



**ENSTA**  
**BRETAGNE**



---

# Development of a software architecture for an autonomous sailboat

---

Assistant Engineer Internship Report  
FISE ROB 2025  
April 2024 - August 2024

**Titouan LEOST**  
*ENSTA Bretagne - Autonomous Robotics*  
titouan.leost@ensta-bretagne.org

Supervisor:

**Dr. Jian WAN**  
*Aston University - College of Engineering and Physical Sciences Mechanical,  
Biomedical & Design Engineering*  
j.wan3@aston.ac.uk

## Acknowledgement

*I would like to express my deepest thanks Dr. Jian Wan, Lecturer at Aston University, for the opportunity he offered me with this internship, as well as for his assistance throughout the project. I would also like to thank my professor, Luc Jaulin, for enabling me to benefit from this opportunity.*

## Résumé

L'objet de ce rapport est de présenter le travail effectué lors d'un stage de quatre mois à l'université d'Aston à Birmingham. L'objectif du projet était de développer une architecture logicielle facilement réutilisable afin de permettre à un voilier de réaliser des missions en autonomie. De plus, l'idée était d'utiliser le matériel le plus simple possible afin de fournir une solution peu couteuse et low-tech.

Le projet a donc consisté dans un premier temps à mettre en place une architecture permettant de récolter et traiter les données des différents capteurs afin de nourrir un algorithme contrôlant le voilier à l'aide de servomoteurs.

Après s'être assuré du bon fonctionnements des capteurs, la robustesse de l'architecture a été vérifiée en réalisant des tests sur un lac, en s'appuyant sur un algorithme simple dont l'efficacité avait déjà été prouvée.

## Abstract

The purpose of this report is to present the work carried out during a four-month internship at Aston University in Birmingham. The aim of the project was to develop an easily reusable software architecture to enable a sailing boat to carry out autonomous missions. In addition, the idea was to use the simplest possible hardware to provide a low-cost, low-tech solution.

The project therefore initially involved setting up an architecture to collect and process the data from the various sensors in order to feed an algorithm controlling the sailboat via servomotors.

After ensuring the sensors were functioning correctly, the robustness of the architecture was verified by conducting tests on a lake, using a simple algorithm whose effectiveness had already been proven.

## Keywords

Sailboat, Software Architecture, Autonomous Control Algorithm, Arduino, Sensors, Low-tech

# Contents

<b>Acknowledgement</b> .....	<b>1</b>
<b>Résumé</b> .....	<b>2</b>
<b>Abstract</b> .....	<b>2</b>
<b>Keywords</b> .....	<b>2</b>
<b>1 Introduction</b> .....	<b>4</b>
1.1 Context and Problematic .....	4
1.2 System description .....	4
<b>2 Hardware</b> .....	<b>5</b>
2.1 Circuit board .....	6
2.2 Sensors .....	6
2.2.1 IMU .....	6
2.2.2 GNSS .....	7
2.2.3 Wind sensor .....	7
2.3 Actuators .....	8
2.4 RC Receiver .....	8
2.5 SD Card .....	8
2.6 Power Supply .....	9
<b>3 Software</b> .....	<b>9</b>
3.1 Architecture .....	9
3.1.1 Environment .....	9
3.1.2 Observer .....	11
3.1.3 Command .....	12
3.1.4 Supervisor .....	13
3.1.5 Recorder .....	13
3.2 Algorithms .....	13
3.2.1 Line Following .....	13
3.3 Config .....	14
3.4 Main Program .....	14
<b>4 Results</b> .....	<b>15</b>
<b>5 Conclusion</b> .....	<b>21</b>
<b>References</b> .....	<b>23</b>
<b>Appendices</b> .....	<b>24</b>
<b>A Wiring diagrams</b> .....	<b>24</b>
<b>B Hardware list</b> .....	<b>26</b>
<b>C Class diagram</b> .....	<b>28</b>
<b>D Final results</b> .....	<b>29</b>
<b>Acronyms</b> .....	<b>43</b>

# 1 Introduction

## 1.1 Context and Problematic

With over two-thirds of our planet's surface covered by oceans, their exploration has become an increasingly critical issue. It is within this context that unmanned surface vehicles (USVs) come into play. While most USVs are powered by motors, a new type emerged in the 2000s: autonomous sailboats. Powered by wind, these sailboats offer even greater autonomy than conventional USVs while reducing environmental impact, making them ideal candidates for long-term missions like coastal surveillance. [1–3]

Given that the development of autonomous sailboats is relatively new and technically complex, it is crucial to train future engineers in the control methods of such vessels. The goal of this project is to provide a hardware and software platform that can be easily adopted by universities and schools to educate students on the development of autonomous sailboats. Therefore, the platform must utilize standard, and preferably low-cost and low-tech components, making it accessible and aligned with environmental sustainability.

This project is based at Aston University in Birmingham, England, specifically within the Smart and Sustainable Manufacturing Research Centre, which Dr. Jian Wan is a part of. The research center's mission is to offer innovative solutions that help industry transition towards digital and environmental sustainability.

## 1.2 System description



Figure 1: The sailboat used during the project

Sailing typically requires controlling both the rudder and the sail. The wind must also be considered in order to adjust the boat's course and the sail's trim. Figure 2 outlines the notations and reference

frames used in this report, with variables described alongside figure 3 .

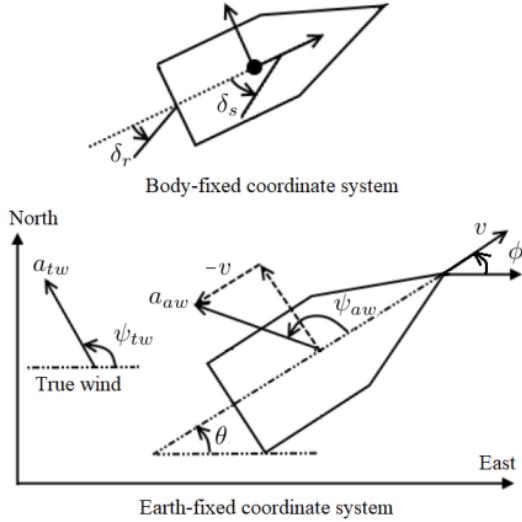


Figure 2: The variables and coordinate systems (figure taken from [4])

$\theta$	heading of the boat
$v$	velocity of the sailboat
$\phi$	course of the boat
$a_{aw}, \psi_{aw}$	speed and direction of the apparent wind
$a_{tw}, \psi_{tw}$	speed and direction of the true wind
$\delta_r$	angle of the rudder
$\delta_s$	angle of the sail

Figure 3: Notations

As such, several variables ( $\delta_r, \delta_s$ ) are defined in the boat's reference frame, while others ( $\theta, v, \phi, a_{aw}, \psi_{aw}, a_{tw}, \psi_{tw}$ ) are defined in an East-North-Up (ENU) coordinate system, with its origin fixed to the Earth.

## 2 Hardware

As mentioned earlier, one of the key points of this project was to use inexpensive and standard equipment. This helps to keep the overall project cost relatively low while allowing anyone to easily reproduce the project. Additionally, using simple and low-power components results in low energy consumption, giving the sailboat greater autonomy. Finally, the environmental aspect of the project is not to be overlooked.

At the beginning of the project, a choice was made between developing the architecture on a Raspberry Pi or an Arduino. The latter option was selected as it was considered the most suitable and best aligned with the project's goals.. However, based on the work of previous students [4], the setup with Arduino turned out to be much less compact than the one with Raspberry Pi (see comparison in Appendices with figures 27 and 28).

## 2.1 Circuit board

An Arduino Mega 2560 was chosen for the electronic board, which has enough UARTs, I2C and SPI ports to support all the sensors and servomotors. Moreover, the board is low-cost and consumes very little power, which means it meets the constraints of the project. Its low power consumption goes hand in hand with low power, which can be a constraint but is not an obstacle in the context of this project, as the software architecture is intended to be as simple as possible.

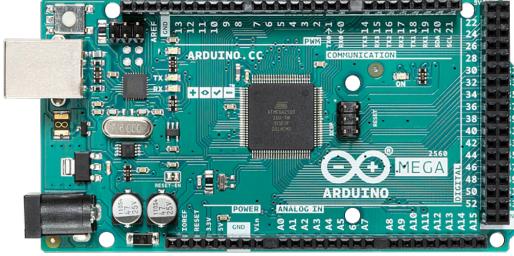
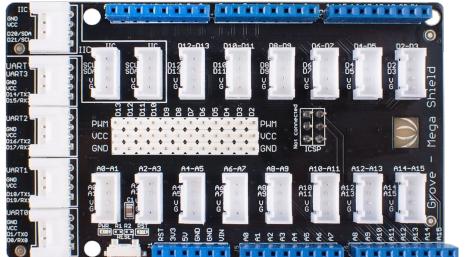


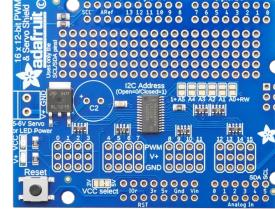
Figure 4: Arduino Mega 2560

For cleaner, more stable wiring, the Arduino is topped by a Grove - Mega Shield, which standardises all connectors to 4 pins (Signal 1, Signal 2, VCC, GND).

Finally, a second shield (Adafruit 16 x 12-bit PWM & Servo Shield) has been added to the assembly. This provides the PWM connectors needed to control the servomotors, as well as a separate power input dedicated to the servomotors. The PWM outputs of the interface are controlled via I2C, with the Arduino board acting as the master and the PWM outputs as the slaves.



(a) Grove Mega Shield



(b) Adafruit 16 x 12-bit  
PWM & Servo Shield

Figure 5: The two shields used to complement the Arduino board

## 2.2 Sensors

### 2.2.1 IMU

The IMU used is the CMPS12 powered by the Bosch BNO055. This IMU supports 9 degrees of freedom (3-axis magnetometer, 3-axis gyro and 3-axis accelerometer) and has the advantage of being self-calibrating, with the user simply having to make a few movements when starting up the CMPS12. In this project, only the heading value measured by the magnetometer is used. The CMPS12 uses UART protocol to communicate with the Arduino board through a serial port.



Figure 6: CMPS12

As the boat can pitch under the effect of waves and wind, thus disturbing the heading value, a filter has been applied to the latter to smooth it out. This filtered is detailed in the section 3.1.2.

### 2.2.2 GNSS

The GNSS used is the Grove - GPS Module. This is used to obtain the boat's position during the experiments. Communication between the GNSS and the Arduino board is carried out via a serial port using the UART protocol.



Figure 7: Grove - GPS Module

### 2.2.3 Wind sensor

To obtain the wind speed and direction, an anemometer is installed at the front of the boat. Although this system is somewhat imprecise, it still allows for the calculation of the true wind direction, which subsequently helps in adjusting the sail's position and adapting the course followed by the sailboat. The wind sensor is plugged to an analog pin to communicate the wind direction, and to an interrupt pin to give the wind speed.



Figure 8: Davis Anemometer - SKU 7911

As the action of gusts of wind can interfere with the measurement of wind direction, the latter is filtered. The filter applied is detailed in section 3.1.2 .

## 2.3 Actuators

To control the boat, two servomotors are required. The first one must allow direct control of the rudder over an angular range from  $-45^\circ$  to  $+45^\circ$ . The second is used to control the maximum sail opening angle, which needs to reach up to  $90^\circ$ . To meet these requirements, the Hitec HS-5645MG and HS-785HB were selected for the rudder and sail, respectively. The servomotors are controlled with PWM signals.



(a) Hitec HS-5645MG



(b) Hitec HS-785HB

Figure 9: Actuators used

## 2.4 RC Receiver

For safety and convenience, the sailboat can be remotely controlled. To achieve this, an RC receiver linked to a remote control is installed on the Arduino. Additionally, this setup allows the algorithm to be started once everything is ready by sending a signal with the remote control. Without this system, the algorithm and data recording would be triggered as soon as the battery is connected to the Arduino board, which is not desirable.

To control the two servomotors, two channels of the RC receiver are used. These are connected to interrupt pins to allow the Arduino to detect when the controller is on (although only one interrupt is actually needed).



Figure 10: FlySky FS-R6B RC Receiver

## 2.5 SD Card

To keep the system as simple as possible, no real-time data monitoring system was installed on the boat. Therefore, it is even more important to install an SD card on the Arduino to save mission data

for later analysis. Since the Arduino Mega 2560 does not have a built-in SD card slot, it was necessary to add an SD card adapter. The SD card adapter uses SPI protocol to gather the data from the Arduino board.

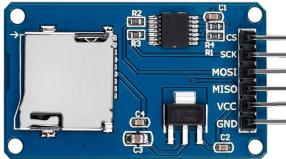


Figure 11: MicroSD Card Adapter

## 2.6 Power Supply

A battery with two output is used to power the system. The first output is connected to the Arduino Mega 2560 via its USB-B connector and provides 5v with 1A. The second output powers the Adafruit servo shield with 5v and 2.5A.

# 3 Software

## 3.1 Architecture

To facilitate code understanding and debugging, the software architecture has been divided into five main blocks, as illustrated in figure 12.

### 3.1.1 Environment

The environment block constitutes the lowest level of the architecture. It includes the drivers for the sensors and actuators so that they can be reused elsewhere in the code. This block enables the sailboat to interact with its environment.

Each sensor and actuator has its own class:

- The **CMPS12** class contains methods to communicate with the IMU to obtain the yaw, pitch and roll of the boat. The class also stores the filtered yaw value and handles the calibration of the CMPS12.
- The **GPS** class uses the *TinyGPS++* library to handle the GPS. It allows to obtain the latitude, longitude, speed and course of the boat as well as the number of satellites seen by the GPS and the date and time of the experiment. Moreover, since it is often easier to work with Cartesian coordinates rather than latitude and longitude, the class includes a method to get the coordinates converted into an ENU frame with

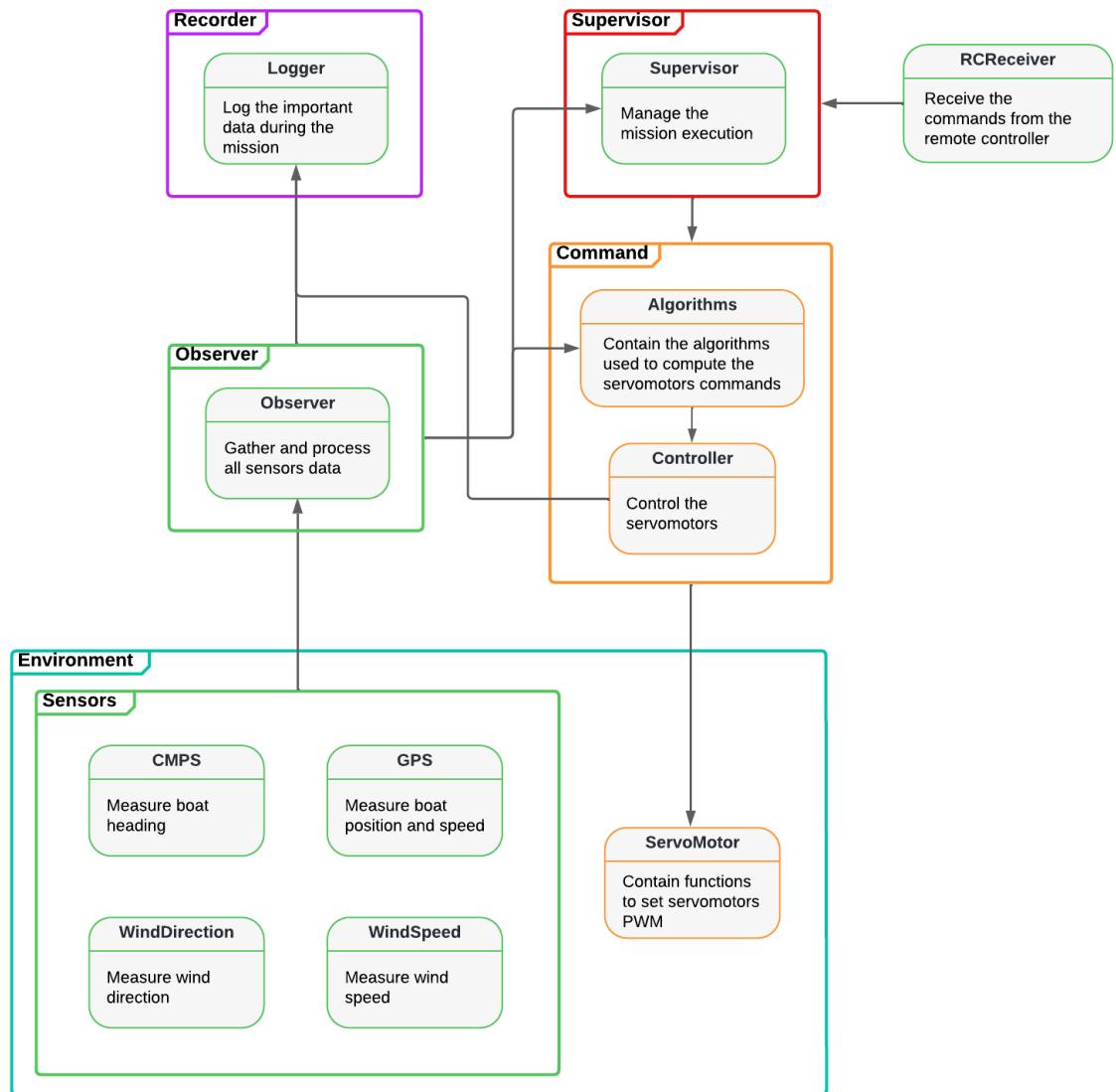


Figure 12: Simplified class diagram

$$x = R_{\text{earth}} \cdot \cos(lat) \cdot \frac{pi}{180} \cdot (lon - lon_{\text{ref}}) \cdot \frac{pi}{180}$$

$$y = R_{\text{earth}} \cdot (lat - lat_{\text{ref}}) \cdot \frac{pi}{180}$$

where  $R_{\text{earth}}$  is the Earth radius,  $lat$  and  $lon$  are the latitude and longitude measured by the GNSS, and  $lat_{\text{ref}}$  and  $lon_{\text{ref}}$  are the latitude and longitude of reference.

- The **WindDirection** class reads the wind direction by mapping the value provided by the sensor's potentiometer to a range between 0 and 360 degrees. This class can also store the filtered wind direction value.
- The **WindSpeed** class allows for obtaining the wind speed. To achieve this, the anemometer is connected to an interrupt pin. This pin is triggered with each rotation of the sensor, calling a function that increments a counter. The counter is then used to determine the number of rotations over a three-second period, which allows for calculating the wind speed using the formula provided in the documentation [5]

$$V = P \cdot \frac{2.25}{T} \cdot 1.609,$$

where  $V$  is the wind speed in km/h,  $P$  is the number of pulses per sample period and  $T$  is the sample period in seconds.

- The **ServoMotor** class uses the *Adafruit\_PWM\_Servo\_Driver* library to set the PWM value of the servomotors. This PWM is computed with the following formula

$$PWM = (PWM_{\max} - PWM_{\min}) \cdot x + PWM_{\min},$$

where  $x$  is a float between 0.0 and 1.0 representing the percentage of the command, and  $PWM_{\min}$  and  $PWM_{\max}$  are the minimum and maximum values that can be sent to the servomotor.

The sensors' classes contain an *update* method that is used by the observer to update the state of the boat.

### 3.1.2 Observer

The observer collects and processes the information gathered by the sensors. To do this, the **Observer** class contains an *updateSensors* method, which calls the *update* method of each sensor. Then, the heading and wind direction values are filtered. The filter used is based on a very simple low-pass filter

$$y_n = y_{n-1} + a \cdot (\theta_n - y_{n-1}),$$

where  $y_n$  is the new filtered value,  $y_{n-1}$  the previous filtered value,  $\theta_n$  the actual raw value,  $a$  the filter coefficient. This filter is then adjusted so that  $y$  and  $\theta$  vary between 0 and 360°. The resulting filter becomes

$$y_n = \text{wrap}_{360}(y_{n-1} + a \cdot \text{wrap}_{360}(\theta_n - y_{n-1})),$$

where

$$\text{wrap}_{360}(\theta) = \theta - 360 \cdot \left\lfloor \left( \frac{\theta + 180}{360} \right) \right\rfloor$$

Finally, the *updateSensors* method computes the true wind direction ( $TW_D$ ), i.e., the wind direction in the ENU frame. For that, the observer needs the following information [6]:

- the apparent wind speed and direction in the boat reference frame ( $AW_S$  and  $AW_D$  respectively), measured by the anemometer.
- the speed and course over the ground of the boat ( $SOG$  and  $COG$ ) measured by the GNSS.
- the heading ( $H$ ) of the boat measured by the IMU.

$$TW = \begin{pmatrix} SOG \cdot \sin(COG) - AW_S \cdot \sin(AW_D + H) \\ SOG \cdot \sin(COG) - AW_S \cdot \cos(AW_D + H) \end{pmatrix}$$

$$TW_D = \text{angle}(TW)$$

Due to some issues with the GNSS that are explained in section 4, the course over the ground was replaced with the heading of the boat. This is not ideal because those two values are not necessarily the same. However, as the sailboat was not drifting too much during the experiments, it was considered a better solution than using the GNSS data.

### 3.1.3 Command

The command block calculates and sends commands to the servomotors based on the algorithms used. Two types of algorithms can be implemented:

- Type 1 Algorithms: Allow the boat to reach a waypoint (e.g., Line Following).
- Type 2 Algorithms: Enable the sailboat to perform a task for a given duration once the waypoint is reached (e.g., Station Keeping).

During a mission, the sailboat can either rely solely on a type 1 algorithm or use both types of algorithms.

The command block is then divided into two stages: the algorithmic part (developed in section 3.2), which calculates the command, and the control part, which updates the servomotors. The latter is implemented with the **Controller** class, which initializes, tests, and updates the servomotors.

### 3.1.4 Supervisor

The supervisor oversees the mission's progress. To do this, the **Supervisor** class relies on a predefined list of waypoints. When the current waypoint is reached, the supervisor moves on to the next one and sends this information to the command block. The supervisor also selects the appropriate algorithm for the command block to use based on the situation. Finally, the supervisor allows switching from autonomous mode to manual control mode using the remote control.

### 3.1.5 Recorder

The recorder collects a set of data sent by other blocks and saves it onto the SD card for later analysis. These data can include the boat's position, its heading (both filtered and raw), the control mode (manual or autonomous), wind speed, and more. This block is therefore of critical importance, as it is used to ensure that the code is working properly and helps with debugging if needed.

In parallel with the software architecture developed for the sailboat, a Python script is provided to display graphs and a video of the collected data to simplify analysis. Additionally, two other files are included to retrieve the data from the SD card by reading it directly from the Arduino. This allows you to save the data to your computer without needing an adapter to connect the SD card to the computer. However, reading data from the Arduino is very slow and is not suitable for long missions.

## 3.2 Algorithms

The algorithmic part of the project is the most important for its modular aspect. Indeed, it is essential that algorithms can be easily added to the code. To achieve this, a parent class, **AlgorithmInterface**, introduces the class methods and variables necessary for the proper functioning of the various algorithms. Each new algorithm will therefore inherit from this class and will only require the implementation of one or two methods, depending on the type of algorithm.

### 3.2.1 Line Following

To verify the functionality of the architecture, a relatively simple line-following algorithm was implemented. This algorithm, described in figure 13, is based on the work of Mr Jaulin and Mr Le Bars [7].

The algorithm takes as input the position of the boat **m**, its heading  $\theta$ , the true wind direction  $\psi$ , and the two points **a** and **b** that define the line to follow. Moreover, a state variable  $q \in \{-1, 1\}$  is used to define the favored tack. The algorithm computes the error  $e$  to the line between **a** and **b**, and the angle  $\varphi$  of the line. It then computes the nominal angle  $\theta^*$  that the sailboat should follow and adapts this angle depending on the true wind direction  $\psi$  and the closed haul angle  $\zeta$ . Finally, it computes the angle of the rudder  $\delta_r$  and the max opening angle of the sail  $\delta_s^{max}$ .

<b>Function</b> in: $\mathbf{m}, \theta, \psi, \mathbf{a}, \mathbf{b}$ ; out: $\delta_r, \delta_s^{max}$ ; inout: $q$
<pre> 1   <math>e = \det\left(\frac{\mathbf{b}-\mathbf{a}}{\ \mathbf{b}-\mathbf{a}\ }, \mathbf{m} - \mathbf{a}\right)</math> 2   if <math> e  &gt; \frac{r}{2}</math> then <math>q = \text{sign}(e)</math> 3   <math>\varphi = \text{atan2}(\mathbf{b} - \mathbf{a})</math> 4   <math>\theta^* = \varphi - \frac{2\cdot\gamma_\infty}{\pi} \cdot \text{atan}\left(\frac{e}{r}\right)</math> 5   if <math>\cos(\psi - \theta^*) + \cos\zeta &lt; 0</math> 6       or (<math> e  &lt; r</math> and <math>(\cos\psi - \varphi + \cos\zeta &lt; 0)</math>) 7           then <math>\bar{\theta} = \pi + \psi - q \cdot \zeta</math> 8           else <math>\bar{\theta} = \theta^*</math> 9   end 10  <math>\delta_r = \frac{\delta_r^{max}}{\pi} \cdot \text{sawtooth}(\theta - \bar{\theta})</math> 11  <math>\delta_s^{max} = \frac{\pi}{2} \cdot \left(\frac{\cos(\psi - \bar{\theta}) + 1}{2}\right)^{\frac{\log\left(\frac{\pi}{2\beta}\right)}{\log(2)}}</math> </pre>

Figure 13: Description of the line-following algorithm used with  $\text{sawtooth}(x) = 2 \cdot \text{atan}(\tan\left(\frac{x}{2}\right))$

### 3.3 Config

The Config file is also important for the modularity of the project. Indeed, all variables that can or must be modified are declared in this file. The idea is to centralize all potential changes within a single file to simplify the use of the project. This file defines, among other things, the algorithms to be used, the list of waypoints to reach, the reference position for the ENU frame, the filter coefficients, and the extreme PWM values for the servomotors.

### 3.4 Main Program

The main program is the Arduino code uploaded to the Arduino board. This program relies on all the previously defined files to control the boat. An initialization phase is carried out in the setup of the code. The object associated with the RC receiver is initialized, and the code waits for a signal from the remote control to continue execution. Then, the objects implementing the observer, controller, supervisor, and logger are initialized, and the program moves on to the loop. The control loop is very simple and operates in four steps:

- The observer updates all the sensors.
- The supervisor uses this new data to update the mission.
- The controller uses both the data provided by the observer and the supervisor to update the servomotor control.
- The logger records the latest data.

This process is then repeated until the supervisor declares the mission complete, meaning when the final waypoint is reached.

## 4 Results

During the experiments conducted on an artificial lake (figure 14), numerous issues arose.



Figure 14: Picture of the sailboat during an experiment

Firstly, wind speed frequently caused issues. Although the sailboat used does not require a large amount of wind to move, the anemometer measuring wind direction lacked sensitivity. It was therefore not uncommon for the anemometer to remain fixed relative to the boat's frame despite significant changes in the boat's heading, as shown in figure 16, indicating that the sensor was not rotating with the wind. This led to inaccurate calculations of the true wind direction, preventing the boat from properly adjusting its course and sail, often resulting in it stalling. Based on experimental results, the sensor appears to function properly for wind speeds over 4 km/h, even though the manufacturer's documentation indicates a range starting at 1 mph (1.6 km/h) [5].

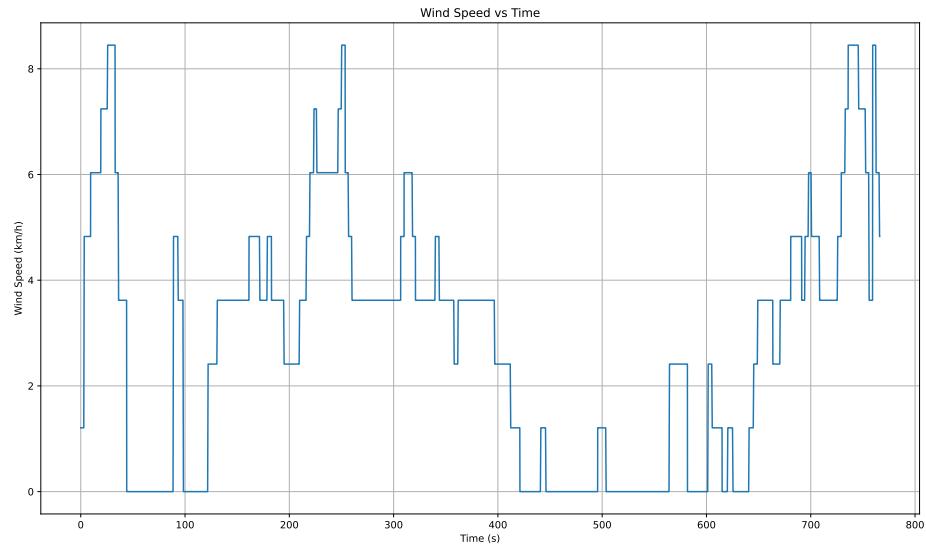


Figure 15: Wind speed readings from the test conducted on 30/07/24

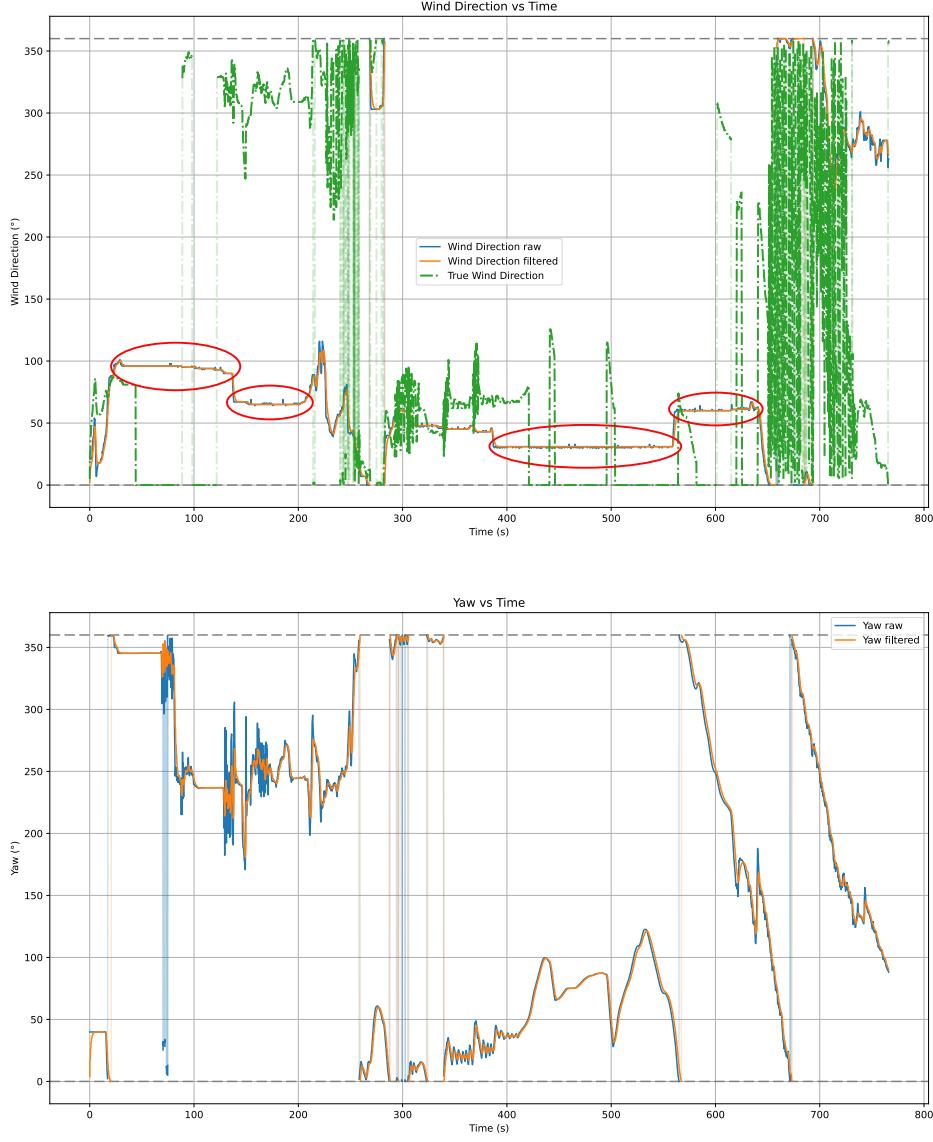


Figure 16: Comparative analysis of wind direction and yaw readings from the test conducted on 30/07/24. The areas outlined in red highlight the lack of rotation of the anemometer despite changes in the boat's heading.

One solution to address this issue would be to install a more sensitive sensor. However, this would likely require a more expensive and complex sensor, which would conflict with the project's low-tech philosophy. Thus, this issue with low wind speeds highlights one of the project's limitations.

In rarer cases, the opposite problem occurred with strong winds. In this situation, the lightweight sailboat could not resist and was sometimes blown off course. This scenario is illustrated in figure 17, where the boat veered around the waypoint due to strong westerly winds. Unfortunately, no data

on the wind speed could be collected during this trial, as one of the anemometer's cables became disconnected while installing the components in the boat.

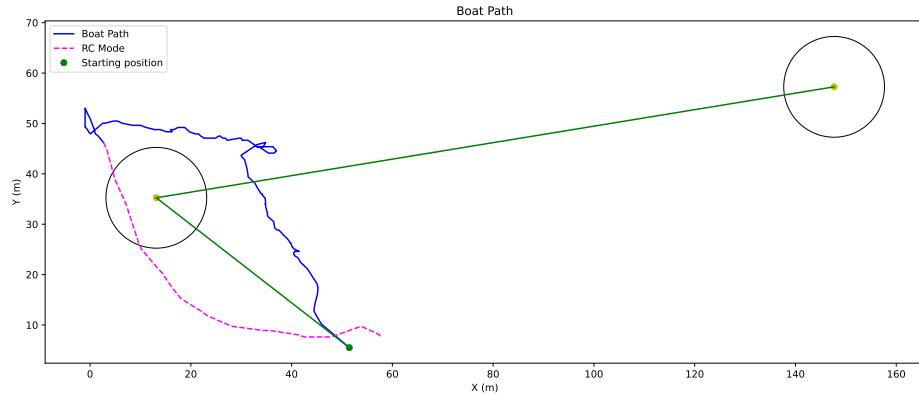


Figure 17: Boat trajectory from the test conducted on 09/08/24. The dark circles represent the radius to validate a waypoint.

This therefore highlights a second source of problems. The onboard electronics, being bulky and having many cables, are challenging to install in the boat's hull without disconnecting something. Furthermore, the hatch through which the components are inserted is quite narrow, and the internal space of the hull is limited, complicating the installation process (figure 18). A potential solution would be to use shorter cables. Replacing the Arduino Mega 2560 with an Arduino Uno would also significantly improve compactness. However, the Arduino Uno has less memory and fewer ports than the Mega 2560, so porting the project to the former may not be feasible. Lastly, it is possible to modify the boat's hull to install a larger hatch, although this would require alterations to the boat, limiting the project's ease of reproduction.

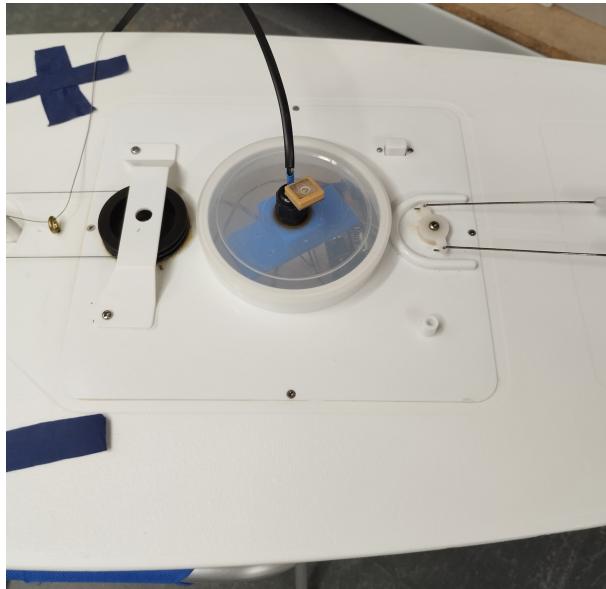


Figure 18: Boat's hatch

The GPS module represents the third source of error. On occasion, it failed to update the boat's position despite significant movement, resulting in incorrect rudder command calculations. This problem is illustrated in figure 19, where the GPS course of the boat remains constant for extended periods, despite changes in heading. Figure 20 shows the distance covered by the boat without updating its position, which could exceed ten metres. Although this significantly impacted the sailboat's performance, this issue occurred infrequently during trials, and the cause remains unclear.

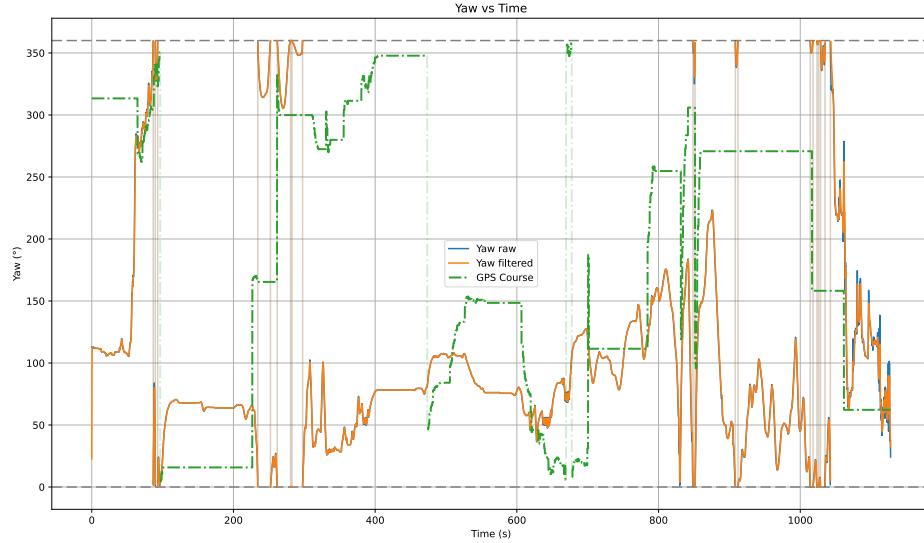


Figure 19: Heading and GPS course from the test conducted on 06/08/24

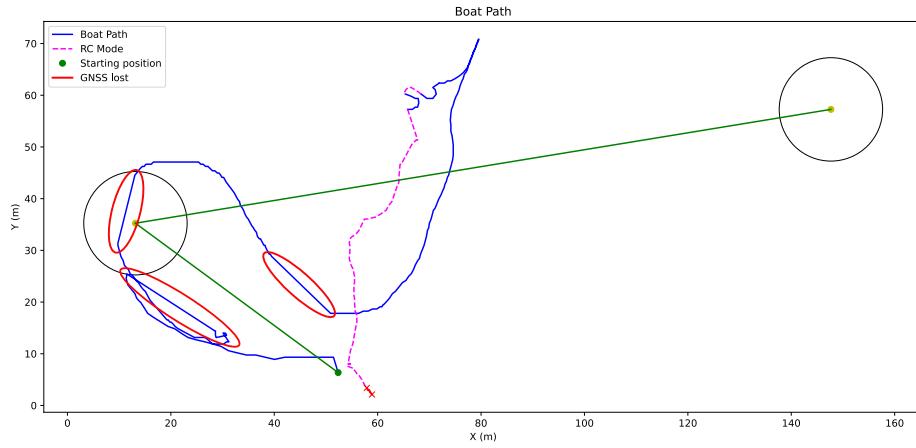


Figure 20: Boat trajectory from the test conducted on 06/08/24. The dark circles represent the radius to validate a waypoint, and the green line is the line to follow. The areas outlined in red highlight the lack of update of the GPS.

Lastly, the main problem encountered stemmed from the IMU. It regularly returned incorrect

values, as seen in figure 21, where the boat was initially set up pointing approximately north. Due to the narrowness of the hull, the IMU is installed between the two servomotors and is very close to them. The servomotors' magnetic fields then interfere with the IMU's measurements. Since the IMU cannot be relocated, it is impossible to avoid disturbances from the servomotor power supply, which is not constant. However, these are minor compared to the magnetic fields generated by the servomotors' ferromagnetic components. As these are constant, the solution was to calibrate the IMU once it was installed in the hull so that the calibration accounted for this disturbance. The result is shown in the figure 22. We can see that the yaw is closer to  $0^\circ$  at the start of the experiment and that the heading measured by the IMU is also closer to the GPS course.

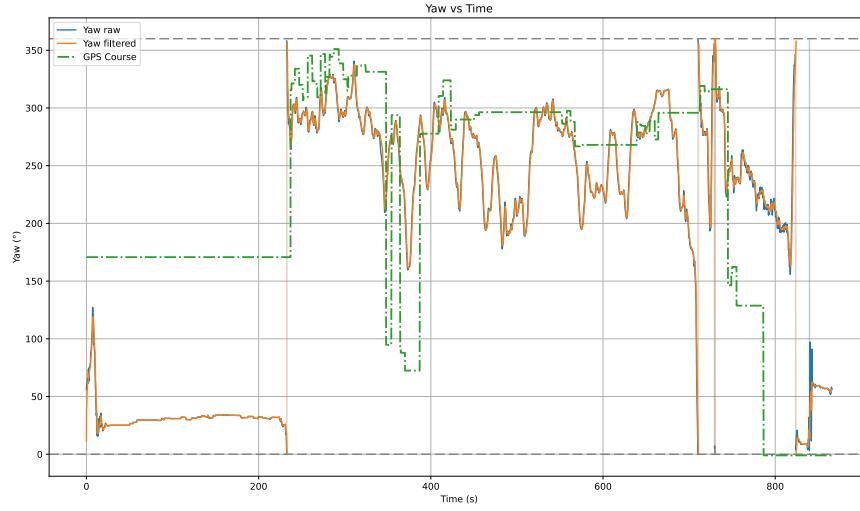


Figure 21: Heading and GPS course from the test conducted on 09/08/24

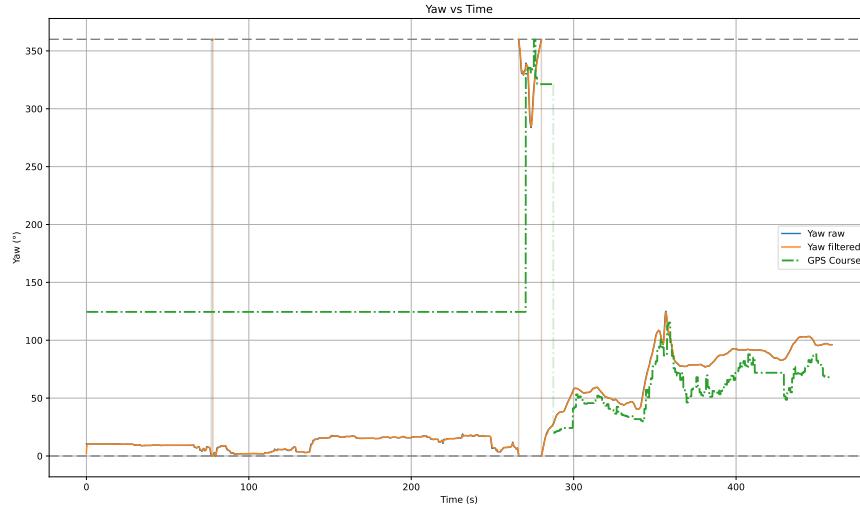


Figure 22: Heading and GPS course from the first test conducted on 15/08/24

The IMU sends a byte indicating its calibration status, which allows us to determine which elements are calibrated. This is important because different movements are required to calibrate the accelerometer, gyroscope, and magnetometer. To provide visual feedback on calibration progress, the rudder motor receives a command proportional to the value sent by the IMU, allowing the user to adjust movements, greatly simplifying an otherwise complex calibration process.

Finally, once the IMU issue was resolved, the software architecture was successfully validated over three consecutive tests (figures 23, 24, 25), with the full logs available in the Appendices and the videos generated with the logs [8–10]. It should be noted that during the third test, the boat missed the second waypoint. However, this was due to insufficient tuning of the algorithm rather than a sensor or software architecture problem, so the test was still considered a success as the goal was to validate the architecture, not the algorithm.

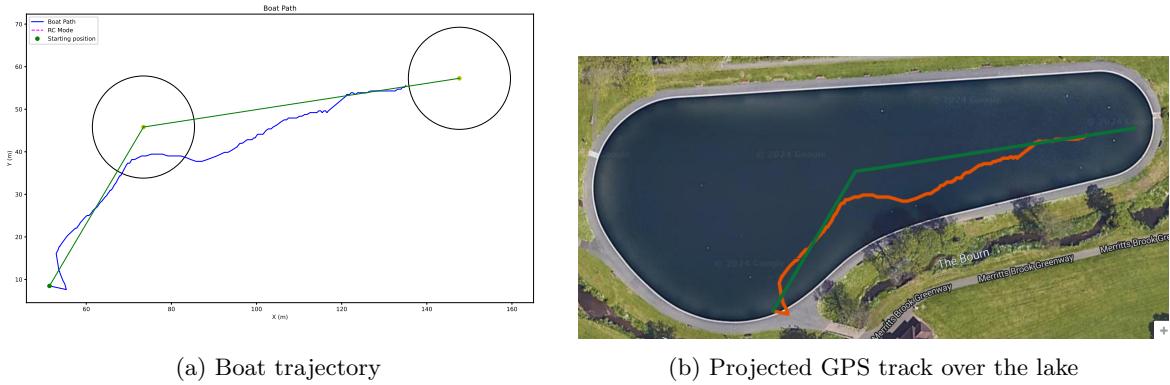


Figure 23: Readings from the first test conducted on 15/08/24

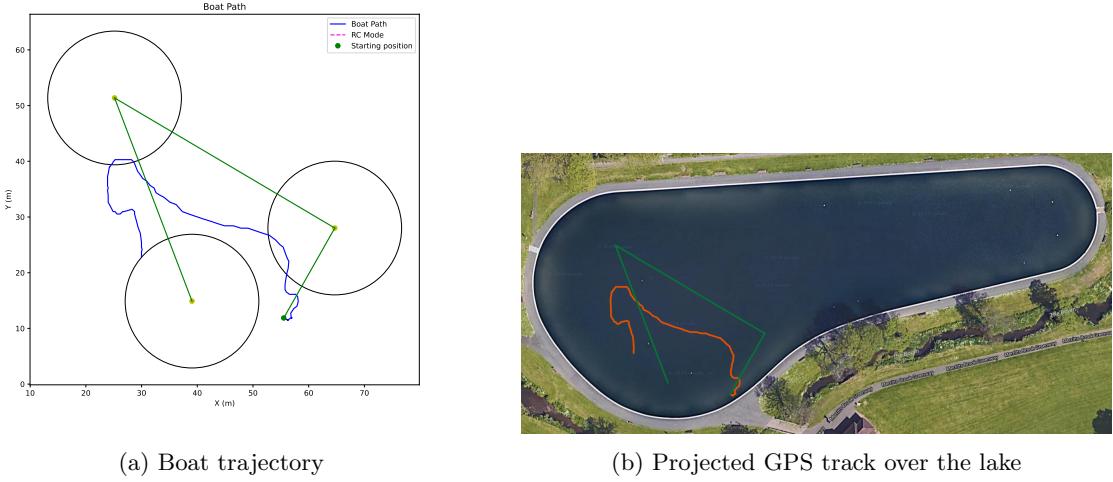


Figure 24: Readings from the second test conducted on 15/08/24

Additionally, due to time constraints, no type 2 algorithms were tested in the field. To validate the switching between algorithms, a dry test was conducted. For this, the condition for waypoint validation was replaced by a 20-second timer, and the algorithms used sent constant commands for easy differentiation. The execution duration of the type 2 algorithm was set to 10 seconds. It can be seen in figure 26 that the transition between algorithms is functioning properly.

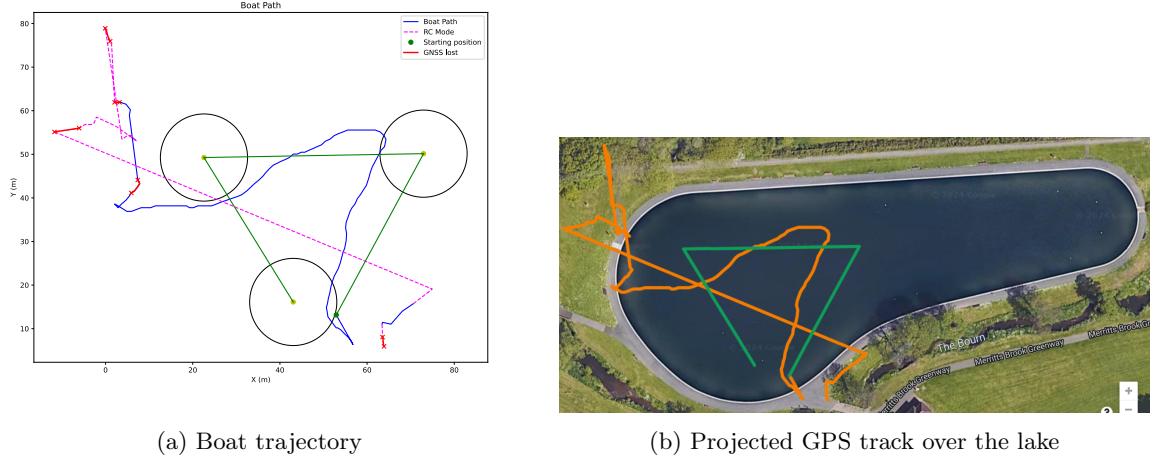


Figure 25: Readings from the third test conducted on 15/08/24

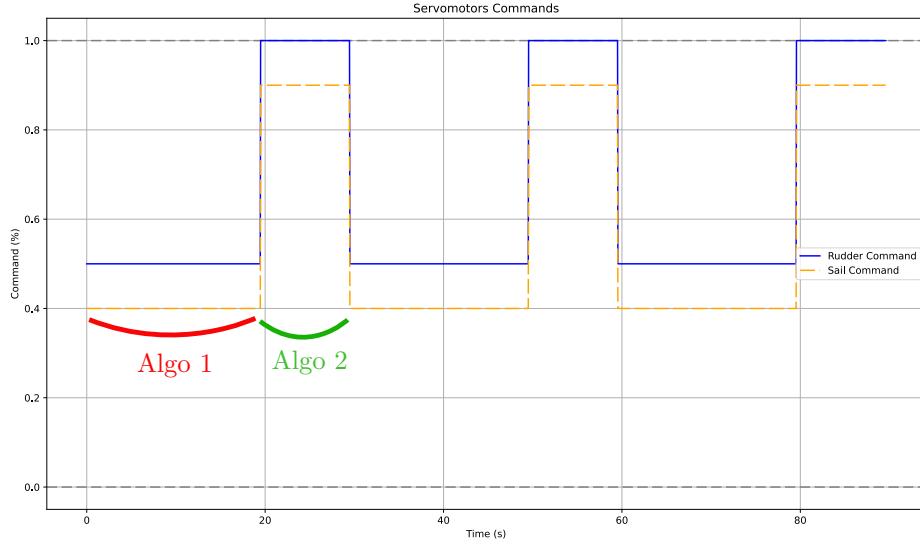


Figure 26: Commands send to the servomotors to illustrate the switch between the two types of algorithms

## 5 Conclusion

The objective of this project was to implement a software architecture using low-tech hardware to control an autonomous sailboat. This goal was achieved and experimentally validated. However, the work is not without its shortcomings, as the modularity of the project, though present, remains limited. For instance, only a portion of the IMU data is collected, and there is only one filter, which is not suitable for all types of data. A similar issue arises with data recording and display. A potential solution could be to add a way to select the data to use and record within the Config file.

Despite these limitations, working with modularity in mind made the code development process very interesting and quite different from what I was used to. Moreover, the intrinsic limitations of the low-tech aspect of the project pushed me to find innovative solutions, preventing me from relying solely on my prior knowledge. Additionally, significant effort has been put into the project's GitHub repository and documentation [11, 12], allowing anyone to adopt and build upon it. The prospect of sharing this project encouraged me to adopt a rigorous coding standard, ensuring my code is easily understandable.

Lastly, my involvement in this project gave me the opportunity to contribute to the writing of a scientific paper.

## References

- [1] Y. Tipsuwan, P. Sanposh and N. Techajaroontit, Overview and control strategies of autonomous sailboats—A survey, *Ocean Engineering* **281**, (2023).
- [2] Z. Liu *et al*, Unmanned surface vehicles: An overview of developments and challenges, *Annual Reviews in Control* **41**, p71-93 (2023).
- [3] Y-T. Ang *et al*, *An Autonomous Sailboat for Environment Monitoring* (IEEE, 2022).
- [4] A. Morge, *Autonomous Sailboat*, (2022).
- [5] Davis Instruments, *Anemometer for Weather Monitor or Wizard - SKU 7911*.
- [6] V. Pelle, *Development of an autonomous sailboat using Arduino*, p13 (2023).
- [7] L. Jaulin, F. Le Bars, *A simple controller for line following of sailboats*.
- [8] T. Leost, *Boat trajectory from the first test conducted on 15/08/24* (2024).
- [9] T. Leost, *Boat trajectory from the second test conducted on 15/08/24* (2024).
- [10] T. Leost, *Boat trajectory from the third test conducted on 15/08/24* (2024).
- [11] T. Leost, *Aston Autonomous Sailboat 2024* (2024).
- [12] T. Leost, *Documentation for Aston Autonomous Sailboat 2024* (2024).

## Appendices

### A Wiring diagrams

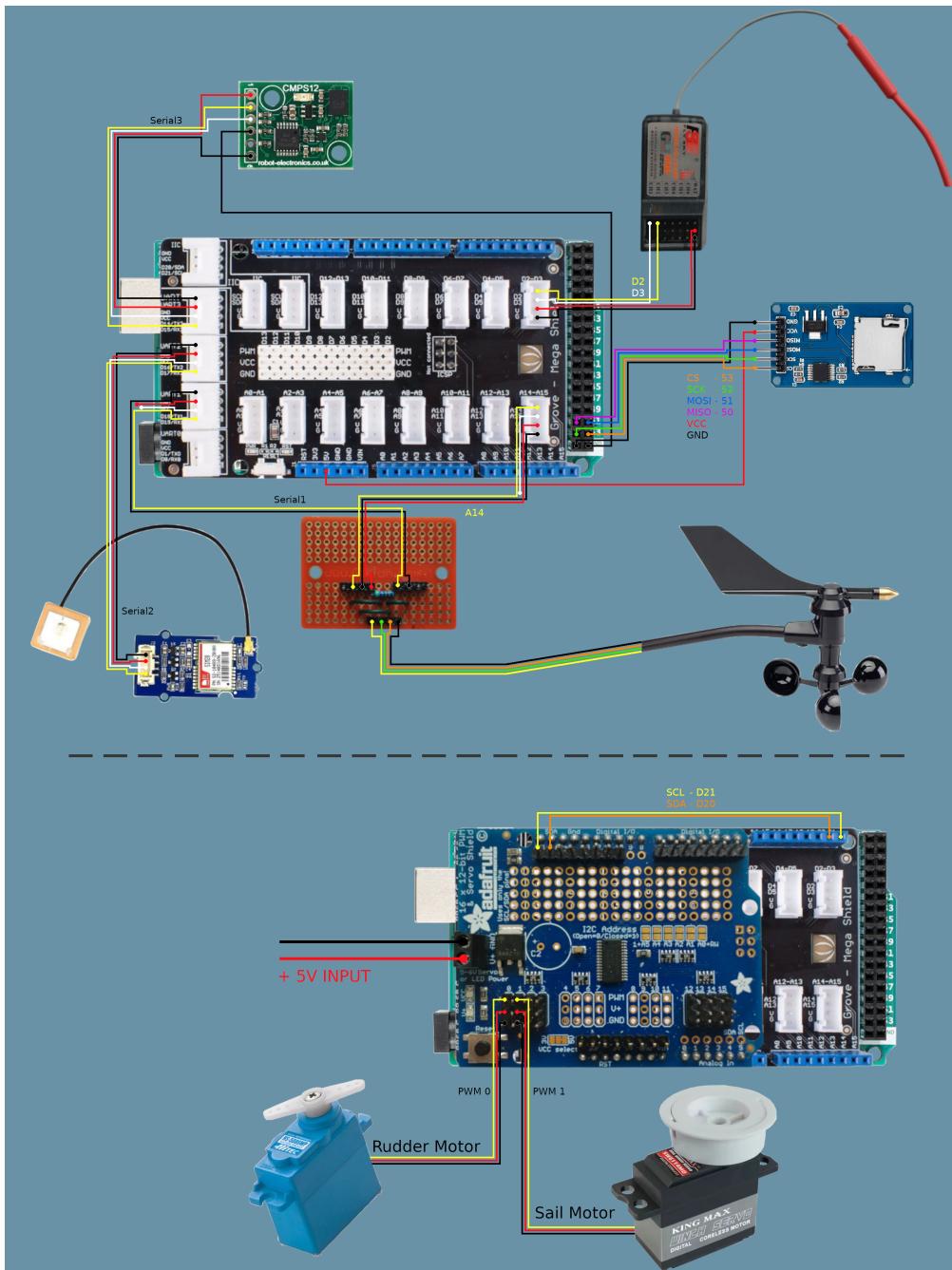


Figure 27: Arduino wiring diagram

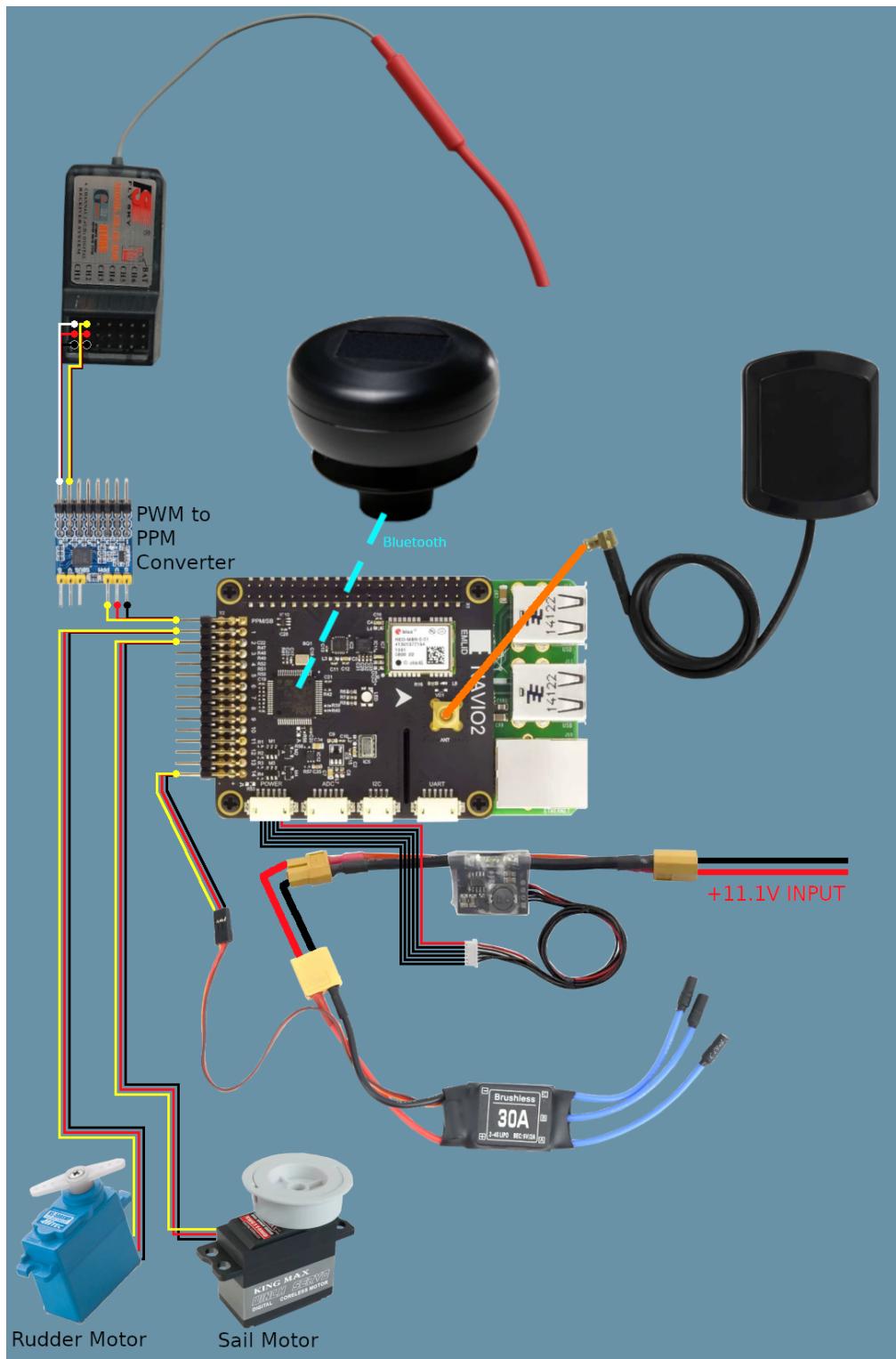


Figure 28: Raspberry Pi wiring diagram

## B Hardware list

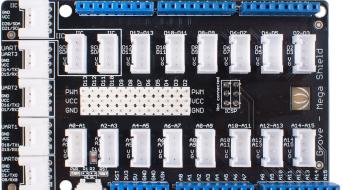
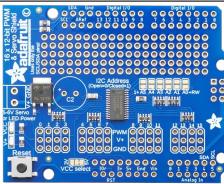
Name	Image	Power Requirement
Arduino Mega 2560		7-12V
Grove - Mega Shield		N/A
Adafruit 16 x 12-bit PWM & Servo Shield		N/A

Table 1: Circuit board/Shields used

Name	Image	Communication Protocol	Power Requirement
CMPS12		UART	3.3-5V 18mA
Grove GPS v1.2		UART	3.3V-5V
Anemometer - SKU 7911		1 Interrupt pin + 1 Analog pin	N/A

Table 2: Sensors used

Name	Image	Communication Protocol	Speed	Power Requirement
HS-5645MG		I2C-controlled PWM	0.23s@60°	4.8V
HS-785HB		I2C-controlled PWM	1.68s@60°	4.8V

Table 3: Actuators used

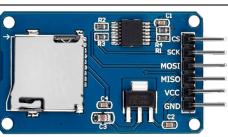
Name	Image	Communication Protocol	Power Requirement
FlySky FS-R6B		2 interrupt pins	4-6.5V
MicroSD Card Adapter		SPI	5V

Table 4: Extra

## C Class diagram

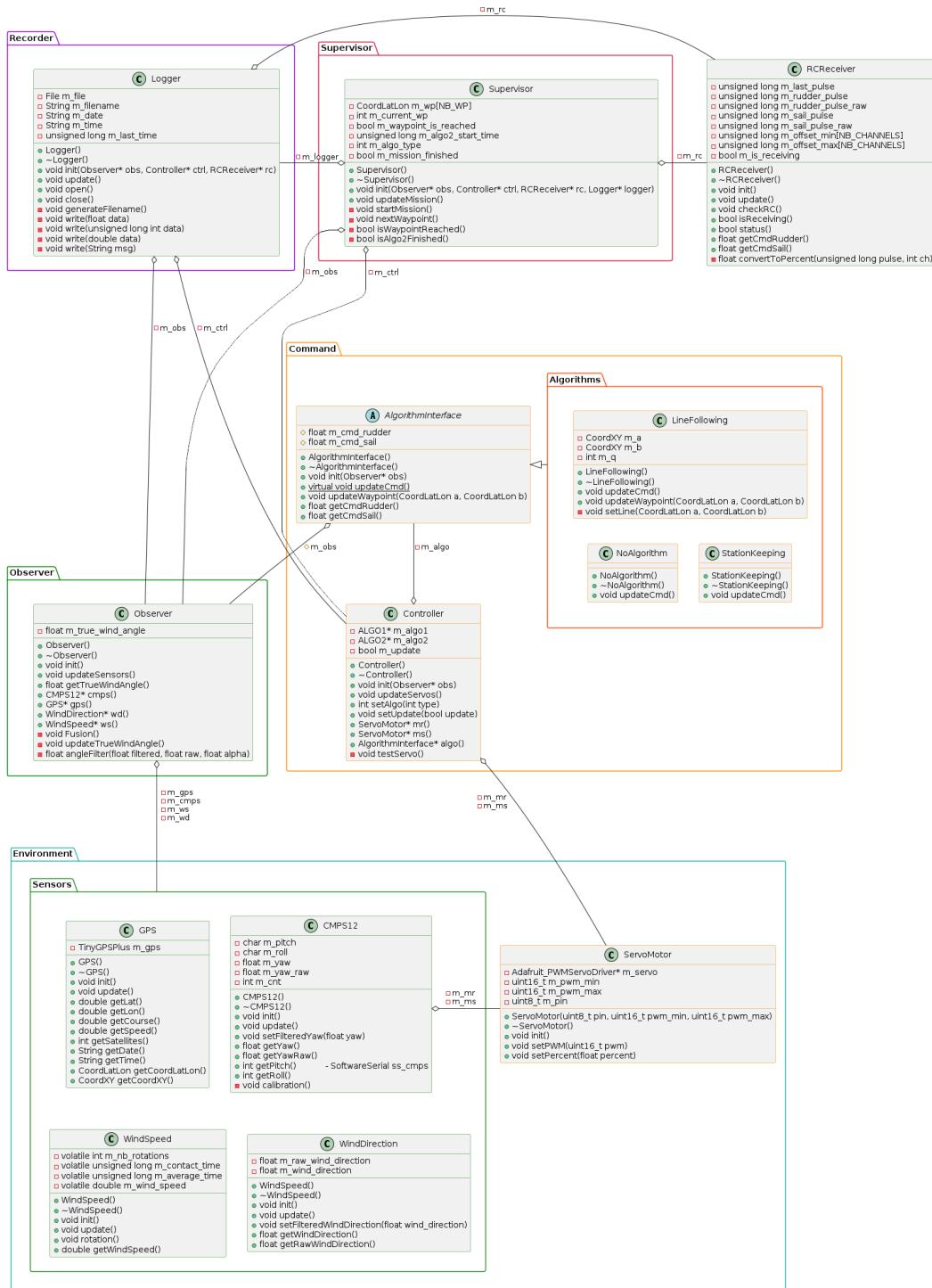


Figure 29: Class diagram

## D Final results

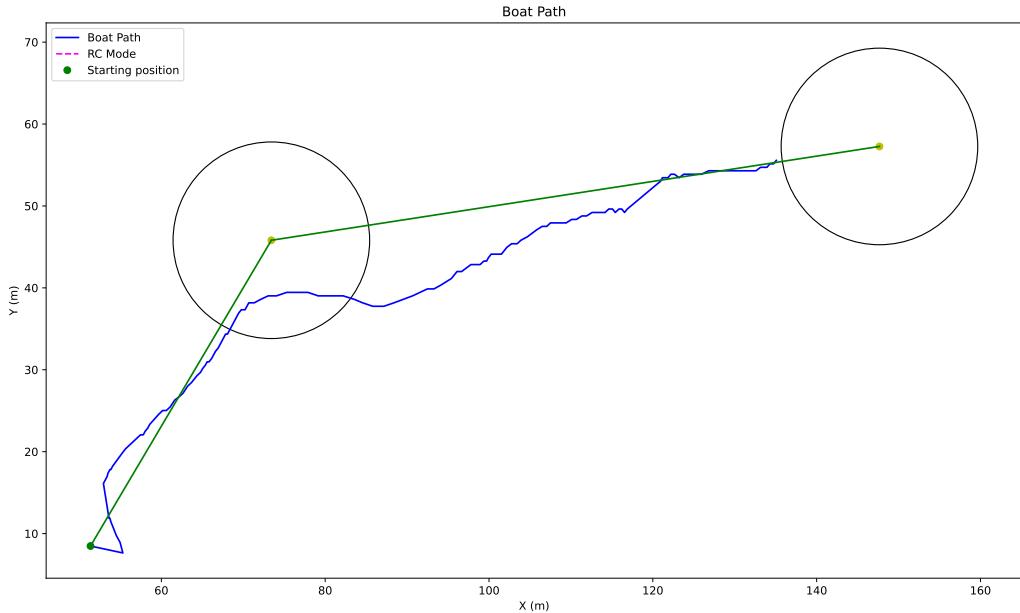


Figure 30: Boat trajectory from the first test conducted on 15/08/24



Figure 31: Projected GPS track over the lake from the first test conducted on 15/08/24

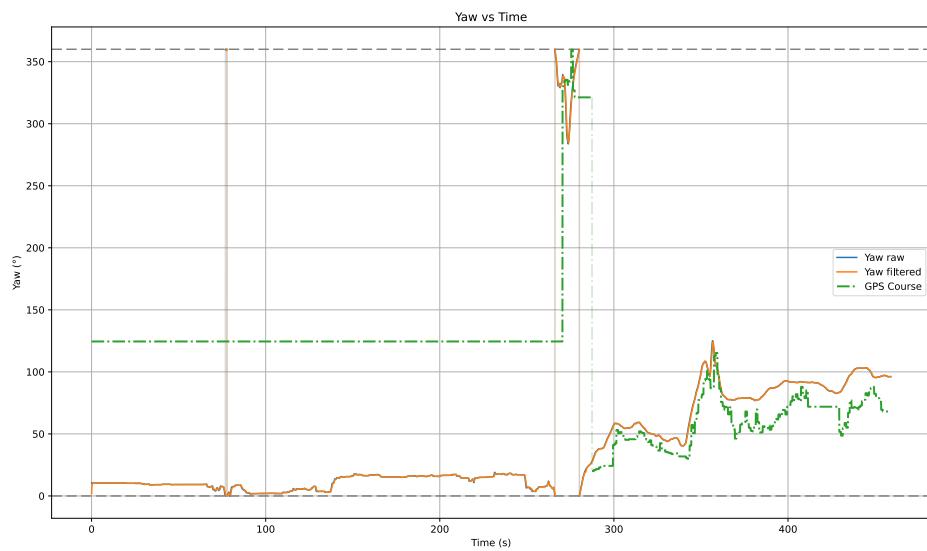


Figure 32: Heading and GPS course from the first test conducted on 15/08/24

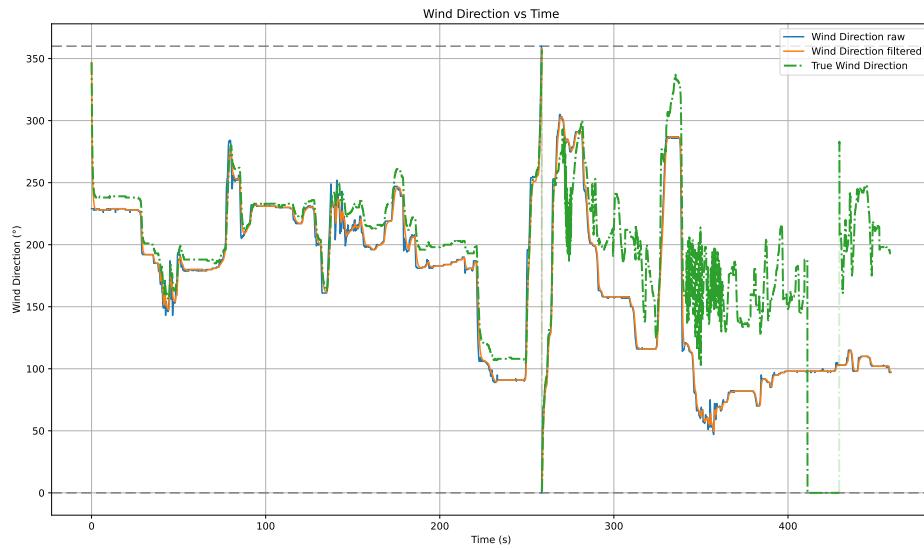


Figure 33: Wind direction from the first test conducted on 15/08/24

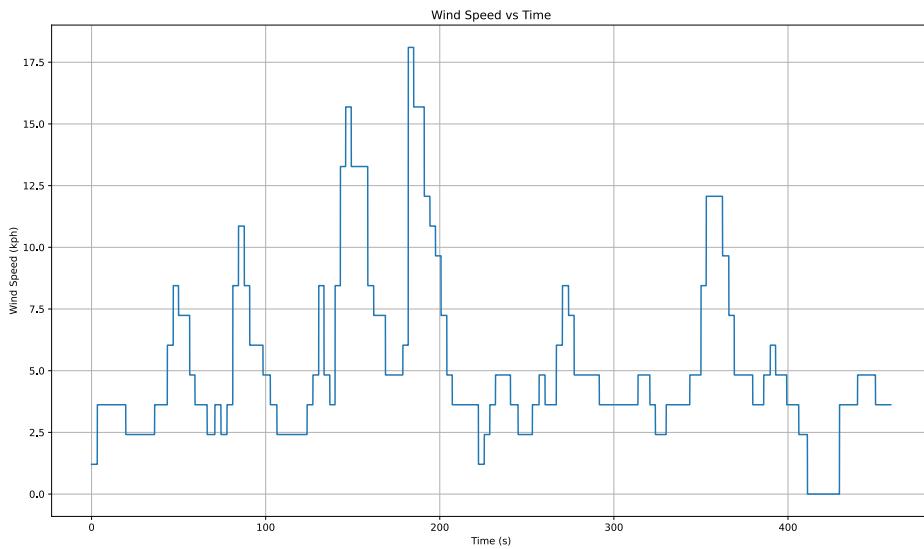


Figure 34: Wind speed from the first test conducted on 15/08/24

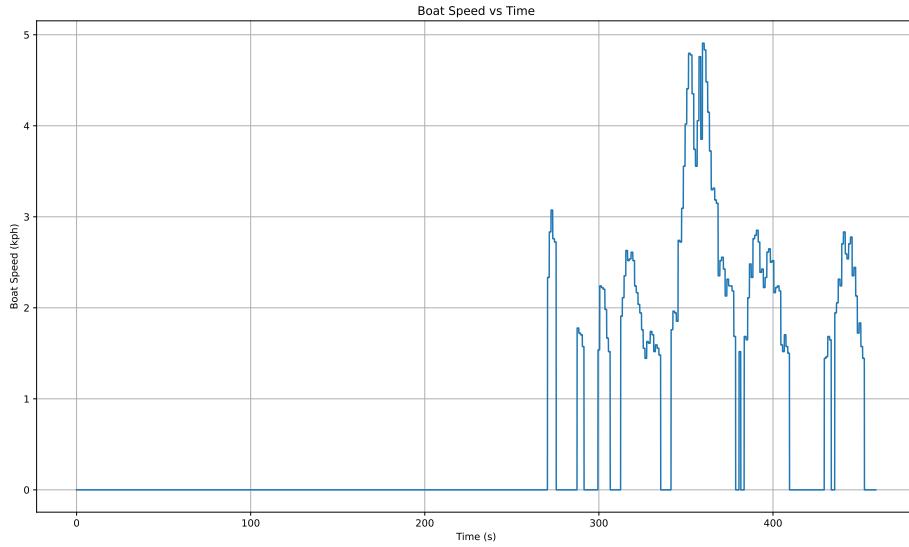


Figure 35: Boat speed from the first test conducted on 15/08/24

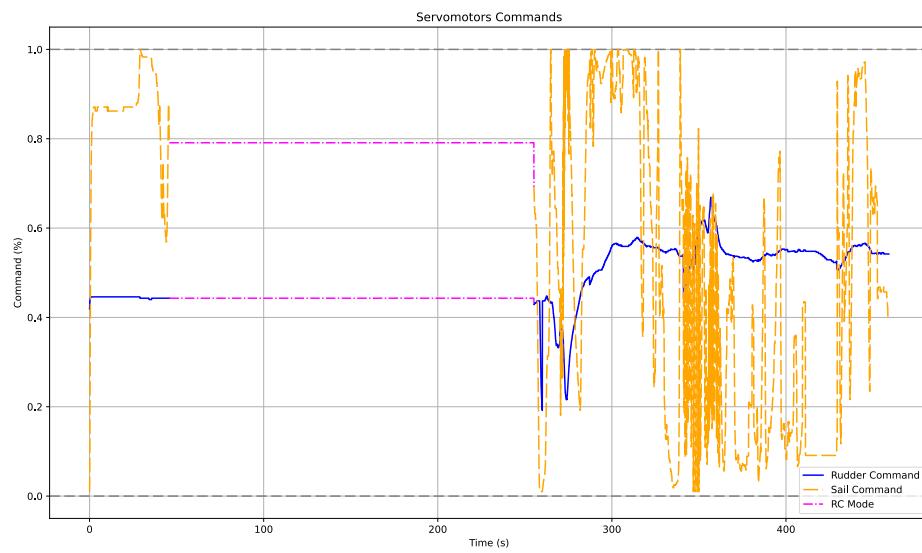


Figure 36: Commands sent to the servomotors from the first test conducted on 15/08/24

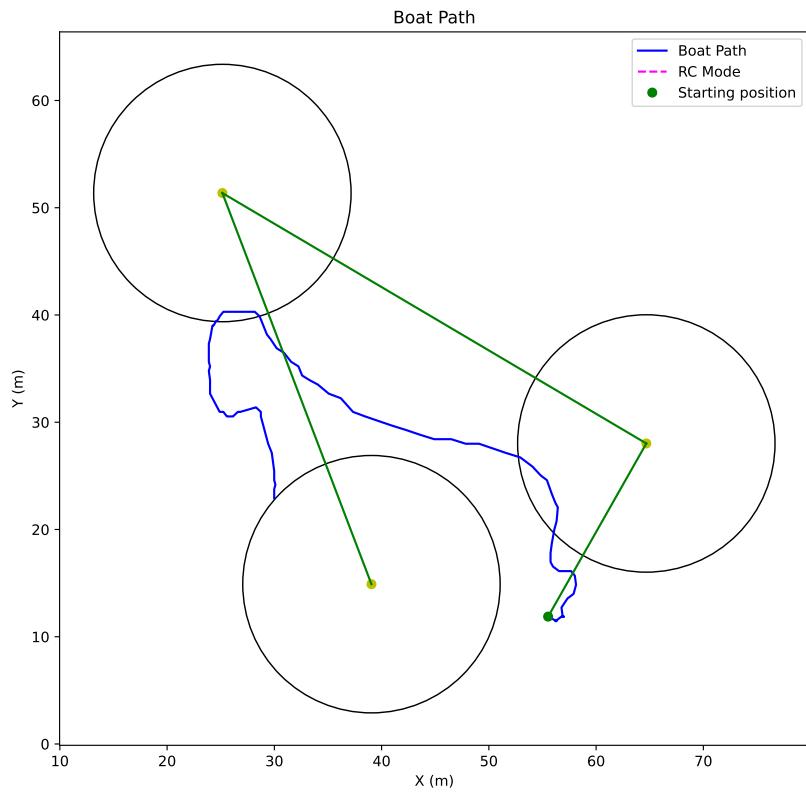


Figure 37: Boat trajectory from the second test conducted on 15/08/24



Figure 38: Projected GPS track over the lake from the second test conducted on 15/08/24

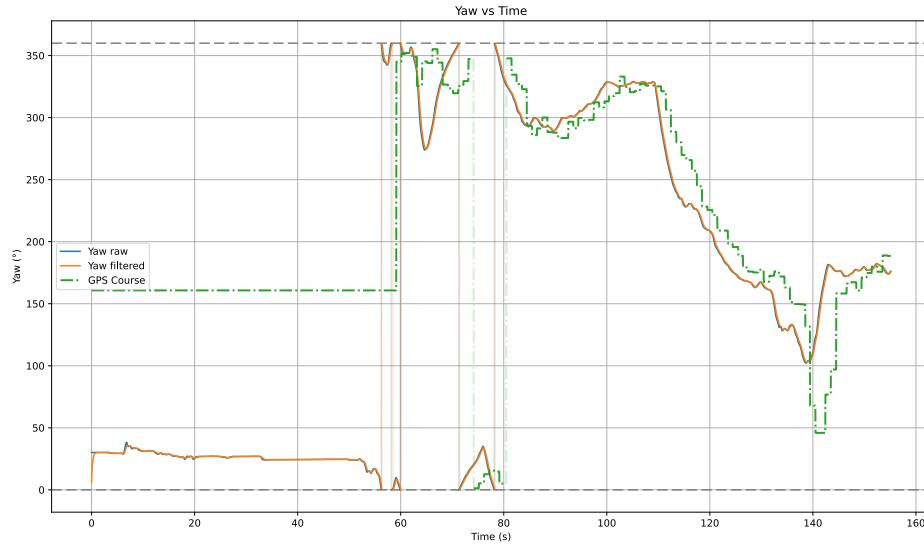


Figure 39: Heading and GPS course from the second test conducted on 15/08/24

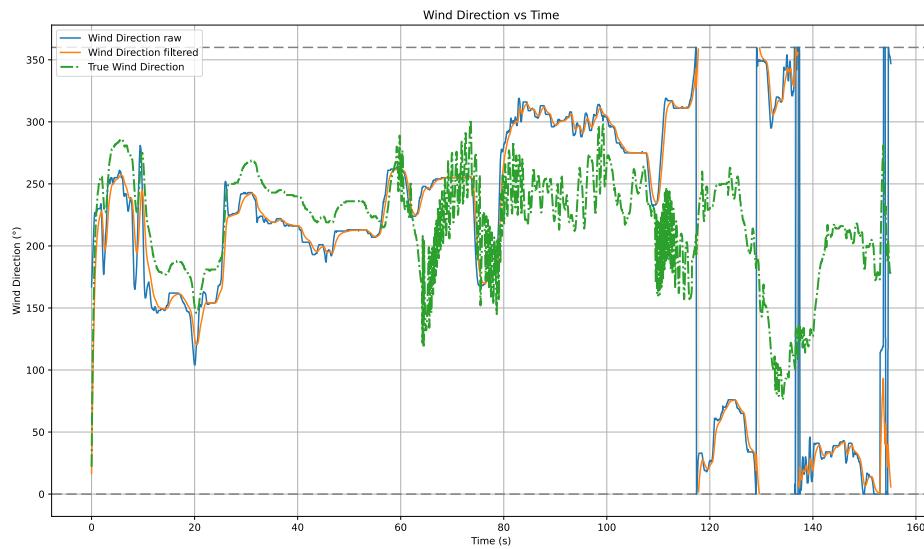


Figure 40: Wind direction from the second test conducted on 15/08/24

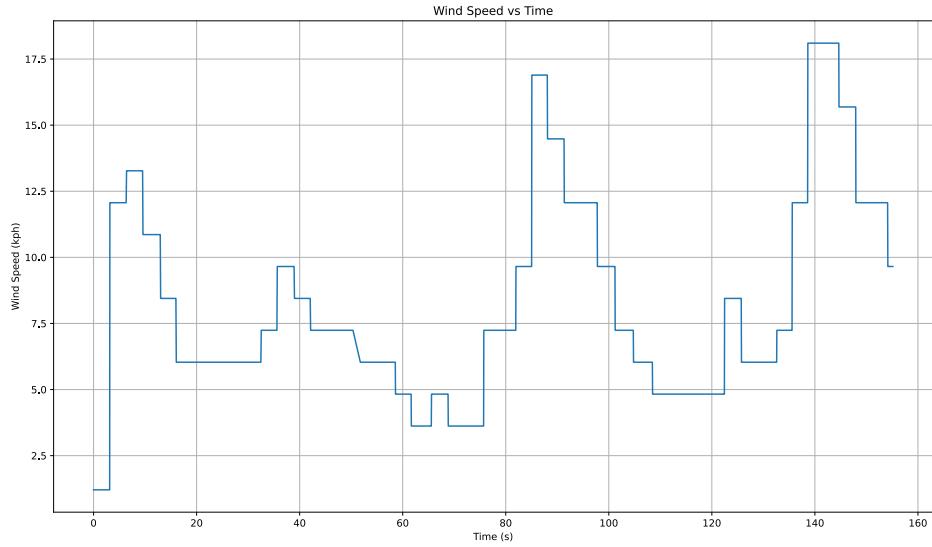


Figure 41: Wind speed from the second test conducted on 15/08/24

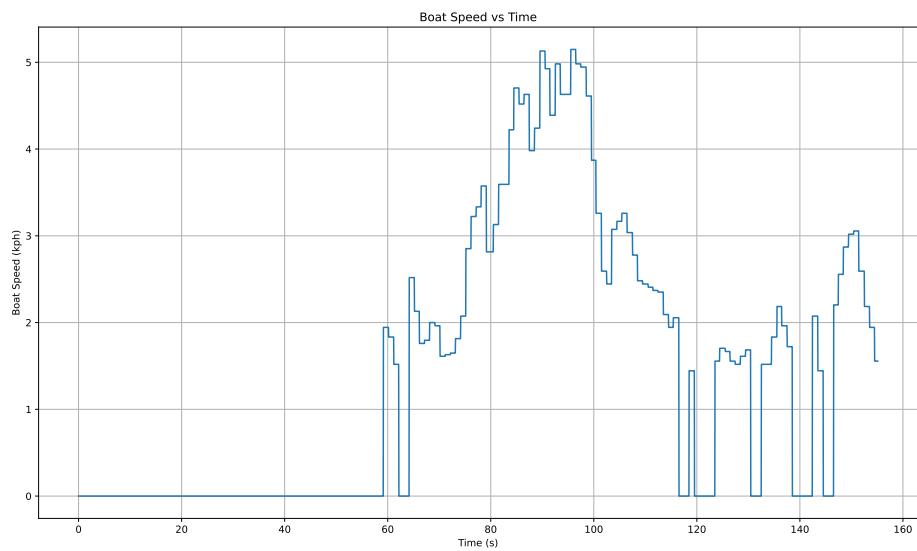


Figure 42: Boat speed from the second test conducted on 15/08/24

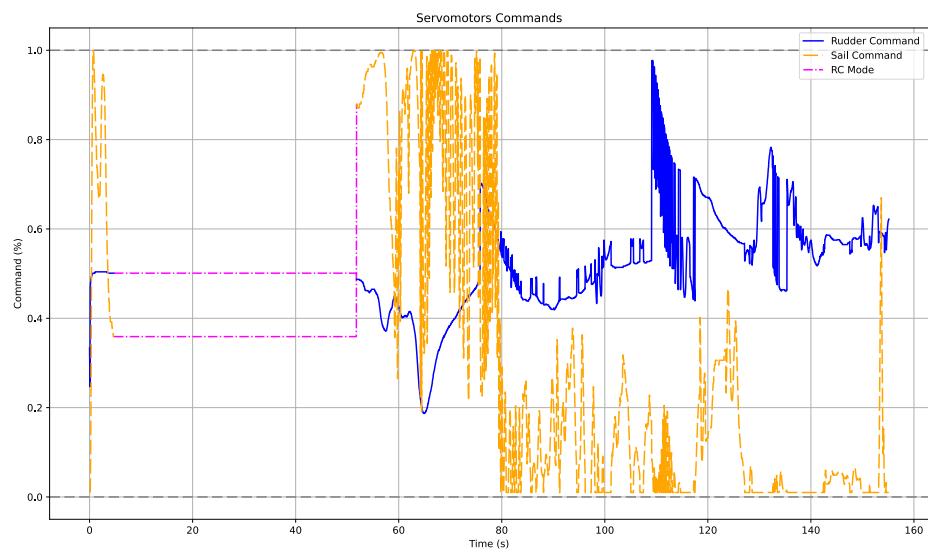


Figure 43: Commands sent to the servomotors from the second test conducted on 15/08/24

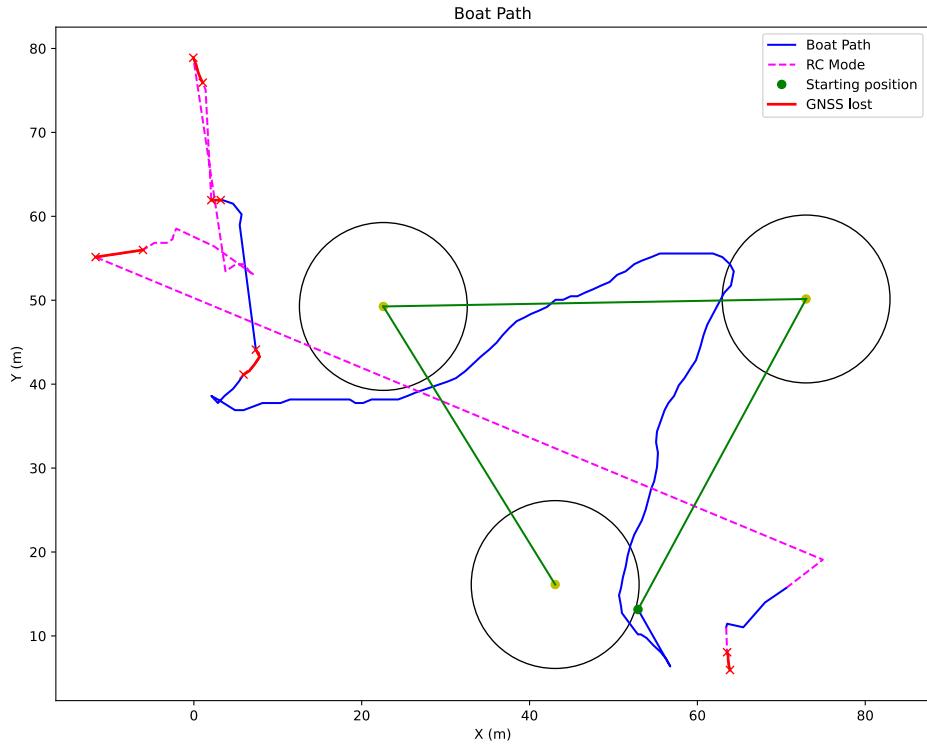


Figure 44: Boat trajectory from the third test conducted on 15/08/24

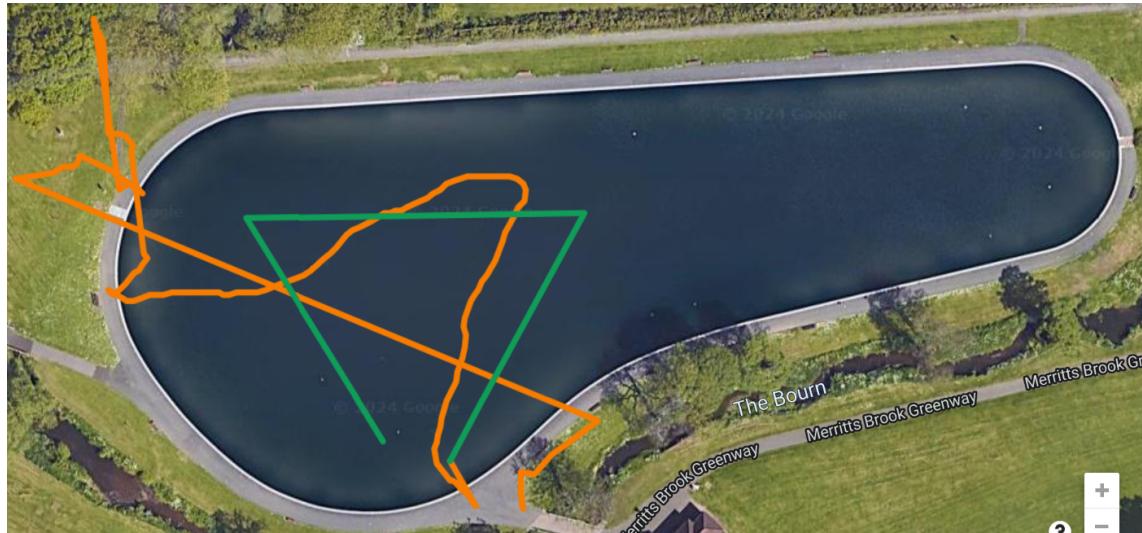


Figure 45: Projected GPS track over the lake from the third test conducted on 15/08/24

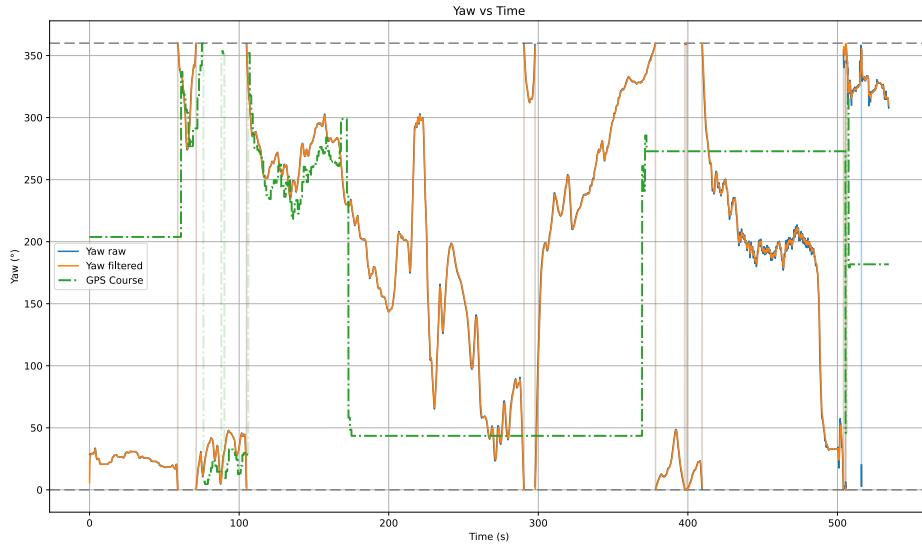


Figure 46: Heading and GPS course from the third test conducted on 15/08/24

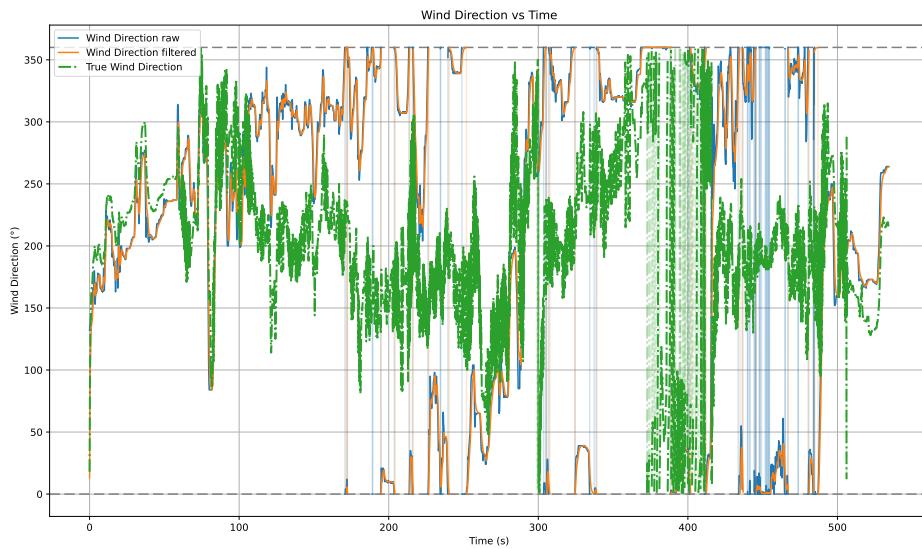


Figure 47: Wind direction from the third test conducted on 15/08/24

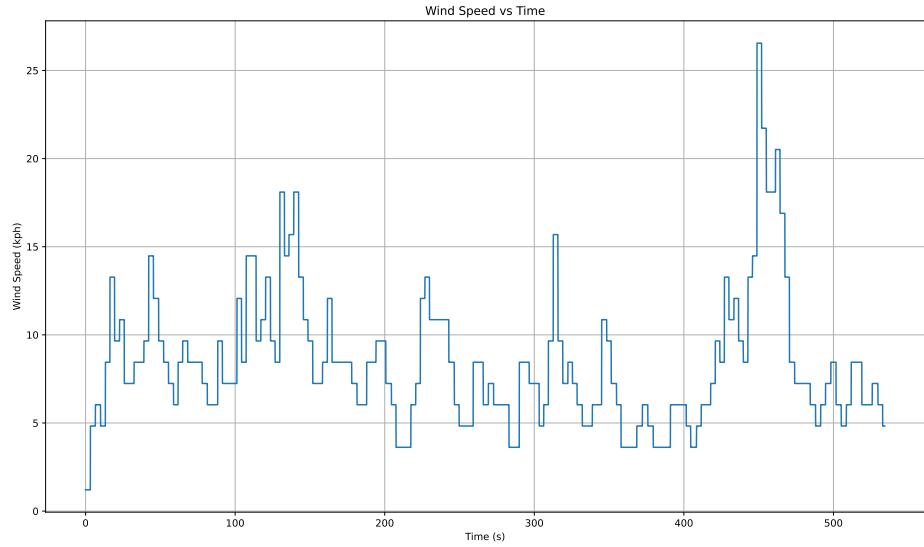


Figure 48: Wind speed from the third test conducted on 15/08/24

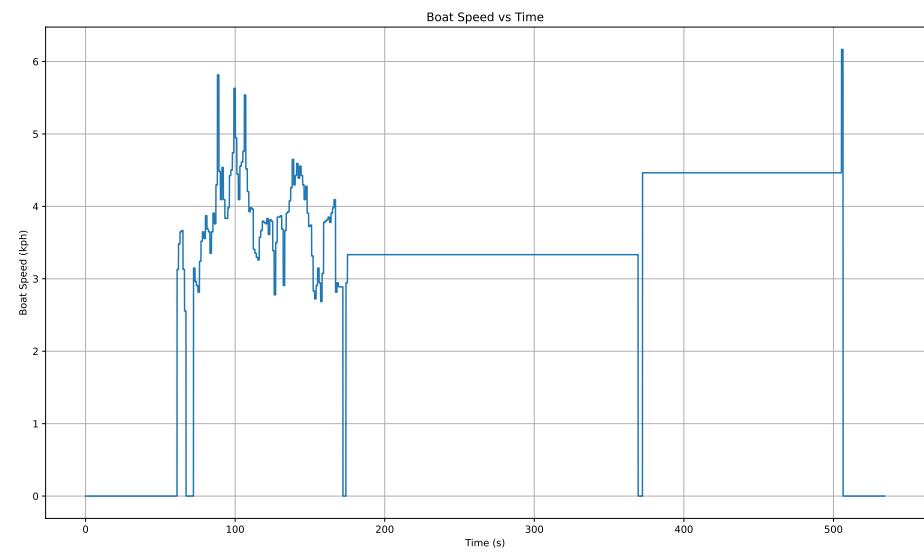


Figure 49: Boat speed from the third test conducted on 15/08/24

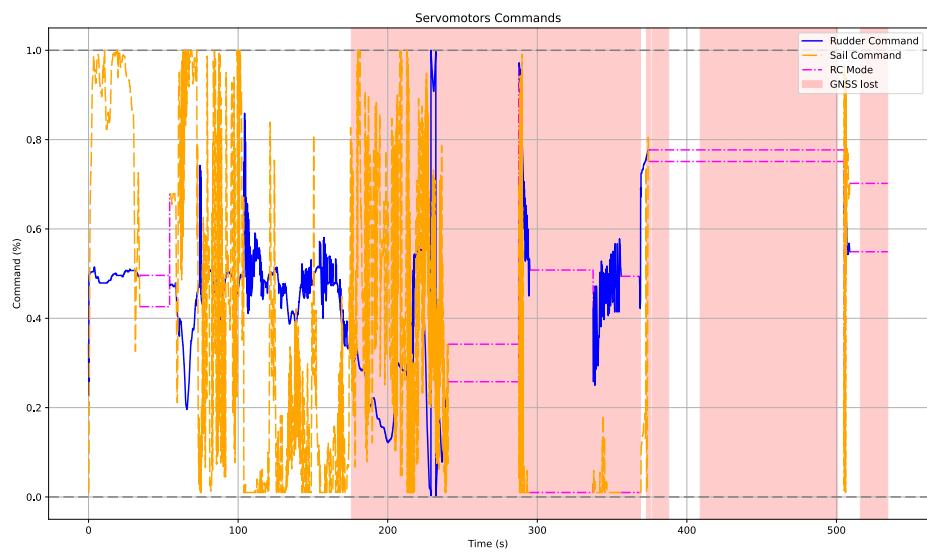


Figure 50: Commands sent to the servomotors from the third test conducted on 15/08/24

Merci de retourner ce rapport par courrier ou par voie électronique en fin du stage à :  
*At the end of the internship, please return this report via mail or email to:*

*ENSTA Bretagne – Bureau des stages – 2 rue François Verny – 29806 BREST cedex 9 – FRANCE*  
*■ 00.33 (0) 2.98.34.87.70 / [stages@ensta-bretagne.fr](mailto:stages@ensta-bretagne.fr)*

**I - ORGANISME / HOST ORGANISATION**

NOM / Name Aston University

Adresse / Address Aston Street, Birmingham, B4 7ET, UK

Tél / Phone (including country and area code) +44 07475104087

Nom du superviseur / Name of internship supervisor Jian Wan

Fonction / Function Lecturer in Mechanics and Robotics

Adresse e-mail / E-mail address jwan3@aston.ac.uk

Nom du stagiaire accueilli / Name of intern

Titoan LEOST

**II - EVALUATION / ASSESSMENT**

Veuillez attribuer une note, en encerclant la lettre appropriée, pour chacune des caractéristiques suivantes. Cette note devra se situer entre A (très bien) et F (très faible)  
*Please attribute a mark from A (excellent) to F (very weak).*

**MISSION / TASK**

- ❖ La mission de départ a-t-elle été remplie ?  A  B  C  D  E  F  
*Was the initial contract carried out to your satisfaction?*
- ❖ Manquait-il au stagiaire des connaissances ?  oui/yes  non/no  
*Was the intern lacking skills?*

Si oui, lesquelles ? / If so, which skills? \_\_\_\_\_

**ESPRIT D'EQUIPE / TEAM SPIRIT**

- ❖ Le stagiaire s'est-il bien intégré dans l'organisme d'accueil (disponible, sérieux, s'est adapté au travail en groupe) / Did the intern easily integrate the host organisation? (flexible, conscientious, adapted to team work)

A  B  C  D  E  F

Souhaitez-vous nous faire part d'observations ou suggestions ? / If you wish to comment or make a suggestion, please do so here \_\_\_\_\_

**COMPORTEMENT AU TRAVAIL / BEHAVIOUR TOWARDS WORK**

Le comportement du stagiaire était-il conforme à vos attentes (Ponctuel, ordonné, respectueux, soucieux de participer et d'acquérir de nouvelles connaissances) ?

*Did the intern live up to expectations? (Punctual, methodical, responsive to management instructions, attentive to quality, concerned with acquiring new skills)?*

A  B  C  D  E  F

Souhaitez-vous nous faire part d'observations ou suggestions ? / If you wish to comment or make a suggestion, please do so here \_\_\_\_\_

#### **INITIATIVE – AUTONOMIE / INITIATIVE – AUTONOMY**

Le stagiaire s'est-il rapidement adapté à de nouvelles situations ?

A B C D E F

(Proposition de solutions aux problèmes rencontrés, autonomie dans le travail, etc.)

*Did the intern adapt well to new situations?*

A  B  C  D  E  F

*(eg. suggested solutions to problems encountered, demonstrated autonomy in his/her job, etc.)*

Souhaitez-vous nous faire part d'observations ou suggestions ? / If you wish to comment or make a suggestion, please do so here \_\_\_\_\_

#### **CULTUREL – COMMUNICATION / CULTURAL – COMMUNICATION**

Le stagiaire était-il ouvert, d'une manière générale, à la communication ?

A  B  C  D  E  F

*Was the intern open to listening and expressing himself/herself?*

Souhaitez-vous nous faire part d'observations ou suggestions ? / If you wish to comment or make a suggestion, please do so here \_\_\_\_\_

#### **OPINION GLOBALE / OVERALL ASSESSMENT**

❖ La valeur technique du stagiaire était :

A  B  C  D  E  F

*Please evaluate the technical skills of the intern:*

#### **III - PARTENARIAT FUTUR / FUTURE PARTNERSHIP**

❖ Etes-vous prêt à accueillir un autre stagiaire l'an prochain ?

*Would you be willing to host another intern next year?*  oui/yes

non/no

Fait à \_\_\_\_\_, le \_\_\_\_\_  
In \_\_\_\_\_, on \_\_\_\_\_

Signature Entreprise Dianan  
Company stamp

Signature stagiaire  
Intern's signature



*Merci pour votre coopération  
We thank you very much for your cooperation*

## Acronyms

<b>ENU</b>	East North Up
<b>GND</b>	Ground
<b>GNSS</b>	Global Navigation Satellite System
<b>GPS</b>	Global Positioning System
<b>I2C</b>	Inter-Integrated Circuit
<b>IMU</b>	Inertial Measurement Unit
<b>PWM</b>	Pulse Width Modulation
<b>RC</b>	Radio Control
<b>SD</b>	Secure Digital
<b>SPI</b>	Serial Peripheral Interface
<b>UART</b>	Universal Asynchronous Receiver Transmitter
<b>USV</b>	Unmanned Surface Vehicle
<b>VCC</b>	Voltage Common Collector