



ENSTA Bretagne  
2, rue François Verny  
29806 BREST cedex  
FRANCE  
Tel +33 (0)2 98 34 88 00  
[www.ensta-bretagne.fr](http://www.ensta-bretagne.fr)

Reaport  
ROB 2024  
September 28, 2023

# Engineering Assistant Internship

*College of Engineering and Physical Sciences*

*Aston University*

Supervisor: Jian WAN

Conception and Control  
of an  
Autonomous Sailboat

Ludovic MUSTIÈRE  
Engineering Student in Autonomous Robotics  
[ludovic.mustiere@ensta-bretagne.org](mailto:ludovic.mustiere@ensta-bretagne.org)

# Contents

|  |           |
|--|-----------|
| <b>Introduction</b>  | <b>5</b>  |
| Engineering Assistant Internship in ENSTA Bretagne . . . . . | 5         |
| Aston University . . . . .                                   | 5         |
| Autonomous Sailboats . . . . .                               | 6         |
| <b>1 Context</b>   | <b>8</b>  |
| 1.1 Autonomous Sailboat Project . . . . .                    | 8         |
| 1.2 Objectives of the internship . . . . .                   | 9         |
| 1.3 Stakes of the internship . . . . .                       | 9         |
| <b>2 System Description</b>                                  | <b>10</b> |
| 2.1 Hardware . . . . .                                       | 10        |
| 2.1.1 Sailboat . . . . .                                     | 10        |
| 2.1.2 Navio2 . . . . .                                       | 11        |
| 2.1.3 Raspberry Pi 4 . . . . .                               | 11        |
| 2.1.4 Servomotors . . . . .                                  | 11        |
| 2.1.5 Radio Control . . . . .                                | 11        |
| 2.2 Software . . . . .                                       | 12        |
| 2.2.1 OS . . . . .   | 12        |
| 2.2.2 Python . . . . .                                       | 12        |
| <b>3 Drivers</b>   | <b>14</b> |
| 3.1 Modularity . . . . .                                     | 14        |
| 3.2 Calypso Anemometer . . . . .                             | 14        |
| 3.3 Multithreading . . . . .                                 | 14        |
| 3.4 Package . . . . .  | 15        |
| 3.5 Kalman Filter . . . . .                                  | 16        |
| 3.6 Sailboat Class . . . . .                                 | 16        |
| <b>4 Control Algorithms</b>                                  | <b>17</b> |
| 4.1 Apparent Wind . . . . .                                  | 17        |
| 4.2 Tacking . . . . .  | 18        |
| 4.3 Line Following . . . . .                                 | 19        |
| 4.4 Station Keeping . . . . .                                | 19        |
| <b>5 Simulation</b>  | <b>21</b> |
| 5.1 Model . . . . .  | 21        |
| 5.2 Configuration Files . . . . .                            | 21        |
| 5.3 Graphical User Interface . . . . .                       | 22        |
| <b>6 Test and Results</b>                                    | <b>24</b> |
| 6.1 Test Location . . . . .                                  | 24        |
| 6.2 Line Following Log Analysis . . . . .                    | 24        |
| 6.3 Station Keeping Log Analysis . . . . .                   | 25        |

|                             |          |
|-----------------------------|----------|
| <b>Annexes</b>              | <b>i</b> |
| List of Acronyms . . . . .  | i        |
| List of Figures . . . . .   | ii       |
| List of Tables . . . . .    | iii      |
| References . . . . .        | iv       |
| Assessment Report . . . . . | v        |
| Config Files . . . . .      | viii     |

## Résumé

Durant ma deuxième année d'étude en robotique autonome à l'École Nationale Supérieure des Techniques Avancées Bretagne (ENSTA Bretagne), j'ai réalisé un stage en tant qu'assistant ingénieur au sein du College of Engineering and Physical Sciences de l'Université d'Aston, à Birmingham, en Angleterre. Il s'est déroulé du 01 mai 2023 au 18 août 2023, sous la supervision de JIAN WAN, maître de conférence en conception technique, mécanique et biomédicale. L'objectif était d'autonomiser un voilier miniature et d'expérimenter différents algorithmes de contrôle.

## Abstract

During my second year of study in autonomous robotics at École Nationale Supérieure des Techniques Avancées Bretagne (ENSTA Bretagne), I carried out an internship as an assistant engineer at the College of Engineering and Physical Sciences at Aston University, Birmingham, England. It took place from May 01, 2023 to August 18, 2023, under the supervision of JIAN WAN, senior lecturer in engineering, mechanical and biomedical design. The aim was to empower a miniature sailboat and experiment with different control algorithms.

## Thanks

First, I would like to thank my tutor JIAN WAN for allowing me to carry out an internship under his supervision. His advices were always precious for the success of this project. With his help, patience and care, I surpassed myself and my limits.

Then, I would like to thank Aston University for the warm welcome they gave me and their help in administrative procedures. The working environment was ideal and is, for sure, partially responsible for the good results of this project.

Finally, I would like to thank Catherine Rizk and Harendra Rangaradjou for their kindness and their hard work. They were the best partners I could have hoped for and their presence was precious when obstacles and difficulties stood in our way. Thanks to them, we've gone further than we ever expected.



## Introduction

### Engineering assistant internship at ENSTA Bretagne

ENSTA Bretagne's engineer cursus covers the last year of the Bachelor's degree and both years of the Master's degree. The Engineering assistant internship takes place one year after the specialization in autonomous robotics which happened during the first year of the Master's degree.

The goal of the Engineering assistant internship in the first year of the Master's program is to allow the student to confront his or her knowledge with the activities of the internationalized industrial and technological sectors corresponding to the openings in the program and to put this knowledge into perspective for the final phase of professionalization that constitutes the third and final year of the program.

### Aston University

Aston University is a public research university situated in the city center of Birmingham, England. They focus on high quality, translatable research that has an impact on the real world around us. They have some of the highest quality labs and computer centers to complete those pioneering research. The campus is home to five schools (Business, Engineering and Applied Science, Languages and Social Sciences Life and Health Sciences and the Medical School) offering foundation, undergraduate, postgraduate taught and research programmes. It has over 11,000 undergraduates and 2,000 postgraduate students from over 120 countries [3]. Aston University is a pioneer research center about subjects revolving around ecology and sustainable development. The working environment is ideal, with an emphasis on the well-being of students and lecturers.



Figure 1: Aston University, Birmingham, England.

The Mechanical Engineering & Design Department is part of the College of Engineering and Physical Sciences. Its main focus revolves around the design and

development of new technologies and systems. However, the aim of the laboratories is to provide cutting-edge researches around various subjects such as naval architecture and autonomous robotics with the Autonomous Sailboat Project lead by JIAN WAN. It is composed of a study room which can also be used as a classroom, two meeting rooms and a large number of offices to accommodate researchers and students.

## Autonomous Sailboats

Sailing boats were the most common means of transport for thousands of years, enabling people to explore and travel the world until the advent and democratization of the airplane. Today, they are used only for cruises and competitions, and have almost all been replaced by faster, more efficient motorboats.



Figure 2: Louis-Philippe Crépin, *Le Redoutable à Trafalgar*, 1807, Musée national de la Marine. *The Redoutable* at the battle of Trafalgar, between the *Victory* on the left and the *Temeraire* on the right.

Nowadays, ships are mainly used to transport goods or raw materials. Container ships therefore logically make up the majority of fleets, with over 100,000 vessels worldwide carrying more than 10 billion tonnes in 2017 [12]. Their mechanical propulsion means they are not dependent on weather conditions, and maneuver easily despite their gigantic dimensions. In recent years, a great deal of research and experimentation has gone into the use of autopilots and semi-autonomous navigation.

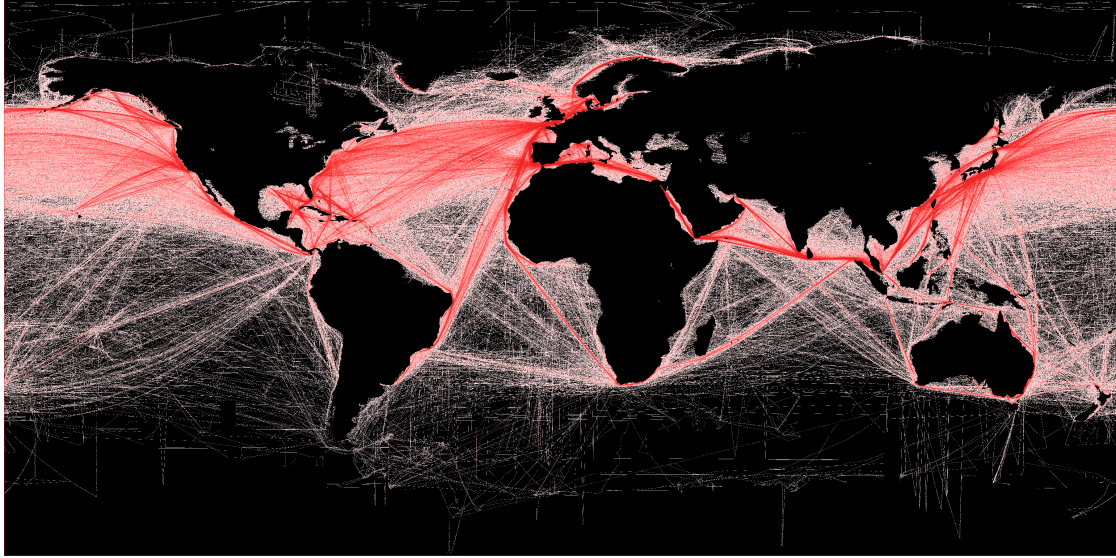


Figure 3: Shipping density (commercial). A Global Map of Human Impacts to Marine Ecosystems, showing relative density (in color) against a black background. Scale: 1 km. More details available at:

<http://spatial-analyst.net/worldmaps/shipping.rdc>

However, the ecological impact of these giant vessels is extremely harmful to the planet, and viable alternatives are urgently needed. Autonomous sailboats could provide a solution to these problems. Using the wind as a means of propulsion, their energy consumption would be extremely low. The main problem is to ensure that these vessels can withstand the extreme conditions of the ocean and increase their transport load. Recently, interest in autonomous sailing vessels has been growing. Several groups have already deployed vessels to retrieve data, or have completed the Microtransat challenge.



# 1 Context

## 1.1 Autonomous Sailboat Project

The Autonomous Sailboat Project consists in studying, designing, prototyping and testing the functionalities of an unmanned navigation system and base its integration on a modular open source system. The project will use a modular architecture to perform a wide range of missions, to allow upgrades and modifications of functionalities and to use the same system on various platforms such as aerial and ground platforms.

The project can be divided into two subparts, the mechanical part and the software part. The mechanical part consists in designing and building the sailboat by choosing the right components and sensors and integrating them into it. The software part consists in programming the algorithms that will control the boat and drivers that will read the sensors data, and to test them in simulation and in real conditions.



Figure 4: Ragazza 1 Meter Sailboat V2: RTR

The mechanical part of the project has already been done by the previous interns. The sailboat is based on a Ragazza™ 1-Meter Sailboat RTR [10] model from PROBOAT like the one in the Figure 4. Its hull design is based on full-scale racing hulls and to meet the International One Metre (IOM) class regulations [8]. It is exactly 1 meter long

and has a 1.65 meter mast. The hull is made of fiberglass and the mast of carbon fiber. The sailboat is equipped with a 2.4GHz radio system and a 3kg high torque waterproof rudder servo. The heavy duty sail winch was also pre-installed. In order to make the boat autonomous, a solar powered bluetooth ULTRASONIC Calypso anemometer [14] was added at the rear of the boat. Finally, a Raspberry Pi 4 and a Navio2 autopilot Hardware Attached on Top (HAT) [6] were added to control the boat. The power supply is provided by a two Lithium Polymer (LiPo) 11.1V 3S batteries.

The software part of the project is the one that we developed during the internship. It consists in programming the algorithms that will control the sailboat and the drivers that will read the sensors data. The control algorithms are implemented will be a station keeping algorithm and a path following algorithm. The station keeping algorithm will allow the boat to move to a given area and to stay in it. The path following algorithm will allow the boat to follow a given line. The drivers will read the data from the sensors and send them to the control algorithms. The sensors used are the Global Navigation Satellite System (GNSS), the Inertial Measurement Unit (IMU), the anemometer and the compass. Finally, a simulation will be developed to test the algorithms in a virtual environment.

## **1.2 Objectives of the internship**

The objectives of the internship are to program the control algorithms, to ensure the proper functioning of each of the sensors, to develop a simulator and to test the algorithms in simulation and in real conditions on a lake near Aston University. If the objectives are achieved, the sailboat will be able to perform the missions mentioned above. The next and last objective will be to develop new control algorithms using Machine Learning to increase the performance of the sailboat. The sailboat will then be able to perform more complex missions in various environmental conditions.

## **1.3 Stakes of the internship**

The stakes of the internship are to export the pluridisciplinarity of ENSTA Bretagne internationally, to develop an innovative tool for research and for the future. The sailboat will be used by the students of Aston University for their research projects. The sailboat will also be used for the IOM competitions in France and in England. My experience as an ENSTA Bretagne student in autonomous robotics will be very useful for the project. I will be able to use the knowledge I acquired during my studies to develop the algorithms and to understand the functioning of the sensors. Furthermore, I already worked with autonomous boats during my scholarship.

## 2 System Description

### 2.1 Hardware

#### 2.1.1 Sailboat

The Ragazza™ 1-Meter Sailboat RTR like the one in the Figure 5 is composed of four parts:

- A **hull**: It represents the body of the boat. It is the part that is in contact with both the water and the air. It is made of fiberglass and is optimized to reduce the drag of the boat, which is the force that opposes the movement of the boat. It also ensures the buoyancy and the waterproofness of the boat. Inside the hull are all the electronic components of the boat [16].
- A **keel**: It is a fin that is attached to the bottom of the hull. It is used to prevent the boat from drifting sideways and to provide stability and control. It counteracts the force of the wind on the sails.
- A **rudder**: It is a fin that is attached to the rear of the hull. It is used to steer the boat. It is controlled by a servo motor which is inside the hull. It rotates between  $-\pi/4$  and  $\pi/4$  radians.
- A **sail**: It is the part that captures the wind to propel the boat. It is attached to the mast and can rotate around it. It is controlled by a servo motor which is inside the hull. It rotates between 0 and  $\pi/2$  radians.



Figure 5: Ragazza 1 Meter Sailboat V2: RTR with his four main parts

### 2.1.2 Navio2

The Navio2 like the one in the Figure 6 is an autopilot HAT for the Raspberry Pi 4. It is compatible with the Raspberry Pi 3 and 3+ but the Raspberry Pi 4 is recommended for better performance. It is equipped with a GNSS receiver, an IMU, a barometer, a Pulse Position Modulation (PPM) encoder, a Pulse Width Modulation (PWM) driver and a Radio Control (RC) I/O co-processor [4]. It is used to send the command to the sailboat servomotors and to read the data from the sensors. It is connected to the Raspberry Pi 4 via the General Purpose Input/Output (GPIO) pins and is powered by it. It is also connected to a GNSS antenna via a Micro Coaxial (MCX) connector.



Figure 6: Navio2 HAT on top of a Raspberry Pi 4

### 2.1.3 Raspberry Pi 4

The Raspberry Pi 4 like the one in the Figure 6 is a single-board computer. It is used to run the control algorithms and the drivers. It is connected to the Navio2 via the GPIO pins and is powered by an external battery. It runs on the latest Emlid Raspberry Pi Operating System (OS) Buster image [11] in order to be compatible with the Navio2.

### 2.1.4 Servomotors

The sailboat is equipped with two servomotors, one for the sail and one for the rudder. They are both controlled by the Navio2 via the PWM driver. The servomotor for the sail is a Hitec HS-785HB. It is a heavy duty servo with a maximum torque of 11.1 kg-cm. It is used to control the sail angle. The servomotor for the rudder is a Hitec HS-5645MG. It is a high torque servo with a maximum torque of 9.6 kg-cm. It is used to control the rudder angle. They are both powered by an Electronic Speed Controller (ESC) and a their own LiPo battery due to their high power consumption.

### 2.1.5 Radio Control

The radio control is used to control the sailboat manually. It is composed of a transmitter and a receiver. The transmitter is a Hobbyking HK-T4A like the one in the Figure 7. It is used to send the commands to the receiver. The receiver is a Hobbyking HK-TR6A V2 2.4GHz Radio Receptor. It is used to receive the commands from the transmitter and to send them to the Navio2 via the PPM encoder.



Figure 7: Hobbyking HK-T4A transmitter and HK-TR6A V2 receiver

## 2.2 Software

### 2.2.1 OS

The Raspberry Pi 4 runs on the latest Emlid Raspberry Pi OS Buster image. It is a Debian-based OS that is optimized for the Raspberry Pi. The Emlid Raspberry Pi OS Buster image is mandatory in order to be compatible with the Navio2.

### 2.2.2 Python

The control algorithms and the drivers are written in Python 3.11. Python is a high-level programming language that is easy to learn and to use. It is also very popular and has a large community. It is the recommended language for the Navio2. The python library used in the project are listed in the Table 1. Navio2 provides a Python library to control the Navio2. It is used to read the data from the sensors and to send the commands to the servomotors.



| Library                           | Version | Description               |
|-----------------------------------|---------|---------------------------|
| <a href="#">numpy</a>             | 1.25.1  | Scientific computing      |
| <a href="#">scipy</a>             | 1.11.1  | Scientific computing      |
| <a href="#">matplotlib</a>        | 3.7.2   | Plotting library          |
| <a href="#">tkinter</a>           | 8.6     | GUI toolkit               |
| <a href="#">pygubu</a>            | 0.31    | GUI builder               |
| <a href="#">PyYAML</a>            | 6.0.1   | YAML parser               |
| <a href="#">pyproj</a>            | 3.6.0   | Cartographic projection   |
| <a href="#">calypsoanemometer</a> | 0.6.0   | Calypso Anemometer driver |
| <a href="#">Navio2</a>            | 1.0.0   | Navio2 driver             |

Table 1: Python Libraries

## 3 Drivers

### 3.1 Modularity

The drivers are the modules that read the data from the sensors and send them to the control algorithms. The sensors used are the GNSS, the IMU and the anemometer. The actuators are the servomotors. The drivers are all implemented in a modular way. Each driver is a class that can be used with other robots. For example, the servomotors driver allows for any number of servomotors to be used with any configuration for each of them. The only specific driver is the sailboat driver. It is a class that uses the other drivers to control the sailboat. The drivers are implemented in the `Drivers` package.

### 3.2 Calypso Anemometer

The Calypso anemometer like the one in the Figure 8 is a sensor that measures the wind speed and direction [14]. It is solar powered and communicates with the sailboat using Bluetooth. It is a very useful sensor because it is very precise and it is wireless. However, it has some issues. The first issue is that it is not very reliable. Sometimes, it stops sending data for no apparent reason. The second issue is that it is not very robust. It is not waterproof and it is not very resistant to shocks. The third issue is that it is not very easy to use. The Bluetooth connection is not very stable and establishing connection can prove to be difficult. Most of the issues were due to the battery. It only lasted for a few hours and it was hard to recharge it because the solar panel was not very efficient and needed a lot of light to charge.



Figure 8: Calypso anemometer

### 3.3 Multithreading

The drivers are all implemented using multithreading. Each driver is a [13] thread that runs in parallel with the other drivers. This allows for the drivers to communicate with each other without blocking the execution of the other drivers. For example, the GNSS driver sends the position to the sailboat driver. At the same time, the IMU driver sends the orientation to the sailboat driver. The sailboat driver then computes the apparent wind and the command to send to the servomotors. The servomotors driver then sends the command to the servomotors. This is all done in parallel to allow for the fastest possible execution as shown in Figure 9.

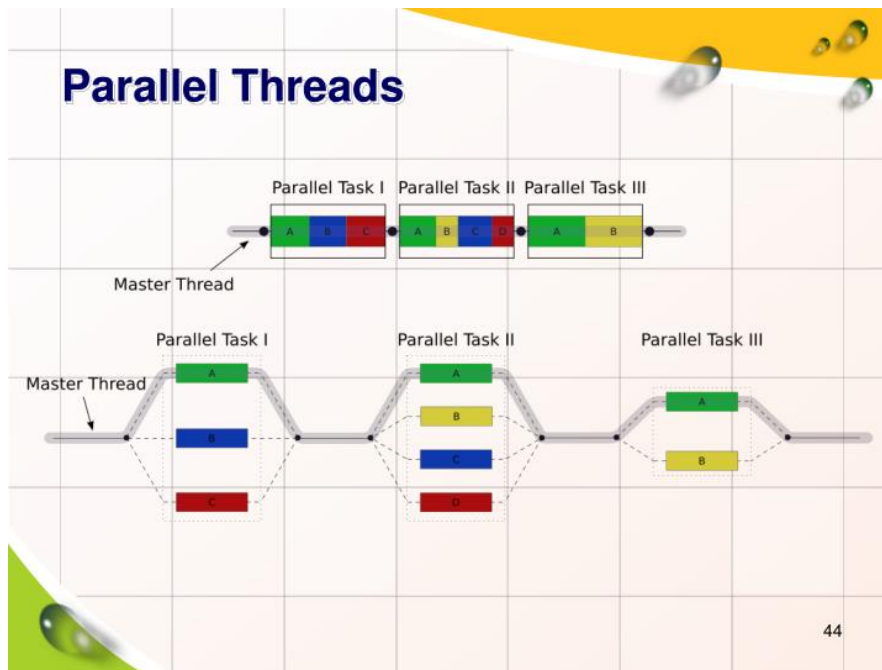


Figure 9: Parallel Threads tasks management

### 3.4 Package

The drivers are implemented in the `Drivers` package. This is made possible using the `__init__.py` file. This file is used to tell Python that the directory is a package. It is also used to import the drivers in the `__init__.py` file of the [1] package. This allows for the drivers to be imported using the following syntax: `from Drivers import *`. This is very useful because it allows for the drivers to be imported in the main file without having to specify the name of the package. This makes the code more readable and easier to use. It also removes the need to add the drivers directory to the `PYTHONPATH` environment variable. The `__init__.py` file of the `Drivers` package is shown in the following Python code.

Listing 1: Drivers Package `__init__.py` File

---

```
from platform import machine
if machine() in ('arm61', 'arm71', 'armv71', 'aarch64'):
    from .Boats import Sailboat
    from .Gnss import *
    from .Imu import *
    from .Servomotors import *
from .Calypso import *
from .Manual import *
from .Boats import VirtualSailboat
from .Virtual import *
```

---

### 3.5 Kalman Filter

The IMU driver uses a Kalman filter to fuse the data from the sensors and to compute the orientation of the sailboat. The Kalman filter is a mathematical algorithm that uses a series of measurements observed over time, containing statistical noise and other inaccuracies, and produces estimates of unknown variables that tend to be more precise than those based on a single measurement alone. The Kalman filter is implemented in the **Observation** and the **Supervision** class. The **Observation** class is used to compute the observation matrix and the observation vector. It updates the estimate of the state of the system given the current measurement and the previous estimate. The **Supervision** class is used to compute the supervision matrix and the supervision vector. Supervisors are the highest level form of control the program has over the boat. They define the mission to accomplish and set abstract instrumental targets to achieve it.

### 3.6 Sailboat Class

The sailboat driver is a class that uses the other drivers to control the sailboat. It is the only specific driver to the sailboat. It contains class attributes that are used to configure the sailboat and class variables that are used to store the data from the sensors for an easy access. It also contains the projection method that is used to project the position of the sailboat on a map, a method to send the command to the servomotors and methods that are used to start and stop properly all the sensors and the sailboat.

## 4 Control Algorithms

### 4.1 Apparent Wind

The apparent wind is the wind experienced by an observer in motion and is the relative velocity of the wind in relation to the observer. It is a combination of the true wind and the effective wind created by our motion. If our speed is zero, the apparent wind is the same as the true wind. It can be described by the apparent wind speed and the apparent wind angle.

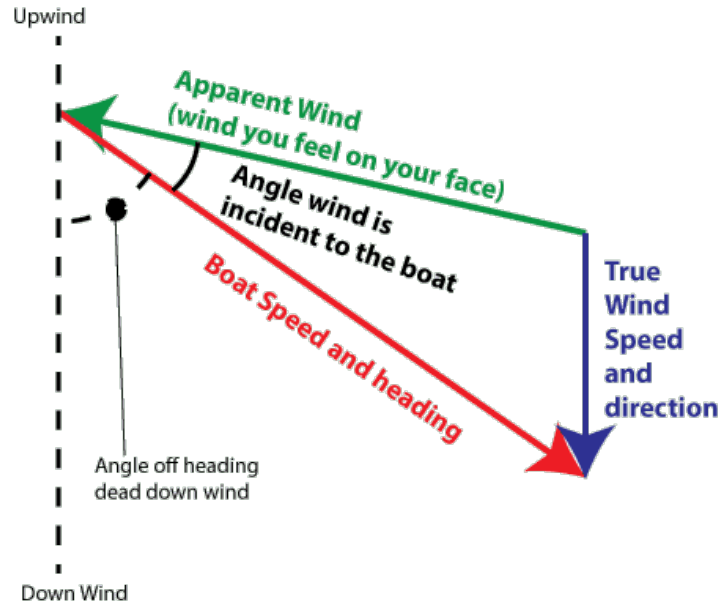


Figure 10: True wind and apparent wind

The Calypso anemometer measures the apparent wind speed and direction. The difference between the apparent wind and the true wind is the speed of the sailboat. It can be neglected or boat speeds less than 10 or 20 percent of the true wind speed. In our case, the sailboat speed is higher than 20 percent of the true wind speed. The true wind speed and direction can be computed using the following equations: [2]

$$V_{tw} = \sqrt{(V_{aw} * \cos(A))^2 + (V_{aw} * \sin(A) + V_s)^2} \quad (1)$$

$$\theta_{tw} = \arctan 2(V_{aw} * \sin(A) + V_s, V_{aw} * \cos(A)) \quad (2)$$

With :

- $V_{aw}$  the apparent wind speed
- $A$  the apparent wind angle

- $V_s$  the sailboat speed
- $V_{tw}$  the true wind speed
- $\theta_{tw}$  the true wind direction

## 4.2 Tacking

Tacking is a sailing maneuver by which a sailing vessel, whose desired course is into the wind, turns its bow toward the wind so that the direction from which the wind blows changes from one side to the other, allowing progress in the desired direction. Here arise various more or less problematic situations which are described on the Figure 11.

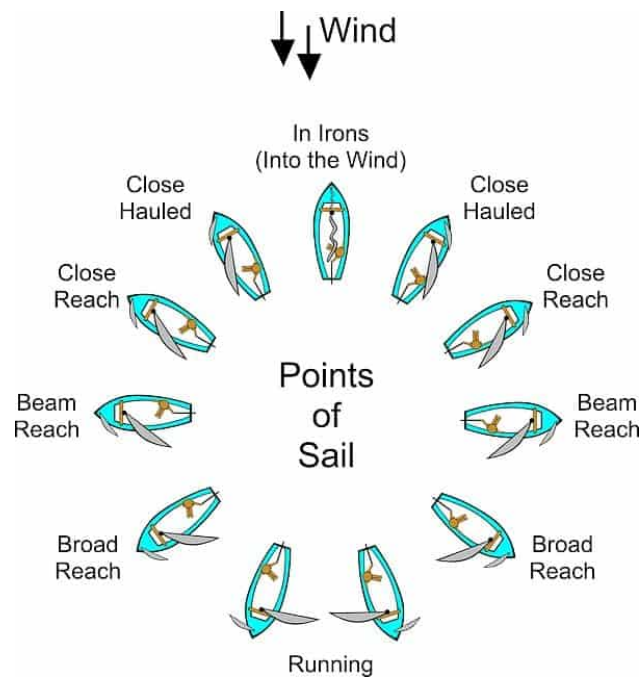


Figure 11: Points of sail

The sailboat can sail on all points of sail except in the no-go zone. The no-go zone is the area where the sailboat cannot sail because the wind is blowing from the wrong direction. When the sailboat is in the Running situation, the sail acts like a parachute and the sailboat is pushed by the wind. When the sailboat is in the Close Hauled, Close Reach, Beam Reach or Broad Reach situation, the wind keeps on pushing the sailboat if the sail is correctly trimmed. When the sailboat is in the Close Hauled situation, the sailboat is sailing as close to the wind as possible.

The sailboat cannot sail directly into the wind. It has to sail on a zigzag course to reach its destination. This is called tacking. The sailboat is sailing on a zigzag course because it cannot sail directly into the wind. It has to sail on a course that is between

45 and 60 degrees from the wind until the sailboat is too far from the desired line to follow and then it has to tack to the other side as shown in Figure 12.

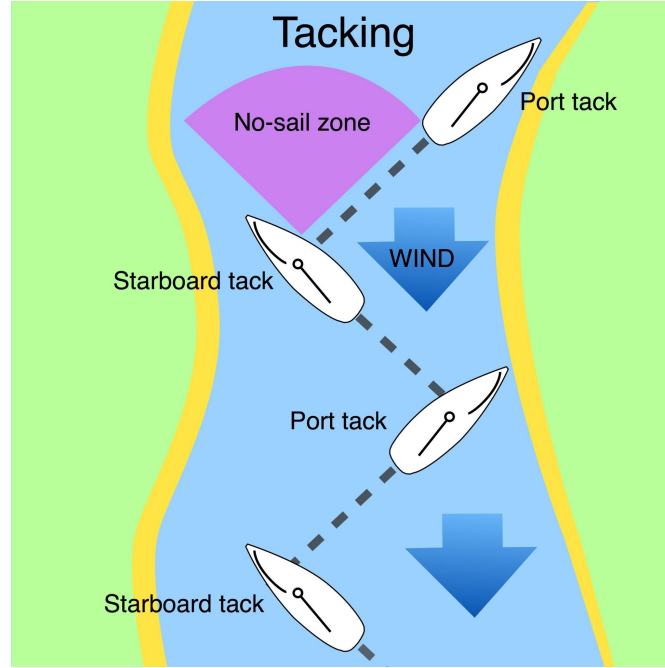


Figure 12: Tacking

### 4.3 Line Following

Line following is a control algorithm that allows the sailboat to follow a line. It is used to follow a line on a map. It is a very simple algorithm that is based on the difference between the current position and the desired position. The algorithm implemented in the sailboat is the one described in the paper *A Simple Controller for Line Following of Sailboats* by LUC JAULIN and FABRICE LE BARS [17]. This algorithm was chosen because it is very simple and it is very efficient. Its low complexity allows for an application on a Raspberry Pi 4 with very limited computation power. It uses a vector field to compute the best route to follow the line and returns a sail angle and a rudder angle to follow the line.

### 4.4 Station Keeping

Station keeping is a control algorithm that allows the sailboat to stay in a given area. It has four main steps. The first step is to compute the desired position and reach it using a line following algorithm. The second step is to turn around the desired position to enter it facing the wind. The third step is a tack strategy to move upwind. Finally, the fourth step is to enter the desired position and orient the boat front to the wind so it moves as little as possible even when pushed by the current. The algorithm implemented in the the sailboat is the one described in the paper *Position keeping control of an autonomous sailboat* by CHRISTOPHE VIEL, ULYSSE VAUTIER, JIAN WAN and LUC JAULIN [19].

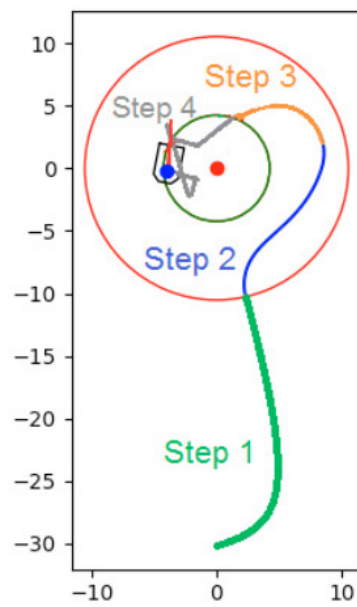


Figure 13: Steps of orientation control.



## 5 Simulation

### 5.1 Model

The sailboat model is a model that is used to simulate the sailboat. It has parameters that can be changed to simulate different sailboats. The goal is to have a model that is as close as possible to the real sailboat in order to accurately test our control algorithms. The model used in our simulation is inspired by the model described in the paper *Modeling, control and state-estimation for an autonomous sailboat* by JON MELIN [18]. The model is given by non-linear differential equations in state space with five states,  $\mathbf{x} = [x \ y \ \theta \ v \ \omega]^T$

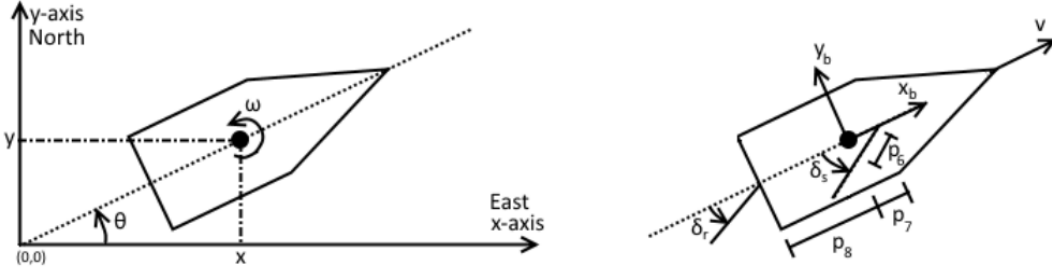


Figure 14: Left: Position  $(x, y)$  and orientation  $(\theta)$  of the sailboat is given in a North-East-Up reference frame. Right: Velocity  $(v)$  and angle of rudder  $(\delta_r)$  and the angle of the sail  $(\delta_s)$  is given in a sailboat fixed reference system together with distances  $(p_6, p_7$  and  $p_8)$ . Source [18].

The state equation is used to compute the state of the sailboat at the next time step. It is computed using the current state of the sailboat and the current command. The state equation is described by the following compact equation,

$$\dot{\mathbf{x}} = f(\mathbf{x}, u, W_{p,tw}) \quad (3)$$

where  $u = [\delta_s \ \delta_r]$  is the command vector and  $W_{p,tw} = [a_{tw} \ \psi_{tw}]$  is the true wind vector. The complete equation is given by,

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \\ \dot{v} \\ \dot{\omega} \end{bmatrix} = \begin{bmatrix} v \cos(\theta) + p_1 a_{tw} \cos(\psi_{tw}) \\ v \sin(\theta) + p_1 a_{tw} \sin(\psi_{tw}) \\ \omega \\ (g_s \sin(\delta_s) - g_r p_{11} \sin(\delta_r) - p_2 v^2) / p_9 \\ (g_s (p_6 - p_7 \cos(\delta_s)) - g_r p_8 \cos(\delta_r) - p_3 v \omega) / p_{10} \end{bmatrix} \quad (4)$$

All the parameters are given in the following Table 2.

### 5.2 Configuration Files

The configuration file is a file that is used to configure the simulation. In our case, two configuration files are used. The first one is used to configure the sailboat

| Parameter                  | Value                             | Parameter                    | Value                    |
|----------------------------|-----------------------------------|------------------------------|--------------------------|
| $p_1$ —                    | Drift coefficient                 | $p_7$ [m]                    | Distance to mast         |
| $p_2$ [kgs <sup>-1</sup> ] | Tangential friction               | $p_8$ [m]                    | Distance to rudder       |
| $p_3$ [kgm]                | Angular friction                  | $p_9$ [kg]                   | Mass of boat             |
| $p_4$ [kgs <sup>-1</sup> ] | Sail lift                         | $p_{10}$ [kgm <sup>2</sup> ] | Moment of inertia        |
| $p_5$ [kgs <sup>-1</sup> ] | Rudder lift                       | $p_{11}$ —                   | Rudder break coefficient |
| $p_6$ [m]                  | Distance to sail center of effort |                              |                          |

Table 2: Model parameters

parameters. Inside, we can find the parameters of the model as described in the Table 2, the parameters used in the control algorithms such as the cutoff distance for the line following algorithm or the inner and outer radius of the station keeping target.

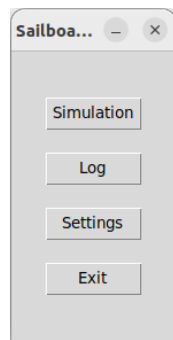
The second configuration file is used to configure the simulation. It contains the parameters of the simulation such as the parameters of the random wind direction and speed generator, the noise added to the sensors, the initial state vector of the sailboat, the duration of the simulation and the time step of the simulation.

We choose to use a YAML file to configure the simulation. YAML is a human-readable data serialization language. It is commonly used for configuration files and in applications where data is being stored or transmitted. It is easily understandable by humans and quickly modified without the need to modify the code of the simulation. It is also very easy to use in Python because there is a YAML package that allows to read and write YAML files.

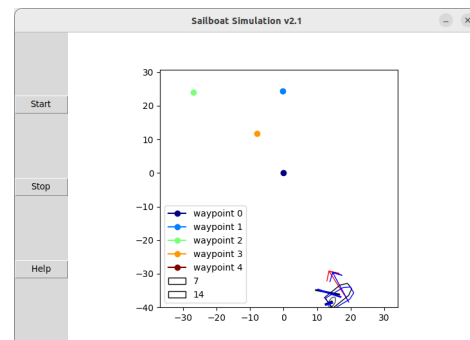
### 5.3 Graphical User Interface

It was not necessary to implement a graphical user interface for the simulation but it was important for us to have a human-friendly interface that anyone could use easily the first time. That is why we choose to implement on top of our already working command line interface a graphical user interface. It is implemented using the `Pygubu` package. It is a Python package that allows to create graphical user interfaces using the `Tkinter` package. It is very easy to use and it allows to create small windows with buttons and labels very quickly [15].

The graphical user interface is composed of two important windows. The first one is the main window. It is the window that is displayed when the simulation is launched. It contains a button to launch the simulation, a button to replay a mission by selecting a log file, a button to change the configuration file and a button to quit the simulation. Each button opens a new subwindow. The second important window is the simulation window. It is the window that is displayed when the simulation is running. It contains a button to start the simulation, a button to pause the simulation and a button to stop the simulation. The main window is shown in the Figure 15a. The other important window is the simulation window. It contains a map that is updated in real time with the position of the sailboat, the desired target to reach, the sailboat with its sail angle, rudder angle,



(a) Main window



(b) Simulation window

Figure 15: Graphical user interface.

heading and speed and finally the true wind vector. The simulation window is shown in the Figure 15b.

You can find the video of a simulation of the line following algorithm is shown in the following [video](#) or at the following url [https://youtu.be/m\\_ThnV1PEVU](https://youtu.be/m_ThnV1PEVU).

## 6 Test and Results

### 6.1 Test Location

The tests were conducted in the lake near the **Bournville Radio Sailing and Model Boat Club** south of Birmingham. It is a very large and shallow lake with a lot of space to test the sailboat. It is also a very quiet place with no current and enough wind for the small sailboat. The lake is shown in the Figure 16.

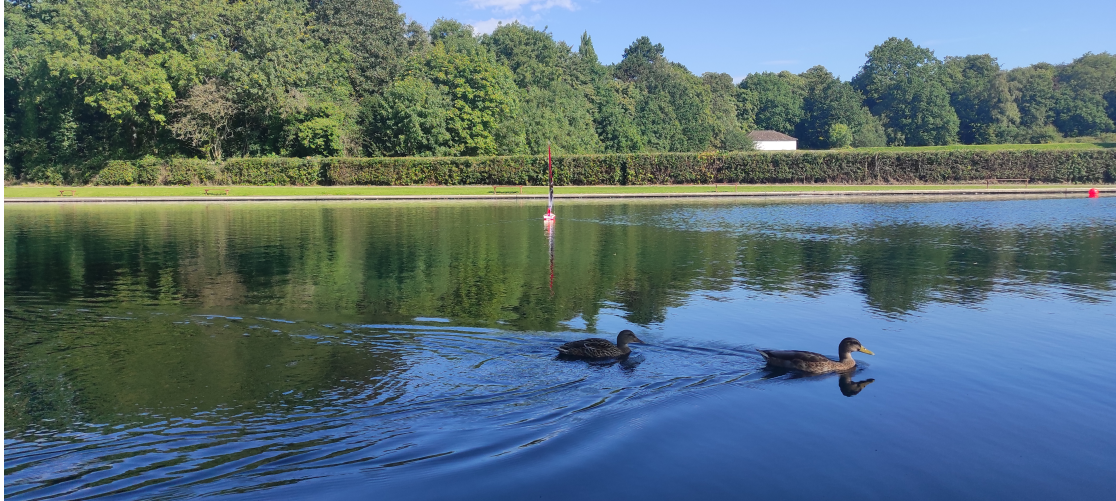


Figure 16: Bournville artificial lake.

The lake is only one meter deep and is used everyday by the member of the **Bournville Radio Sailing and Model Boat Club** to test their small scale models. That makes it a perfect place to test our small sailboat. We manage to test the sailboat in the lake for a few days. We were able to test the line following algorithm and the station keeping algorithm. We were also able to test the sailboat in different wind conditions like a very light wind and in a very strong wind.

### 6.2 Line Following Log Analysis

This test was made the morning of the 03rd of August 2023. We tried the line following algorithm in a very light wind. The wind was blowing from the right side of the sailboat. It was the second time we were going to the lake. The first time, the weather was not good enough to test the sailboat. The log replay can be found in the following [video](#) or at the following url <https://youtu.be/OuOgLJFMaGA>.

The replay is accelerated by a factor of 3. At the beginning, the sailboat was able to reach the line and to follow it for around 8 meters. Then, we can see the wind vector changing his orientation quickly. The sailboat then starts drifting away from the line. When it tries to correct its trajectory, the wind vector keeps on changing again and the sailboat correct to much its trajectory. It then starts spinning on itself so we stopped the program.

The problem was that the wind vector was changing too quickly. The Calypso was already filtered but the filter was not good enough. Furthermore, we found out that the IMU had a lot more noise when we were on the lake than when we were in the lab. Both issue combined made the true wind vector impossible to correctly compute. We then decide to come back to the lab and change all the filters.

### 6.3 Station Keeping Log Analysis

This test was made the morning of the 08th of August 2023. We tried the station keeping algorithm with a light and changing wind. During the previous days, we had implemented a new filter for the IMU and the Calypso. We also corrected a bug in the GNSS driver. The log replay can be found in the following [video](https://youtu.be/n7GqNG0Jt0c) or at the following url <https://youtu.be/n7GqNG0Jt0c>.

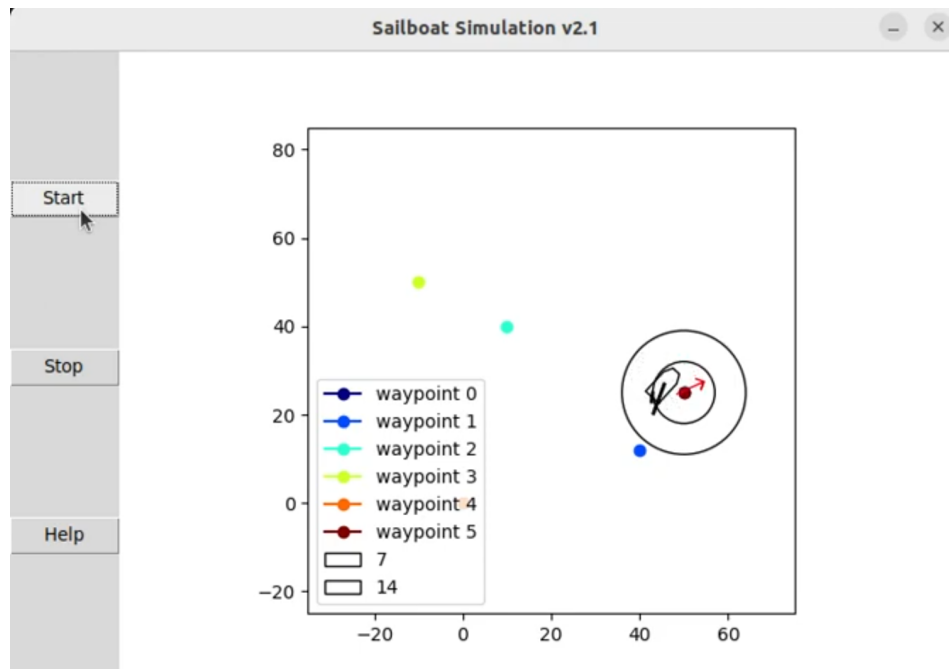


Figure 17: 08th August 2023 Log Replay Image.

The replay is in real time. At the beginning, the sailboat tries to reach the target. The orientation of the wind vector is changing but mostly blows in the best direction. During the line following, the sailboat sometimes loops on itself but, in general, it keeps going in the same direction. When the sailboat reaches the target, it starts to turn around it to reach inside facing the wind. It the reaches inside the target 17 but directly drifts away. It then tries to correct its trajectory but the wind vector changes again and the sailboat starts spinning on itself while going accross the target. At this moment, we stopped the program and brought back the sailboat using the radio control.

The problem was that when we started to use both servomotors, the battery was not powerful enough to power the sailboat entirely. The Raspberry Pi was the last one to be powered and it slowed it down or often shutted it down. We needed to use two batteries at the same time. One for the servomotors and one for the Raspberry Pi. Another issue was that the Navio2 HAT with the IMU was placed on top of the winch servomotor. The winch servomotor was creating a lot of induced current that was interfering with the IMU. In order to solve this issue, it would have been necessary to change the hardware and to rebuild entirely what was inside the sailboat. We only had a few weeks left so we choosed not to do it and to keep on going with this issue.

## Conclusion

This internship was a very interesting experience. I learned a lot about the hardware and the software of a small autonomous sailboat. I also learned a lot about the control algorithms used to control it. I learned how to properly integrate the sensors and create my own modular drivers. I also learned how to create a graphical user interface to make the simulation more user-friendly. Sadly, the test results were not as good as we expected. The sailboat was not able reproduce the same results as in the simulation and did not have the same results for the same algorithm when we tested it in the lake.

I think that the main issues could be solved by changing the position of the hardware inside the boat. The IMU should be placed as far as possible from the servomotors and the battery. The battery should also be placed as low as possible to lower the center of gravity of the sailboat and fixed to the hull to avoid any movement. An important part of the last week was dedicated to writing the documentation of the code so it could be easily used by the next interns. We also wrote README files for the GitLab to explain how to use the code, launch and use simulation and launch all the programs.

All the work produced during this internship is available on the following [GitLab](https://gitlab.ensta-bretagne.fr/rangarha/aston-autonomous-sailboat-2023.git) or at the following url <https://gitlab.ensta-bretagne.fr/rangarha/aston-autonomous-sailboat-2023.git>.

## List of Acronyms

**ENSTA Bretagne** École Nationale Supérieure des Techniques Avancées Bretagne

**IOM** International One Metre

**HAT** Hardware Attached on Top

**LiPo** Lithium Polymer

**GNSS** Global Navigation Satellite System

**IMU** Inertial Measurement Unit

**PPM** Pulse Position Modulation

**PWM** Pulse Width Modulation

**RC** Radio Control

**GPIO** General Purpose Input/Output

**MCX** Micro Coaxial

**OS** Operating System

**ESC** Electronic Speed Controller

**YAML** YAML Ain't Markup Language



## List of Figures

|    |   |    |
|----|---|----|
| 1  | Aston University, Birmingham, England. . . . .  | 5  |
| 2  | Louis-Philippe Crépin, <i>Le Redoutable à Trafalgar</i> , 1807, Musée national de la Marine. <i>The Redoutable</i> at the battle of Trafalgar, between the <i>Victory</i> on the left and the <i>Temeraire</i> on the right. . . . .  | 6  |
| 3  | Shipping density (commercial). A Global Map of Human Impacts to Marine Ecosystems, showing relative density (in color) against a black background. Scale: 1 km. More details available at: <a href="http://spatial-analyst.net/worldmaps/shipping.rdc">http://spatial-analyst.net/worldmaps/shipping.rdc</a> . . . . .                          | 7  |
| 4  | Ragazza 1 Meter Sailboat V2: RTR . . . . .  | 8  |
| 5  | Ragazza 1 Meter Sailboat V2: RTR with his four main parts . . . . .   | 10 |
| 6  | Navio2 HAT on top of a Raspberry Pi 4 . . . . .   | 11 |
| 7  | Hobbyking HK-T4A transmitter and HK-TR6A V2 receiver . . . . .  | 12 |
| 8  | Calypso anemometer . . . . .  | 14 |
| 9  | Parallel Threads tasks management . . . . .   | 15 |
| 10 | True wind and apparent wind . . . . .   | 17 |
| 11 | Points of sail . . . . .  | 18 |
| 12 | Tacking . . . . .   | 19 |
| 13 | Steps of orientation control. . . . .   | 20 |
| 14 | Left: Position ( $x, y$ ) and orientation ( $\theta$ ) of the sailboat is given in a North-East-Up reference frame. Right: Velocity ( $v$ ) and angle of rudder ( $\delta_r$ ) and the angle of the sail ( $\delta_s$ ) is given in a sailboat fixed reference system together with distances ( $p_6$ , $p_7$ and $p_8$ ). Source [18]. . . . . | 21 |
| 15 | Graphical user interface. . . . .   | 23 |
| 16 | Bournville artificial lake. . . . .   | 24 |
| 17 | 08th August 2023 Log Replay Image. . . . .  | 25 |

## List of Tables

|   |                            |    |
|---|----------------------------|----|
| 1 | Python Libraries . . . . . | 13 |
| 2 | Model parameters . . . . . | 22 |

## References

- [1] 6. Modules.
- [2] Apparent Wind vs. True Wind.
- [3] Aston University UK | Ranking, Courses and Scholarships.
- [4] Hardware setup | Navio2.
- [5] HobbyKing 2.4Ghz Mode 2 4Ch Tx & Rx V2.
- [6] Introduction | Navio2.
- [7] Navio2.
- [8] Pro Boat Ragazza 1 Meter Sailboat V2: RTR [PRB07003] - AMain Hobbies.
- [9] Ragazza 1 Meter Sailboat V2: RTR (PRB07003) | Astra.
- [10] Ragazza 1 Metre Sailboat RTR-Gliders Distribution.
- [11] Raspberry Pi configuration | Navio2.
- [12] Shipping and world trade: driving prosperity.
- [13] threading — Thread-based parallelism.
- [14] Ultrasonic Portable wind meter.
- [15] Alejandro Autalán. Welcome to Pygubu!, September 2023. original-date: 2013-02-13T00:02:24Z.
- [16] Matt C. Basics of Sailboat Hull Design - EXPLAINED For Owners - My Cruiser Life Magazine, July 2022. Section: Boat Parts.
- [17] Luc Jaulin and Fabrice Le Bars. A Simple Controller for Line Following of Sailboats. In Colin Sauzé and James Finnis, editors, *Robotic Sailing 2012*, pages 117–129. Springer Berlin Heidelberg, Berlin, Heidelberg, 2013.
- [18] Jon Melin, Kjell Dahl, and Matias Waller. Modeling and Control for an Autonomous Sailboat: A Case Study. In Anna Friebe and Florian Haug, editors, *Robotic Sailing 2015*, pages 137–149, Cham, 2016. Springer International Publishing.
- [19] Christophe Viel, Ulysse Vautier, Jian Wan, and Luc Jaulin. Position keeping control of an autonomous sailboat. *IFAC-PapersOnLine*, 51(29):14–19, 2018.

# Assessment Report



## RAPPORT D'EVALUATION ASSESSMENT REPORT

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

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

Nom du superviseur / Name of internship supervisor  
Jian Wan

Fonction / Function Lecturer in Mechatronics and Robotics

Adresse e-mail / E-mail address wanj3@aston.ac.uk

Nom du stagiaire accueilli / Name of intern Ludovic Mustiere

### 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* can be more proactive to solve any issue that occurs

#### **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* time management skills can be improved

#### **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 Birmingham, on 05/09/2023

Signature Entreprise  Signature stagiaire  
Company stamp \_\_\_\_\_ Intern's signature

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

## Config Files

### Boatconfig.yaml

```
1 # Sailboat simulation configuration file
2 P0: 0.03 # Drift coefficient (defaults to 0.03)
3 P1: 40 # Tangential friction (defaults to 40)
4 P2: 6000 # Angular friction (defaults to 6000)
5 P3: 200 # Sail lift (defaults to 200)
6 P4: 1500 # Rudder lift (defaults to 1500)
7 P5: 0.5 # Distance to sail center of effort
8 # (defaults to 0.5)
9 P6: 0.5 # Distance to mast (defaults to 0.5)
10 P7: 2 # Distance to rudder (defaults to 2)
11 P8: 300 # Mass of the boat (defaults to 300)
12 P9: 400 # Moment of inertia (defaults to 400)
13 P10: 0.2 # Rudder break coefficient (defaults to 0.2)
14 DELTA_RMAX: 0.6283185307179586 # Maximum angle for the rudder
15 # (defaults to pi/5)
16 BETA: 0.3 # Angle of the sail in crosswind
17 # (defaults to 0.3)
18 R: 40 # Cutoff distance (defaults to 40)
19 INNER_RADIUS: 7 # Inner radius of the station keeping target
20 # (defaults to 7)
21 OUTER_RADIUS: 14 # Outer radius of the station keeping target
22 # (defaults to 14)
23 XSI: 1.0471975511965976 # Close-hauled angle (defaults to pi/3)
24 GAMMA : 0.7853981633974483 # Incidence angle (defaults to pi/4)
```

### Simconfig.yaml

```
1 # Simulation configuration file
2 SEED_SPAWN: 42 # Integer used to spawn the seed used for
3 # random number generation (defaults to 42)
4 STATIC: False # Whether the wind is static (defaults to False)
5 AVG_DUR: 40 # Average duration for which the wind
6 # vector should remain sensibly the same (defaults to 40 s)
7 STD_DUR: 10 # Standard deviation of the characteristic
8 # wind vector evolution duration (defaults to 10 s)
9 AVG_WIND_NORM: 2 # Average dynamic wind force or static wind
10 # force (defaults to 2)
11 STD_WIND_NORM: 1 # Standard deviation of the wind force
12 # (defaults to 1 m/s)
13 WIND_NORM_INTERP: 'cubic' # Interpolation method for the norm of the wind
14 # vector (defaults to "cubic")
15 PSI: 2.356194490192345 # Static wind orientation (defaults to 3*pi/4)
16 STD_PSI: 1.0472 # Standard deviation of the wind orientation
17 # (defaults to pi/3 rad)
18 PSI_INTERP: 'cubic' # Interpolation method for the orientation of
19 # the wind vector (defaults to "cubic")
20 BOAT_CONFIG: 'boatConfig' # Name of the boat configuration file without
```

```

21  # the extension
22  AVG_IMU: 0 # Average noise for the IMU
23  STD_IMU: 0.3 # Standard deviation for the IMU
24  AVG_GNSS_POS: 0 # Average noise for the position read by the
25  # GNSS
26  STD_GNSS_POS: 0.00001 # Standard deviation for the position read
27  # by the GNSS
28  AVG_GNSS_SPEED: 0 # Average noise for the speed read by the GNSS
29  STD_GNSS_SPEED: 0.1 # Standard deviation for the speed read by
30  # the GNSS
31  AVG_CALYPSO_WIND: 0 # Average noise for the wind data read by the
32  # Calypso
33  STD_CALYPSO_WIND: 0.2 # Standard deviation for the wind data read
34  # by the Calypso
35  AVG_CALYPSO_TEMPERATURE: 0 # Average noise for the temperature data read
36  # by the Calypso
37  STD_CALYPSO_TEMPERATURE: 1.5 # Standard deviation for the temperature data
38  # read by the Calypso
39  AVG_CALYPSO_POS: 0 # Average noise for the position data read by
40  # the Calypso
41  STD_CALYPSO_POS: 0.3 # Standard deviation for the position data
42  # read by the Calypso
43  X: 10 # 1st coordinate of the position, 1st
44  # component of the state vector
45  Y: -40 # 2nd coordinate of the position, 2nd
46  # component of the state vector
47  THETA: 0 # Heading, 3rd component of the state vector
48  V: 0 # Speed, 4th component of the state vector
49  W: 0 # Speed rotation, 5th component of the state
50  # vector
51  DELTA_RO: 0 # Angle for the rudder
52  DELTA_SO: 0 # Angle for the sail
53  T_MAX: 300 # Maximal duration for the simulation
54  MODEL_DT: 0.05 # Time interval used to advance the simulation
55  # of the boat model from one iteration to the next
56  ALGO_DT: 0.1 # Time step used in the simulation loop to
57  # control the rate at which the simulation runs
58  TIME: 0 # Initial time of the simulation

```