  bluetooth.print('\t');
  bluetooth.print(dir);
  bluetooth.print('\t');
  bluetooth.print(bearing);
  bluetooth.print('\t');
  bluetooth.print(heading);
  bluetooth.print('\t');
  bluetooth.print(U_L);
  bluetooth.print('\t');
  bluetooth.print(U_R);
  bluetooth.print('\t');
  bluetooth.print(millis());
  bluetooth.println(); // Lets Matlab know the line ends,
  //is read as a NaN and can be ignored
}
```

B.2 getdata_GNSS Function

```
/*  Get GNSS coordinates, calculate bearing and distance
 *   to target position

    Cable connections
    VIN (Red)     5V
    GND (Black)   GND
    RXD (Blue)    Pin 3
    TXD (Yellow)  Pin 2
*/

#include <TinyGPS.h> // Including library for GNSS
```

```
SoftwareSerial gnssSerial(2, 3); // Create gnss sensor connection
TinyGPS gnss; // Create gnss object

void getdata_GNSS() {
  gnssSerial.begin(4800); // Connect gnss sensor at 4800 baud rate

  while (gnssSerial.available()) {       // Check for gnss data
    if (gnss.encode(gnssSerial.read())) { // Encode gnss data
      gnss.f_get_position(&lat, &lon);     // Get latitude and
      //longitude

      // Determine course and distance to target
      bearing = gnss.course_to(lat, lon, lat_ref, lon_ref);
      distance = -1 * gnss.distance_between(lat, lon, lat_ref,
      lon_ref);
    }
  }
}
```

B.3 CMPS2_getheading Function

```
/*  Calculate compass heading from magnetic sensors

    Cable connections
    VCC (Red)     3.3V
    GND (Black)   GND
    SDA (Grey)    Pin A4
    SCL (White)   Pin A5
*/

#include <math.h> // Including math library for trigonometry
#include <Wire.h> // Including library for I2C communication

#define DECLINATION 6.85 // declination (in degrees) in
//Stockholm, Sweden 2021.

unsigned char CMPS2_address = 0x30; //I2C address of the device
float  X, Y;

float CMPS2_getheading(void) {
  float components[2], deg = 0;
  CMPS2_set(false);   // Set the polarity
```

```
CMPS2_read_XYZ();   // Read X, Y, Z components of the
// magnetic field
components[0] = X;  // Save current results
components[1] = Y;

CMPS2_set(true);    // Reset the polarity
CMPS2_read_XYZ();   // Read X, Y, Z components of the
// magnetic field

// Eliminate offset from all components
components[0] = (components[0] - X) / 2.0;
components[1] = (components[1] - Y) / 2.0;

// Calculate headinging from components of the magnetic field
// The formula is different in each quadrant
if (components[0] > 0) // Right half plane
{
  if (components[1] > 0)
  {
    deg = atan(components[1] / components[0]) *
    (180 / 3.14159);        // Quadrant 1
  }
  else
  {
    deg = 360 - atan(-components[1] / components[0]) *
    (180 / 3.14159); // Quadrant 2
  }
}
else { // Left half plane
  if (components[1] < 0)
  {
    deg = 270 - atan(components[0] / components[1]) *
    (180 / 3.14159);  // Quadrant 3
  }
  else
  {
    deg = 180 + atan(components[1] / components[0]) *
    (180 / 3.14159);  // Quadrant 4
  }
}
// Correct heading for declination and the compass being
// mounted backwards
deg = deg - 180 + DECLINATION;
```

```
  if (deg > 360) {
    deg -= 360;
  }
  else if (deg < 0) {
    deg += 360;
  }

  return deg;
}

// Reads measurements in mG
void CMPS2_read_XYZ(void) {
  // Command internal control register 0 bit 0 (measure)
  Wire.beginTransmission(CMPS2_address);
  Wire.write(0x07);
  Wire.write(0x01);
  Wire.endTransmission();
  delay(8);

  // Wait for measurement to be completed
  bool flag = false;
  while (!flag) {
    //jump to status register
    Wire.beginTransmission(CMPS2_address);
    Wire.write(0x06);
    Wire.endTransmission();

    // Read its value
    Wire.requestFrom(CMPS2_address, (uint8_t)1);
    int temporal = 0;
    if (Wire.available()) {
      temporal = Wire.read();
    }

    // If the last bit is 1, data is ready
    temporal &= 1;
    if (temporal != 0) {
      flag = true;
    }
  }

  // Move address pointer to first address
  Wire.beginTransmission(CMPS2_address);
  Wire.write(0x00);
```

```
  Wire.endTransmission();

  // Save data
  Wire.requestFrom(CMPS2_address, (uint8_t)6);
  byte tmp[6] = {0, 0, 0, 0, 0, 0}; // Array for raw data
  if (Wire.available()) {
    for (int i = 0; i < 6; i++) {
      tmp[i] = Wire.read(); // Save it
    }
  }

  // Initialize array for data
  float measured_data[2];

  // Reconstruct raw data from the two bytes in each axis
  measured_data[0] = 1.0 * (unsigned int)(tmp[1] << 8 | tmp[0] );
  // x
  measured_data[1] = 1.0 * (unsigned int)(tmp[3] << 8 | tmp[2] );
  // y

  // Convert raw data to mG
  for (int i = 0; i < 2; i++) {
    measured_data[i] = 0.48828125 * (float)measured_data[i];
  }

  X = measured_data[0];
  Y = measured_data[1];

  return;
}

// Initialize the compass
void CMPS2_init(void) {
  Wire.begin(); // Initialization of I2C bus

  // Command internal control register 0 for set operation
  Wire.beginTransmission(CMPS2_address);
  Wire.write(0x07);
  Wire.write(0x20);
  Wire.endTransmission();
  delay(10);

  // Command internal control register 1 to 16 bit resolution,
  // 8ms measurement time
```

```
  Wire.beginTransmission(CMPS2_address);
  Wire.write(0x08);
  Wire.write(0x00);
  Wire.endTransmission();
  delay(10);
}

// Sets/resets the sensor, changing the magnetic polarity of
// the sensing element
void CMPS2_set(bool reset) {
  // Command internal control register 0 bit 7 (capacitor
  // recharge)
  Wire.beginTransmission(CMPS2_address);
  Wire.write(0x07);
  Wire.write(0x80);
  Wire.endTransmission();
  delay(50);

  if (reset) {
    // Command internal control register 0 bit 6 (reset)
    Wire.beginTransmission(CMPS2_address);
    Wire.write(0x07);
    Wire.write(0x40);
    Wire.endTransmission();
    delay(10);
  }
  else {
    // Command internal control register 0 bit 5 (set)
    Wire.beginTransmission(CMPS2_address);
    Wire.write(0x07);
    Wire.write(0x20);
    Wire.endTransmission();
    delay(10);
  }
  return;
}
```

B.4 getU Function

```
/*  Calculate voltage percentage for both motors based on
 *  values from PID controllers
 */
```

```
float maxU; // maximum voltage percentage

void getU() {
  // Maintains turning if far away
  maxU = abs(v_ref) + 0.15 * abs(w_ref) + minU*100;
  if (maxU > 100){
    if (v_ref > 0){
      v_ref = 100 - 0.15*abs(w_ref) - minU*100;
      }
    else {
      v_ref = -100 + 0.15*abs(w_ref) + minU*100;
      }
    }

  // Calculates right and left motor voltage percentage
  U_R = v_ref + 0.15 * w_ref;
  U_L = v_ref - 0.15 * w_ref;

  // adds minimum necessary voltage for motors to turn
  if (U_R < 0) {
    U_R = U_R - minU*100;
  }
  else {
    U_R = U_R + minU*100;
  }

  if (U_L < 0) {
    U_L = U_L - minU*100;
  }
  else {
    U_L = U_L + minU*100;
  }
}
```

B.5 setMotor Function

```
/*  Set motor speed and direction
 *
 *  Input:
 *  motor: Motor choice ('R' or 'L')
 *  U_x: Voltage percentage (-100 to 100)
 *
 *  Cable connections:
```

```
 *  MotorPinRF  IN2
 *  MotorPinRR  IN1
 *  MotorPinLF  IN1
 *  MotorPinLR  IN2
 *  Red         Out2
 *  Black       Out1
 */

void setMotor(char motor, int U_x){
  if (motor == 'R') { // Right motor
    if (U_x < 0){            // Reverse
      analogWrite(motorPinRF, 0);
      analogWrite(motorPinRR, -U_x*2.55);
    }
    else {                     // Forward
      analogWrite(motorPinRF, U_x*2.55);
      analogWrite(motorPinRR, 0);
    }
  }
  else { // Left mnotor
    if (U_x < 0){             // Reverse
      analogWrite(motorPinLF, 0);
      analogWrite(motorPinLR, -U_x*2.55);
    }
    else {                     // Forward
      analogWrite(motorPinLF, U_x*2.55);
      analogWrite(motorPinLR, 0);
    }
  }
}
```

Appendix C

MATLAB Code

C.1 Code for Bluetooth Communication and Data Storage

```
%% Bluetooth communication between Arduino microcontroller and computer
%
% This program is used to communicate with the microcontroller through
% Bluetooth, in order to receive, save and analyze data.
%
% Authors:  Erik Anderberg and Martin Olanders
% Date:     210509
% Course:   MF133X Bachelor's Degree Thesis Project in Mechatronics
%           at KTH, Royal Institute of Technology
%% Search for Bluetooth devices
list = bluetoothlist("Timeout",15)

%% Pair with Bluetooth module
clear bt;
bt = bluetooth("Sparky BT",1)

%% Recieive data
clc
n_av = bt.NumBytesAvailable
bt.flush % Flush out built up data

N = 2000; % Number desired of data readings

position = zeros(N,2);
distance = zeros(N,1);
direction = zeros(N,1);
bearing = zeros(N,1);
heading = zeros(N,1);
```

```
motorPow = zeros(N,2);
time = zeros(N,1);

ii = 1;
while ii <= N

    n_av = bt.NumBytesAvailable; % Check if anything was sent

    if n_av > 64 % If an entire data set is available
        n_av;
        data = readline(bt) % Read until line ends
        data = str2double(split(data))'; % Split and convert to double

        position(ii,:) = data(1:2);
        distance(ii) = data(3);
        direction(ii) = data(4);
        bearing(ii) = data(5);
        heading(ii) = data(6);
        motorPow(ii,:) = data(7:8);
        time(ii) = data(9);

        ii = ii + 1
    end
end

%% Save data
filename = ['robuoydata'  num2str(jj)  '.mat']
save(filename,'position','distance','direction','bearing',...
    'heading','motorPow','time');
jj = jj + 1;
```

TRITA TRITA-ITM-EX 2021:20