Autonomous Sailboat

Aston University
BIRMINGHAM UK

Promotion 2023

October 16, 2022

Augustin Morge
augustin.morge@ensta-bretagne.org

I express my deepest thanks to Dr. Jian Wan *Lecturer at Aston University for the opportunity of internship, his availability and for the help all along the internship. I would also like to thank* Mr Jaulin *for introducing me to this internship and to the Dr. Wan.*

Contents

Abstract

The main purpose of this internship was to develop a sailboat with a new set of sensors. This sailboat should be able to follow a line or stay at a point with coordinates. Thanks to Aston University, situated in Birmingham, I was able to configure this system and test it in real conditions.

Several tests were done during the internship and the following document highlights how to build, program and communicate with an autonomous sailboat. Firstly, I needed to get used to the different sensors, to get the data and filter them. Once the sensor data has been acquired, the program collected them into a state vector and a controller computed them in order to give an angle for the sail and the rudder. All in all, those angle were converted into a PWM signal for the actuators.

However, it was also important to see what was happening in real time or replay the experiments. This is why I set up a communication system between the Raspberry Pi and the computer to check what was happening. The 3D printing was also a part of the internship to put the system inside the hull. Indeed, the system was new for both hardware and software with a Raspberry Pi using a Navio2 device and a Calypso Ultrasonic Sensor for the wind sensor.

Keywords

Guidance, control, sailing boat, filtering, 3D printing, 2D mapping.

Chapter 1

Introduction

1.1 The University of Aston

The internship takes place at Aston University in Birmingham, UK.

Birmingham is a once industrial city and England's second most populous city. The city is located in the middle west of England. [Fig.1.1][Wik22a]

Figure 1.1: Birmingham in England

Aston University is a public research university situated in the city centre of Birmingham, England. In September 2021, it was announced that Aston University was shortlisted for University of the Year in the Times Higher Education Awards 2021. In 2020, the University was named "University of the Year" by The Guardian.[Wik22b]

1.2 Why autonomous sailboat ?

Marine robotics is one of the less developed field in robotics unlike the ground robotics (autonomous cars, humanoid robots, ...) or the aerial robotics (with autonomous plane, drones, ...). This is due to the global interest of the population in space exploration or use of robotics in their

daily life.
Considering that water covers about 71% of the Earth's surface and the majority of the ocean surface is yet to be explored or mapped, marine robotics has attracted more and more interest from both industry and academy. The maritime robotics can be used for hydrographers and oceanographers in their research, for the military marine defense or global trade. Even championship are now developed with the World Robotic Sailing Championship (WRSC) or the International Robotic Sailing Conference (IRSC).

The autonomous sailboat is an ecological system with a low-cost design that could be used in various context such as studies, researches or transport.

1.3 Purpose of the internship

The purpose of the internship is developed into three axis:

1. To familiarize with various sensors such as IMU, weather vane, camera, GPS receiver, Li-DAR and others for sensor data collection, filtering and fusion
2. To use Arduino/Raspberry Pi/NVIDIA boards to program control and learning algorithms through C++, Python and/or ROS[1]
3. To use some basic mechanical design and 3D print of sensor supporting mechanisms for the autonomous systems.

I started the project with a previous sailboat with all sensors controlled by an Arduino except the GPS which was controlled by a Raspberry. This system was based on ROS.
The aim of my internship was to build a new sailboat using only the raspberry pi with a Navio2[2] on it to see which system was the most efficient.

Thus, I worked on the Raspberry Pi system[3] which was useful here because we can connect the wind sensor in Bluetooth and because it's also more compact: other devices like the IMU or GNSS are directly on the Navio2. The Navio2 is a board to plug on the Raspberry Pi. With this board we can use its set sensors or Ardupilot module to control easily and remotely our system with application like Mission Planner or QGroundControl.

In my case, I use the GNSS receiver and the IMU from the Navio2 and use this device to give to the actuators the right PWM signal. The Raspberry Pi is used to get the wind vector and compute all the data in a main algorithm to control the sailboat. Thus, I had to acquire the data the sensors, filter them and build a controller in order to control the boat with the actuators to create a new set [Fig. 7.1].

1.4 Notations

The purpose of the internship is to control the two servomotors. One is for the sail with δ_s and the other one for the rudder with δ_r.

The parameters used by the sensors to control the actuators are :

[1] Robot Operating System
[2] Autopilot HAT for Raspberry Pi
[3] The Raspberry Pi is a Raspberry Pi 4.

$\mathbf{m}$	position of the sailboat
θ	orientation of the sailboat
v	velocity of the sailboat
ϕ	course angle of the sailboat
δ_r	angle of the rudder, $\lvert\delta_r\rvert \leq \delta_r^{\max}$
δ_s	angle of the sail, $\lvert\delta_s\rvert \leq \delta_s^{\max}$
ψ_{tw}, a_{tw}	direction and speed of the true wind
ψ_{aw}, a_{aw}	direction and speed of the apparent wind
$\mathbf{a}, \mathbf{b}$	the starting and the ending points of a specified line
ξ	the close haul angle
r	the cutting distance to a line
β	the angle of the sail in crosswind
q	a binary variable $q \in \{-1, 1\}$ for tacking
d_i	the inner cycle for station keeping
d_o	the outer cycle for station keeping

Table 1.1: Notations

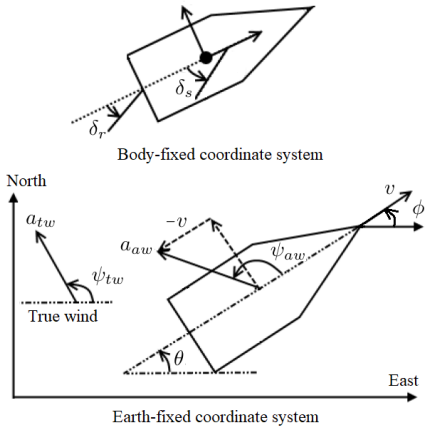

Figure 1.2: The coordinate systems and the true wind

The current GPS position of the boat is m, the direction of the true wind ψ_{tw}, the speed of the true wind a_{tw}, and also the orientation of the boat θ. The velocity of the sailboat is denoted as v and the corresponding course angle for this sailboat speed is denoted as Φ. Due to the sideway forces of the wind, the course angle Φ and the heading angle θ are not necessarily equal. It is worthy to note that m, ψ_{tw}, a_{tw}, θ, and Φ are defined in a East-North-Up (ENU) coordinate system with its origin fixed on an Earth point while δ_r and δ_s are given in a body-fixed coordinate system with its origin fixed on the gravity center of the boat.

Chapter 2

Sensors

In order to acquire the data from the Raspberry Pi I first had to configure it, flashing the raspbian OS on it. Henceforth, I configure the Navio2 with the documentation[Mou08] and check with the library how to acquire the raw data from each sensor.

All the details to configure the Raspberry Pi are on my Gitlab[Aug22].

2.1 IMU

Two IMU sensors are directly integrated on the navio2. [Fig. 2.1]

- MPU9250
- LSM9DS1

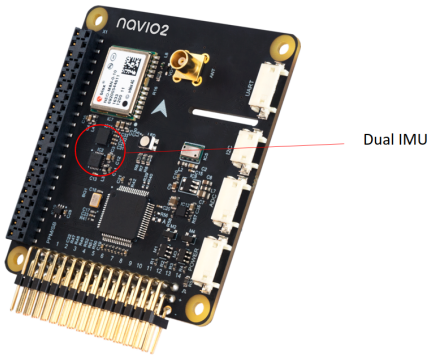

Figure 2.1: Dual IMU on the Navio2

Those sensors collect the data from the gyroscope, the magnetometer and the accelerometer. Those data allow me to know the roll, the pitch and the yaw.

- The accelerometer gives the acceleration vector pointing into the Earth's center. With this vector $\overrightarrow{Acc} = \begin{pmatrix} Acc_x \\ Acc_y \\ Acc_z \end{pmatrix}$ we will obtain the pitch and the roll.

- The magnetometer indicates the position of the magnetic North, by projecting this direction in the space. With this vector we will obtain the yaw. $\overrightarrow{Mag} = \begin{pmatrix} Mag_x \\ Mag_y \\ Mag_z \end{pmatrix}$
- The gyroscope gives information about the angular velocity ω with $\overrightarrow{Gyr} = \begin{pmatrix} Gyr_x \\ Gyr_y \\ Gyr_z \end{pmatrix}$. Indeed, by integrating $\overrightarrow{Gyr}$ we can have the angle.

$$roll = \arctan(\frac{Acc_x}{\sqrt{Acc_y^2 + Acc_z^2}}) \tag{2.1a}$$

$$pitch = \arctan(\frac{Acc_y}{\sqrt{Acc_x^2 + Acc_z^2}}) \tag{2.1b}$$

For the yaw, we need to have the best data while combining the pitch and the roll. We project them in the plan xOy, this way we have less errors when the Raspberry Pi starts to not be in this plan.

We use the roll and picth found in 2.1 into the tilted compensation formula with the raw data from the Magnetometer $\begin{pmatrix} Mag_x \\ Mag_y \\ Mag_z \end{pmatrix}$.

$$\begin{pmatrix} Xh \\ Yh \end{pmatrix} = \begin{pmatrix} Mag_x \\ Mag_y \\ Mag_z \end{pmatrix} \begin{pmatrix} cos(pitch) & 0 & sin(pitch) \\ sin(roll)sin(pitch) & cos(roll) & sin(roll)sin(pitch) \end{pmatrix}$$

The previous formula works because the z axis is not in a direct coordinate system.

Thus,

$$\theta = yaw = \arctan(\frac{Yh}{Xh}) \tag{2.2}$$

The second step was to calibrate the data from Magnetometer. [Fig. 2.2]

To calibrate the IMU, we have to put the system inside the boat with every components : The servomotors or even the battery will disturb the electromagnetic field inside the hull and change the sphere created by the magnetometer vector into an ellipsoid. We suppose that between two experience, the magnetic field won't change.

Thus, to calibrate the magnetometer, we have to turn the sailboat in every direction to obtain a set of points. The vector of the magnetometer will correspond to a point in a 3D space: Mag_x, Mag_y and Mag_z. When I turn the boat in every direction it will create an ellipsoid. This ellipsoid need to be calibrate into a normalized and centred sphere.

Three parameters are used to transform this ellipsoid into a sphere : a rotation matrix R, a transformation matrix T and a normalized matrix N.[cal]
$Mag_{calibrated} = R.T.N.Mag_{raw}$

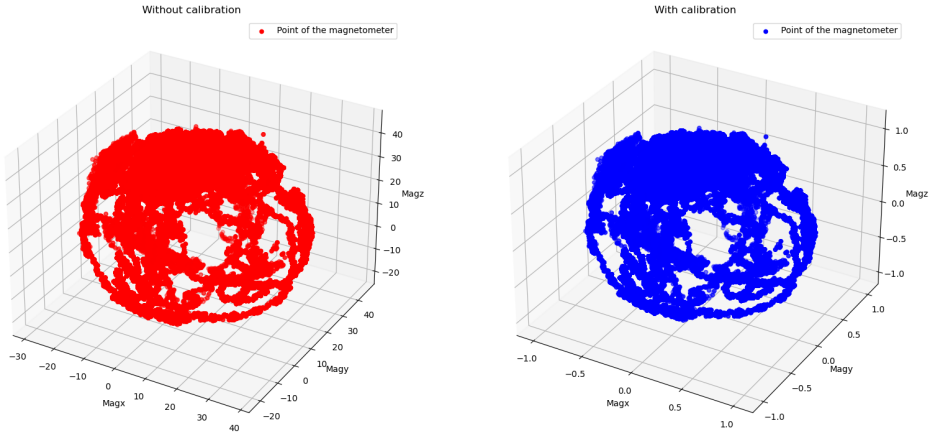

Figure 2.2: Calibration of the LSM IMU

Once it was done I had to implement a Kalman Filter to get better value for the heading data.[Luc15]

Evolution equation :
$x_{k+1} = f(x_k, u_k)$; $f(x_k, u_k) = F_k x_k + v_k$ $v_k = h(\hat{v_k}, u_k)$ - $F_k \hat{x_k}$.
Measurement equation :
$y_k = h(x_k)$; $h(x_k) = H_k x_k$
Prediction step :
$x_{k\|\hat{k-1}} = f(x_{k-\hat{1}\|k-1}, u_k)$ $P_{k\|k-1} = F_k P_{k-1\|k-1} F_k^T + Q_k$
Update step:
$y_k = z_k - h(x_{k\|\hat{k-1}})$ $S_k = H_k P_{k-1\|k-1} H_k^T + R_k$ $K_k = P_{k\|k-1} H_k^T S_k^{-1}$ $x_{\hat{k}\|k} = x_{k\|\hat{k-1}} + K_k y_k$ $P_{k\|k} = (I - K_k H_k) P_{k\|k-1}$
Parameters:
F_k the evolution model; H_k the observation matrix $x_{k\|k-1}$ the predicted estimation; $P_{k\|k-1}$ the predicted covariance $x_{k\|k}$ the corrected estimation; $P_{k\|k}$ the corrected covariance y_k the innovation; S_k the covariance of the innovation K_k the Kalman gain; u_k the control vector Q_k, R_k the covariance matrix for the state and for the measures.

Table 2.1: Kalman filter

Because I wanted to filter the yaw, I chose the input $u_k = (Gyr_z)_k$ with the following variables:

$$z_k = \begin{pmatrix} \cos(yaw_k) \\ \sin(yaw_k) \\ 1 \end{pmatrix}; F_k = F(u_k) = \begin{pmatrix} 1 & -dt.u_k & 0 \\ dt.u_k & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}; H_k(x) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ (x_1)_k & (x_2)_k & 1 \end{pmatrix}$$

I also chose for the initial matrix:

$$P_0 = 10.I_3;\ Q_0 = 0.001.I_3;\ R_0 = 0.01.I_3$$

With u_k the input of the Gyroscope which is the angle speed and x the final value the magnetic vector for the yaw We finally have $yaw = \arctan(x_2/x_1)$. Thus, when I applied the filter for example when the Raspberry Pi is not moving, I could see that the value filtered was closest to the real one instead of the raw data [Fig. 2.3]. Moreover, when I started to turn the Raspberry Pi outside the plan xOy, I could see that I needed a Kalman filter for the IMU because it managed to reduce 20% the error. [Fig. 2.4]

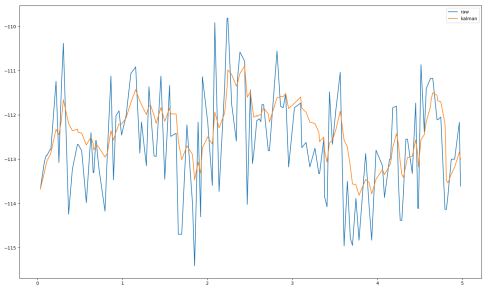

Figure 2.3: Kalman for IMU

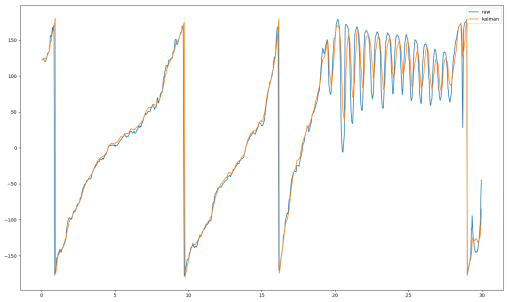

Figure 2.4: Kalman for IMU with vivration outside the plan xOy

2.2 GNSS receiver

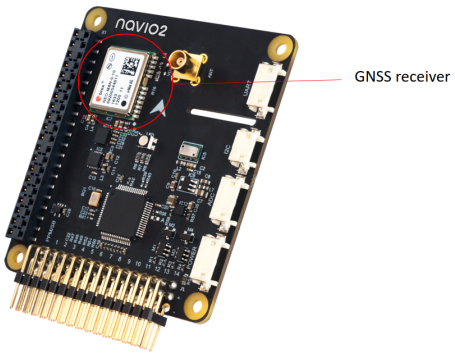

Figure 2.5: GNSS on the Navio2

We obtain the position and the speed from the GNSS receiver. [Fig. 2.5] Those data are coming from the navio2 with the ublox sensor. The longitude and latitude are obtain from the "NAV_POSLLH" message of the ublox and the speed from the "NAV_VELNED" message.

The error on the GPS is low (less than 1m) so I decided to not filter it. However, there are several jump in the speed so I add a low pass filter:

$$v = \alpha v_{measured} + (1 - \alpha) v_{previous} \tag{2.3}$$

$\alpha = 0.05$.

2.3 Weather Sensor

Figure 2.6: Weather Sensor : Calyspo Ultrasonic Sensor

The Weather data comes from the Calypso Portable Anemometer. [Fig. 2.6] This sensor is used in bluetooth from the Raspberry Pi using bluepy module and the mac address of the sensor. Thus, we obtain the vector of the wind: its angle and norm. For this sensor, I used a former program of Mr. Vautier that I adapted without ROS[Vau19].

We also apply a low pass filter for each data because there are also some jumps in the Calypso:
W $= \begin{pmatrix} x_{wind} \\ y_{wind} \\ \theta_{wind} \end{pmatrix}$
and

$$W = \alpha W_{measured} + (1 - \alpha) W_{previous} \tag{2.4}$$

$\alpha = 0.5$.

Chapter 3

Controlling the Servomotors

3.1 Collecting the data

Each sensors has its owm program. The algorithm collect all of the information in a state vector that filter those data. Then, the main program will control the actuator with a controller to the servomotors.

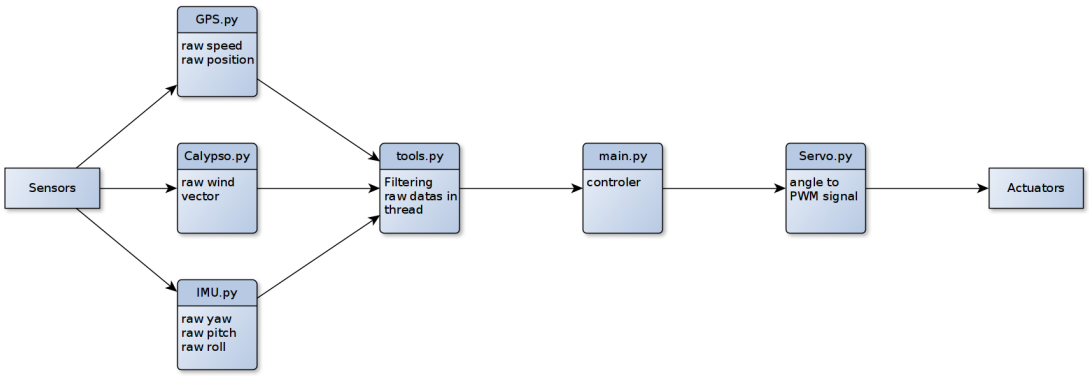

Figure 3.1: Graph of the use of programs.

3.2 Controller

3.2.1 Line following

We want to follow a line y = ax+b.

This line is created with the waypoints we want to follow.

- pwp : previous waypoint
- nwp : next waypoint

However, we only know the apparent wind and not the true wind which is needed for the rudder.

If we add the true wind and the wind that the movement of the boat create (with a velocity v) we get the apparent wind. Thus, we can deduce the true wind from the two others vectors. *However we have to be careful in our data from the wind that we* receive *and the* direction *that has the wind.*

Thus, we use this formula to get the true wind W:

$$T_W = \begin{pmatrix} a_{aw}.\cos(\phi_{aw}+\theta) + v.\cos(\gamma) \\ a_{aw}.\sin(\phi_{aw}+\theta) + v.\sin(\gamma) \end{pmatrix} \tag{3.1a}$$

$$\psi_{tw} = angle(T_W) \tag{3.1b}$$

Then, we can use the line following algorithm of Mr. Jaulin[LJ12].

inputs : $m, \theta, \psi_{tw}, a, b, q$
e = $\det(\frac{b-a}{\|\|b-a\|\|})$ $\phi = \arctan(b-a)$ $\hat{\theta} = \phi - \arctan(e/r)$ if$(\cos(\psi_{tw}-\hat{\theta}) + \cos(\xi) < 0)$ or $((\|e\| - r)$ and $(\cos(\psi_{tw}-\phi) + \cos(\xi) < 0))$: $\hat{\theta} = \pi + \psi_{tw} - q * \xi$ $\delta_r = \frac{\delta_{rmax}}{\pi} * sawtooth(\theta - \hat{\theta})$ $\delta_{smax} = \frac{\pi}{4} * (\cos(\psi_{tw}-\hat{\theta}) + 1)^{\frac{\log(\frac{\pi}{2\beta})}{\log(2)}}$
outputs : $\delta_r, \delta_{smax}, q$

Table 3.1: Controller for a line following

We also have to know which waypoint we have to follow. We reach one waypoint if $(nwp - pwm).(m - nwp) > 0$.

3.2.2 Station Keeping

Basic station keeping

The basic station keeping is to use the line following program and to stop the motors if the boat is around the point with a chosen radius r.
Once the boat is outside the circle, the motors turns on and the boat will try to go again to this point.

Avanced station keeping

To use an advanced station keeping we can use the code of Christophe Viel[Luc12]. The aims is also to reach the target point. When the sailboat is far from its target, the sailboat follows the controller of Mr. Jaulin[3.1] (Step 1). However, the angle of the wind can change the path of the sailboat. Thus, the sailboat needs to slow down when it became close from the target. That's why it first needs to turn in the bigger circle (Step 2) and goes against the wind (Step 3). Once it is against the wind, the sailboat move to the inner circle and wait, always against the wind (Step 4). [Fig. 3.2]

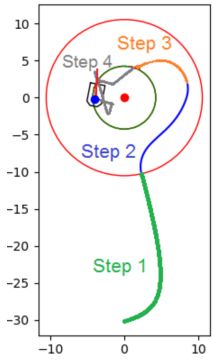

Figure 3.2: Sensor Connection

With this algorithm we create two circles from the new waypoint to reach. The closest from the waypoint has a radius of d_i, the second one has a radius of d_o. Finally, d_{ST} is the distance between the boat and the target waypoint.

The algorithm for the station keeping is described by the Table 3.2.2.

3.3 Servomotors

Two servomotors are needed for this project. Both can have a bigger angle for the rotation than necessary, so these actuators are calibrated by choosing a range of PWM signals corresponding to the desired angles.

1. The rudder : maximum angle of rotation of 180°. We need to control it to give an angle from -45° to +45°.
2. The sail : maximum angle of rotation of 2160°. We need to control it to give an angle from 0° to 90°.

Thus, we can suppose that this is a linear fonction and we can control the rudder with the formula (3.2) (f(δ) in ms).

$$f(\delta) = PWM_{min} + \frac{PWM_{max} - PWM_{min}}{ANGLE_{max} - ANGLE_{min}}(\delta - ANGLE_{min}) \tag{3.2}$$

Station keeping algorithm
inputs: $\mathbf{m}, \theta, \psi_{tw}, \mathbf{b}, q$
1. If $d_t > d_o$, compute δ_r and δ_s via lne following
2. if $d_i < d_t > d_o$, compute δ_r and δ_s as follows: if $cos(\tau - (\psi_{tw} - \frac{\pi}{2})) > 0$ if $cos(\alpha - (\psi_{tw} + \frac{\pi}{2})) > cos(\alpha - (\psi_{tw} - \frac{\pi}{2}))$ $\theta^* = \theta^* + \frac{\pi}{2}$ else $\theta^* = \theta^* + \frac{\pi}{2}$ if $\cos(\psi_{tw} - \theta^* + \cos(\xi)) > 0$ if $\sin(\psi_{tw} - \theta^*) > 0$ q=-1 else q=1 $\hat{\theta} = \psi_{tw} + pi + q\xi$ else $\hat{\theta} = \theta^*$
3. if $d_t > d_i$, compute δ_r and δ_s as follows: if $\cos(\psi_{tw} - \theta^* + \cos(\xi)) > 0$ if $\cos(\theta^* - (\psi_{tw} - \xi)) > \cos(\theta^* - (\psi_{tw} + \xi))$ q=-1 else q=1 $\hat{\theta} = \psi_{tw} + pi + q\xi$ else $\hat{\theta} = \theta^*$
4. if $\cos(\theta - \hat{\theta}) > 0$ $\delta_r = \delta_r^{\max} \sin(\theta - \hat{\theta})$ else $\delta_r = \delta_r^{\max} sign(\sin(\theta - \hat{\theta}))$ $\lvert\delta_s\rvert = \frac{\pi}{4} * (\cos(\psi_{tw} - \hat{\theta}) + 1)$
outputs : δ_r, δ_s and q

Table 3.2: The controller for station keeping

Chapter 4

Communication

4.1 Real time simulation

Why communicate between the computer and the sailboat ?

The first thing needed was a real time simulation of the boat. Thus, I can easily control what happen with our sensors.

I used the XBEE device in order to have the data in real time. The XBEE are long-range radio devices. I used the library serial to encode the data in string to bytes on the Raspberry Pi and decoded them on our computer in the other way. Thus, I plugged one in USB on the Raspberry Pi and the other one on our computer[XBE22] which allowed us to configure them:

Parameters	**XBEE 1**	**XBEE 2**
PAN ID	4820	4820
DH (Destination High)	0	0
DL (Destination Low)	0	1
MY (Source Address)	1	0

I needed to configure all the XBEE on the same channel, then with the same ID. When this was done, one XBEE is used as the coordinator[1] and the others as end points. This way the XBEE could discuss with the coordinator and not between them. With this configuration, I sent on one way the data that the sailboat measured with its sensors to the computer. It allowed to display the boat in an approximated real time with a delay of 0.2s[2]. [Fig.4.1]

In this case I used the library matplotlib to see the sailboat, modifying the code "draw_sailboat" of Mr. Luc Jaulin.[Luc15][Luc15]

However the angle δ_{smax} given by the controller is not the true δ_s of the sail : it should depends on the wind. Thus, I had to add a formula for the visualization which also changed the angle from 0° to 90° to -90° to 90°.[3]

Added part in the algorithm for the visualization of the sail.
$\sigma = \cos(\psi_{ap}) + \cos(\delta_{smax})$
if $\sigma < 0 : \delta_s = \pi + \psi_{ap}$
else: $\delta_s = -sign(\sin(\psi_{ap})) * \delta_{smax}$

[1]The coordinator will broadcast data to all connected end points
[2]which is also the frequency of the main program
[3]This added part should not be in the controller, just for the visualization in matplotlib.

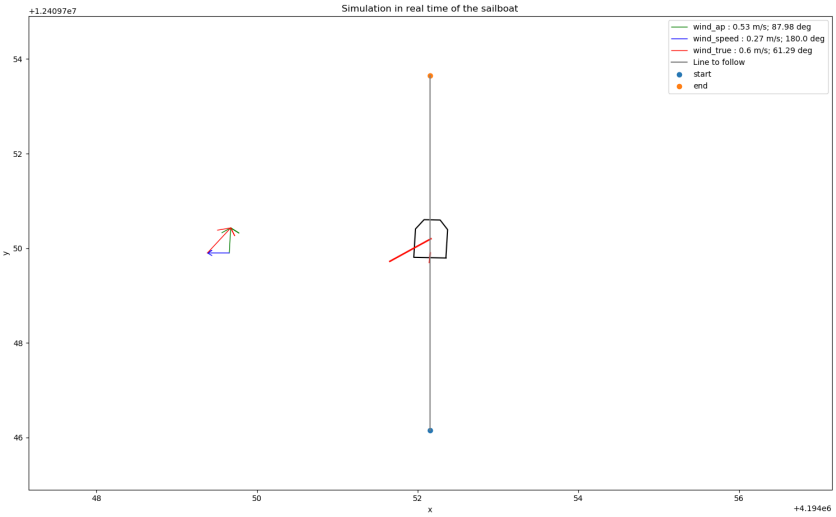

Figure 4.1: Real time simulation

Eventually and because of the trigonometry I had to convert the 0° from east to north. Thus I added $\frac{\pi}{2}$ on θ and γ.

You can see a demonstration of the program here[4].

4.2 Manual control

The second idea was to control the boat using the computer and the XBEE if the mission failed.

- A program could be used to control the boat with the XBEE.
  The Raspberry Pi checks continuously if it receives a message. Thus, if I wanted to control the servomotors, I will send a message with 3 values : a byte 1 for True, the angle of the sail, the angle of the rudder.
  The value of those angles were given with the keyboard. If the XBEE control was on, the autonomous mission will stop.
  To turn off the XBEE control to see the sailboat go back again on its autonomous mission press "r".
- The RC had to be calibrated : the maximum and minimum of the PPM signal created by the joystick match with the maximum and minimum angle of the sail and rudder.
  If I start using the RC, then it will stop the main program or the control of the XBEE.

You can see a demonstration of the program here[5].

[4] I set the speed at 1m/s and blow in the Calypso's direction. Thus, I could see the true wind and the apparant wind created.

[5] When the program is launch it start to send data to the raspberry using XBEE. However the first byte will be false until I set an angle using the directional arrow on our keyboard.

4.3 Display the boat on a map

However, one of the main issue with matplotlib is the fact that I couldn't see the sailboat in real time on a map. Thus, thanks to Mr. Pelle, a student working with me that made a server with open map[Pel22], I was able to create a program to display the boat. [Fig. 4.2]

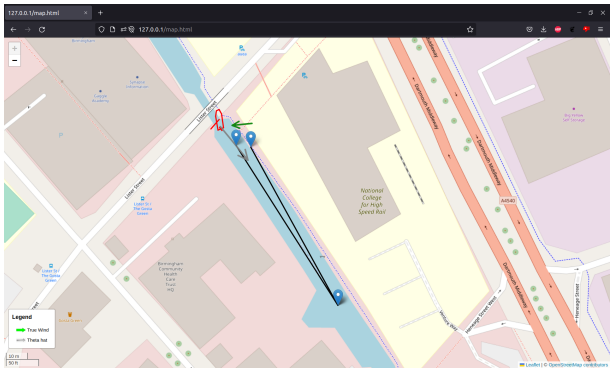

Figure 4.2: Map on the server

Two different way of using this server were useful:

- When I tested the sailboat in real time, to see where the boat was on the lake.
- Once I replayed the tests with logs.

In any cases, the way the algorithm works is the same. Firstly, a python script runs with the catching data from the log file or the receiving data from the XBEE. It sent them to a mysql database. The mysql database is named "sailboat" with several tables (wind, direction, position, waypoints,..). A PHP file gets the sql data and send them in a JavaScript function that a HTML file use to display everything on the internet. To create this map, I used the library leaflet. [Fig. 4.3]

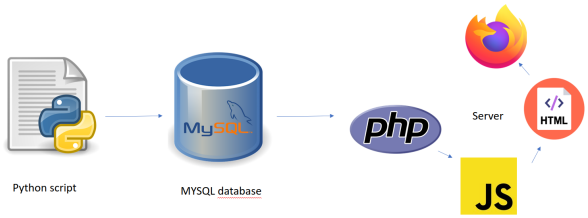

Figure 4.3: Monitoring scheme to display the sailboat

I was now able to see where is the boat, the waypoints on the map and the angle he follows $\hat{\theta}$, the true wind, its yaw and angle of the sail/rudder on my computer.

Chapter 5

Final build and test

5.1 Build a support for the Raspberry Pi

The sailboat I used is called "Regazza proboat". [Fig. 5.1].

Figure 5.1: Sailboat ragazza proboat and the system inside

Two supports are on the hull: one on the front for a camera[1] and another one at the bottom for the calypso weather vane.
Furthermore, this sailboat has two sails which are moving at the same time using a servomotor inside the hull. The first servomotor is a winch (the black cylinder) that stretch the sail with

[1] the sailboat was used for championships and it had to recognize figure when it was on the water.

the rope. The second one is used to control the rudder below the servomotor for the sail.

However and because it is the first year that the boat is fully controlled with a Raspberry Pi there was no support for this one. Thus, I had to build one using the open source application Free Cad. [Fig. 5.2]

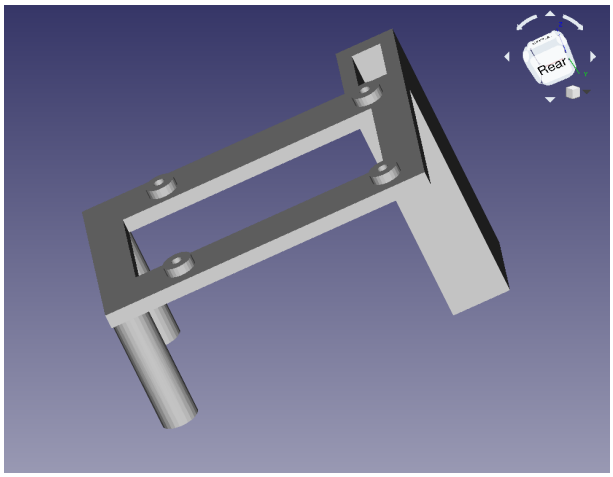

Figure 5.2: Support for the Raspberry Pi

I assembled all of the sensors and servomotors to the Navio2 and the Raspberry Pi [Fig. 5.3].

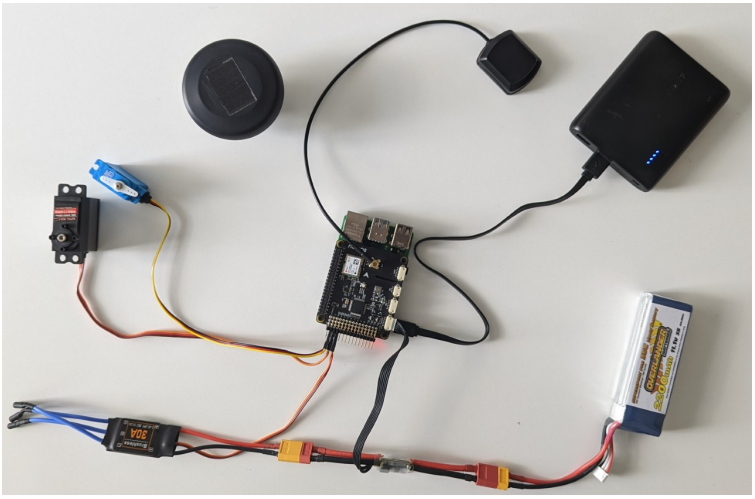

Figure 5.3: Sensor Connection

We can see the two servomotors on the left (the black one for the sail and the blue for the rudder).
On the top left the Calypso which is connected in bluetooth.
The GNSS antenna on the top.
The two batteries are on the right:

- The USB battery is on the top right
- The LiPo battery is on the bottom right

The LiPo doesn't bring enough battery for the raspberry, the Navio and the servomotors so I plug another one in USB. At the end of my internship, I replaced the three cell battery with a 5 cells so everything can be powered by this one.

To control the servomotors I also had to add an ESC[2] (on the bottom of 5.3) that converts the PWM signal to the appropriate level of electrical power for the servomotor and a BEC[3] (on the bottom left of 5.3) which regulates the voltage from the battery to 5V.

Then, I added the communication part with the XBEE and the RC. [Fig. 7.1]
Finally, I was able move on the components inside the hull [Fig. 5.4]:

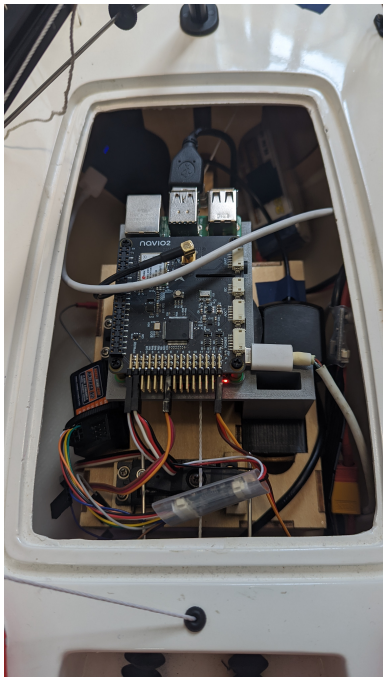

Figure 5.4: Inside of the hull

5.2 Testing the boat on the water

Before sending the boat on the water I had to make a program called "checking" to see if every component is working inside the boat. [Fig. 5.5]

When I checked that everything worked correctly, I finally launched the boat on a canal of Birmingham. [Fig. 5.6] Its mission was to follow a line and come back near the border to catch it.

However, the canal was surrounded by buildings, which led to GPS imprecision of 5m[4]. Thus, I had to go somewhere else : the Bournville Radio Sailing and Model Boat Club. It's a Lake near Birmingham where people used to go there to radio-control sailboats.

[2]Electronic Speed Controller
[3]Battery Elimination Circuit
[4]half the size of the canal

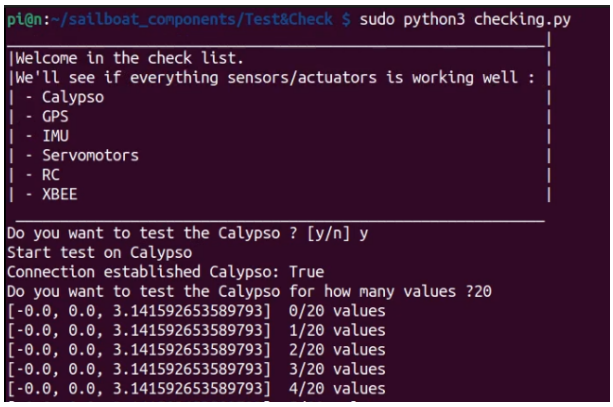

Figure 5.5: Program to test the Sensors and actuators - video

Here I tested the following mission :

1. A line following of two points
2. A line following of three points (triangle)
3. A station keeping
4. A line following with station keeping

On the lake, I used my computer [Fig. 5.7] to see where the sailboat aimed at and where it was in its mission (on the right of my computer) and all the data that I received from the XBEE (on the left).

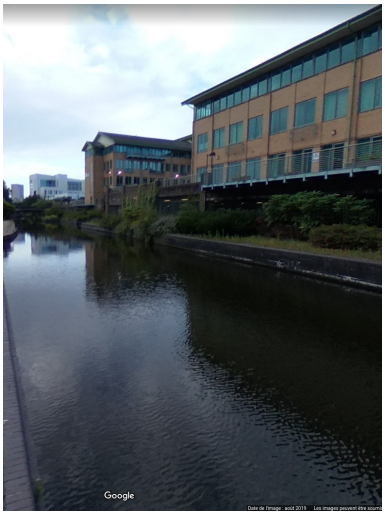

Figure 5.6: Canal near Aston University

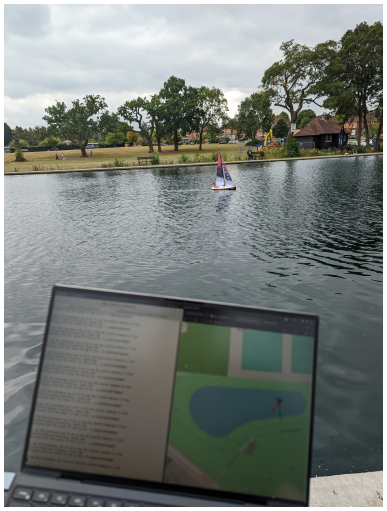

Figure 5.7: Test of the boat on the lake

5.2.1 Test of the line following

The first example of a simple line following mission was to cross the entire lake with two GPS points [Fig. 7.2], doing a loop between them without stopping. This way, I could easily see where

the sailboat was on its mission and if the algorithm was working. [Fig. 7.3]. I quickly saw that the sailboat didn't go completely to the waypoints. It was due to a lost the GNSS signal two times which made the sailboat going beyond the next waypoint. Thus, the algorithm believed that the sailboat reached the next waypoint.

To overcome this issue and because I were running out of time, I had temporarily modify the algorithm to confine the boat inside a rectangle where the lake was : if the position was outside the rectangle the algorithm didn't take the value. I tred this on a line following triangle mission. [Fig. 7.4, 7.5]

However, I also saw that I lost the GPS signal very often. Thus, for the next test, I decided to change the hardware by putting the GPS antenna outside the boat, creating a waterproof pathway.

Thus, the GPS signal was much better and I never lost the GPS signal again [Fig. 7.6, 7.7][5] [Fig. 7.8, 7.9].

Nonetheless, There was not much wind so the boat believed (surely because of the sail) that the wind came in from it. It explained why the sailboat applied the upwind "zig-zag" technique during all along its way.

Thus, I decided to wait until the wind raised a bit more and I applied both station keeping and line following mission for the next experiences.

5.2.2 Test of the station keeping

Firstly, I tested the station keeping [Fig. 7.10, 7.11]. The sailboat had to go to one point and stay for one minute. On the experience he stayed for 84s.

Then, I decided to combine both station keeping and line following. The sailboat had to follow a line to a point and then apply a station keeping mission. Once he reach it for 60s for example, he had to go to the next one. The path was composed of three points [Fig. 7.12, 7.13].

The boat goes starts to follow a line from the border of the lake to a point in the middle upwind: it did the "zig-zag" technique. Then, the sailboat reach the waiting area into with first circle and apply the station keeping strategy for the total time of its mission. After 60s, he went the final point, and started to do another station keeping before I stopped it with the RC.

[5] I had to stop the mission here with the RC because of the lack of wind, the mission was taking too long

Chapter 6

Conclusion

With a new set of sensors, the main purpose of the internship was to build a working sailboat, able to do autonomous missions. Thus, the idea was to rethink a previous system for this sailboat using Arduino and ROS into another one using only Python.

Thanks to Mr. Wan and Aston University, I was able to set up this sailboat and test it in real condition. Concerning the sensors, the IMU and GNSS were both into directly on the Navio2 while the wind sensor was outside the boat, connected in Bluetooth. The two actuators were one for the sail and the other for the rudder.

The first purpose was to collect the data from the sensors into separate programs. Then I had to filter the data, calibrate the IMU in order to have the most accurate values. Once it was done, I had to gather them into a main program to give a value of angle to the actuators. However, some sensors blocked the program while waiting for a value. So, the algorithm for each sensors had to work in parallel, storing each value (position, speed,..) into a state vector. This way, the main program only read this vector and then had a good frequency (quickly calculate the angle to give to the servomotors). Finally, I had to calibrate the servomotors to match a pwm value with a desired angle.

However, even if the software and hardware were working, it is useless if I couldn't communicate between the computer and the sailboat, to see what the boat is doing in real time. Thus, to get rid of the wifi, I used a radio device able to exchange data between the two devices. This way I could control if the values given to the servomotors were correct and if the algorithm was working.

All in all, I tested the sailboat in real condition with line following and station keeping algorithms, building a support for the Raspberry Pi . The first experiment were not good enough because of the error on the GNSS so I had to put it outside. Then the sailboat was able to do its mission properly using both a station keeping and line following mission.

Chapter 7

Appendices

Annex A: Sensors connection

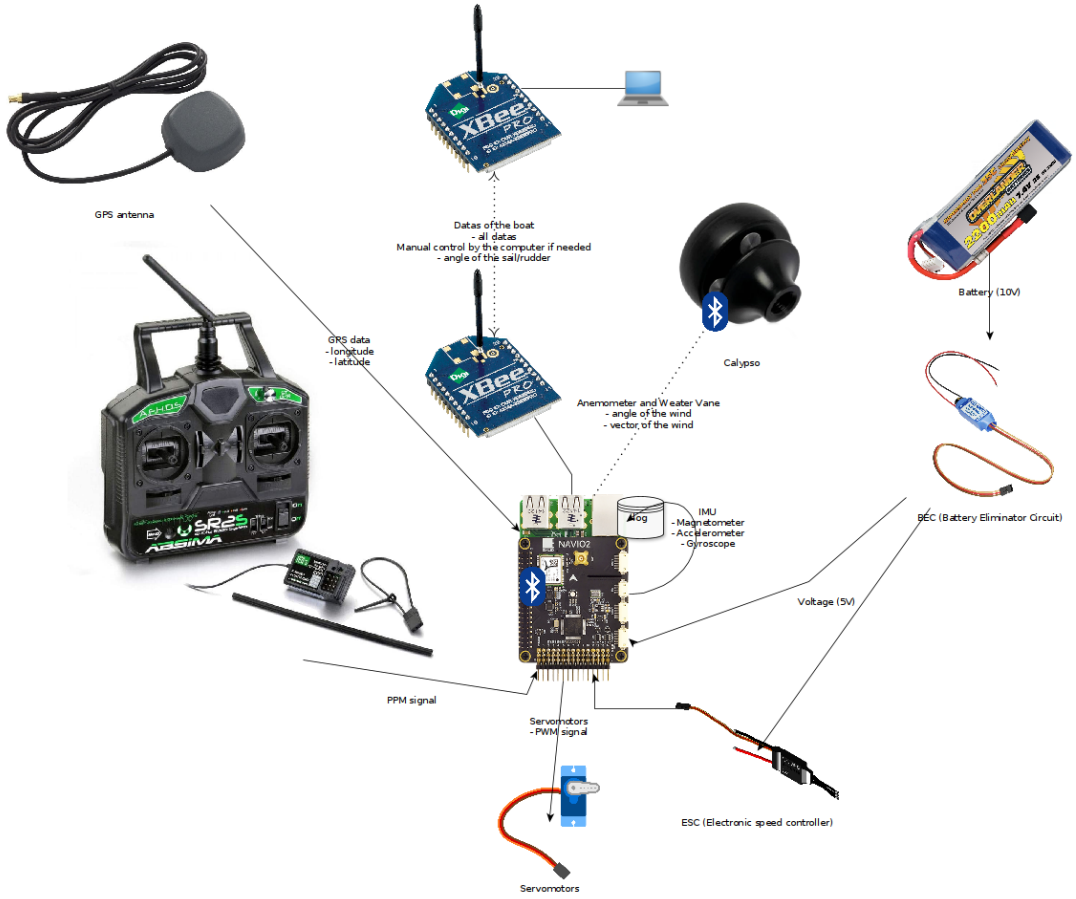

Figure 7.1: Scheme of the sensors connection

Annex B : Results of the experiments

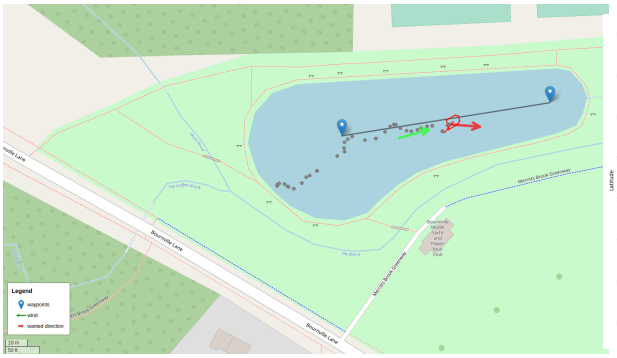

Figure 7.2: First test of simple line following - on the map, demonstration here.

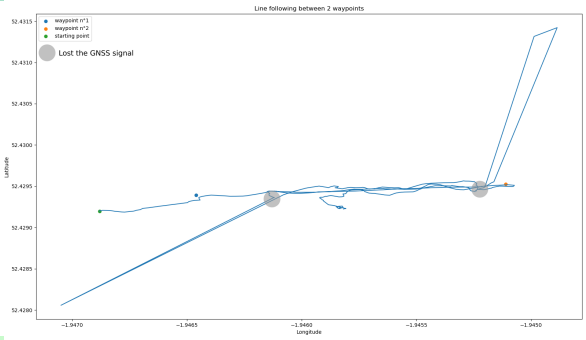

Figure 7.3: First test of simple line following - result

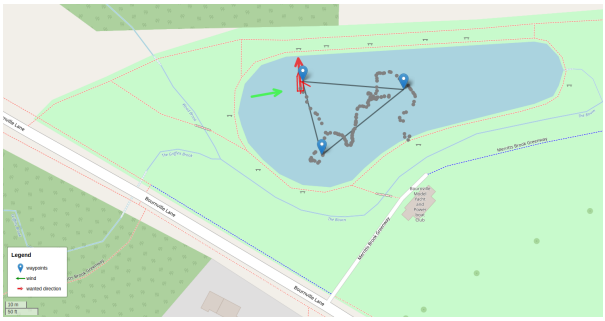

Figure 7.4: First test of triangle line following - on the map, demonstration here.

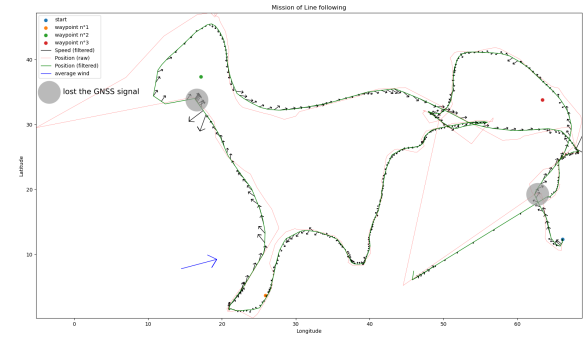

Figure 7.5: First test of triangle line following - result

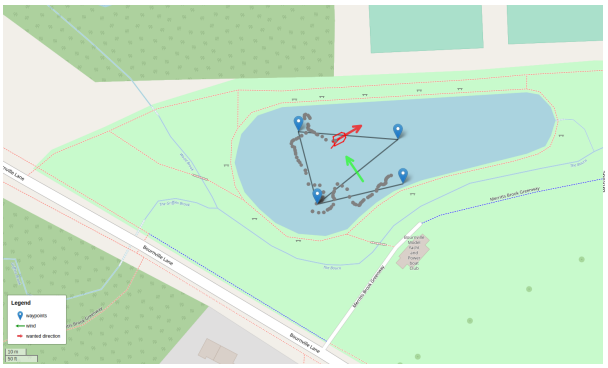

Figure 7.6: Second test of triangle line following - on the map, demonstration here.

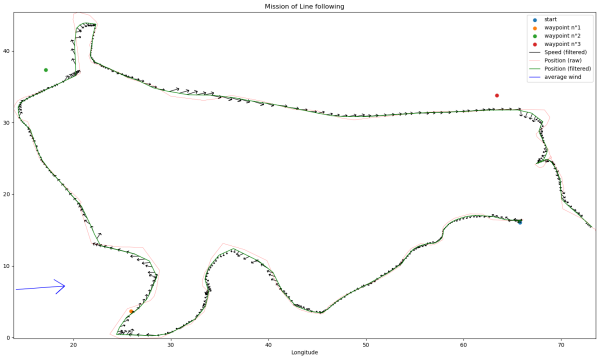

Figure 7.7: Second test of triangle line following - result

Figure 7.8: Third test of triangle line following - on the map, demonstration here.

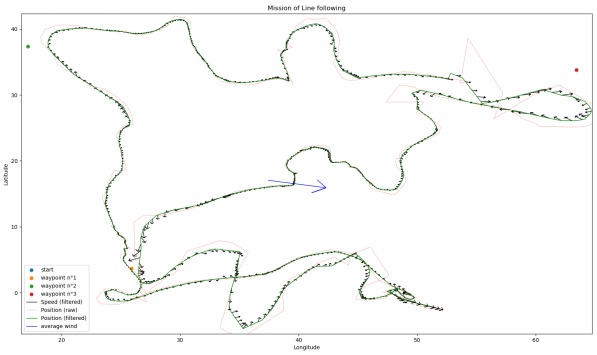

Figure 7.9: Third test of triangle line following - result

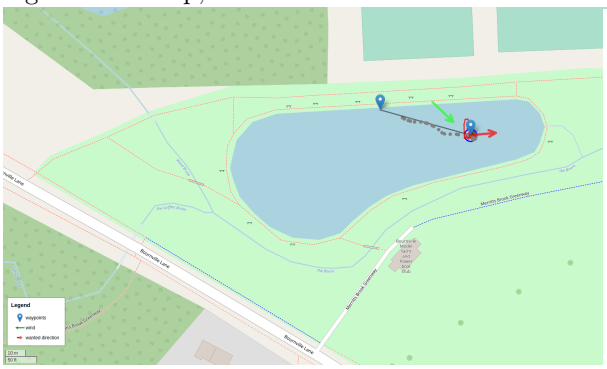

Figure 7.10: First test of station keeping - on the map, demonstration here.

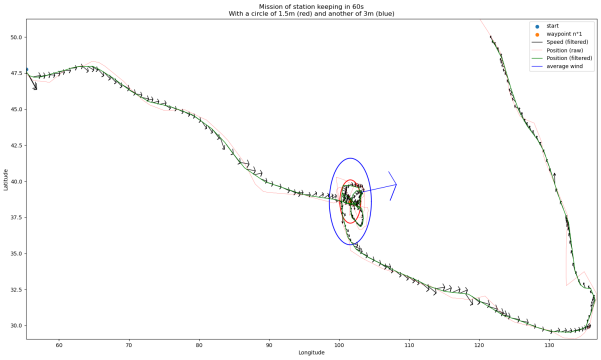

Figure 7.11: First test of station keeping - result

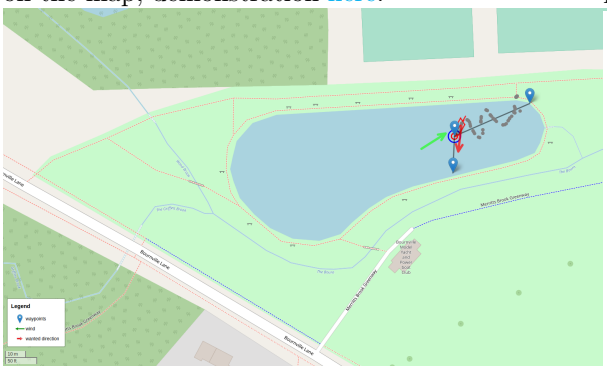

Figure 7.12: Second test of station keeping - on the map, demonstration here.

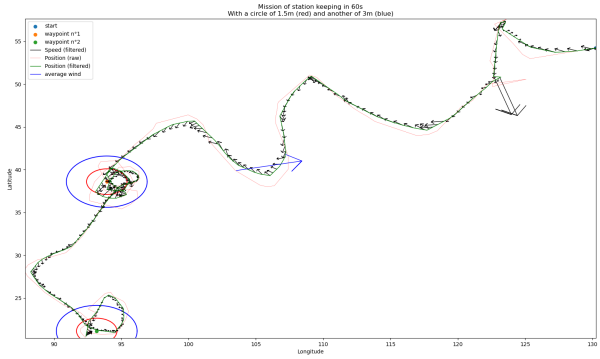

Figure 7.13: Second test of station keeping - result

Bibliography

[Aug22] Morge Augustin. Gitlab Repository, 2022.

[cal] Python Program to transform a ellipsoid into a sphere.

[LJ12] Fabrice Le Bars Luc Jaulin. A simple controller for line following of sail-boats. 2012.

[Luc12] C. Viel; U. Vautier; J. Wan; J. Luc. Experimental results of position keeping for autonomous sailboat. 2012.

[Luc15] Jaulin Luc. *Mobile Robotics*. iSTE, 2015.

[Mou08] M.C. Mouchot. Navio2 Documentation, 2008.

[Pel22] V. Pelle. Github repository, 2022.

[Vau19] U. Vautier. Github repository, 2019.

[Wik22a] Wikipedia. Birmingham, 2022.

[Wik22b] Wikipedia. Aston Univerity, 2022.

[XBE22] Zigbee RF Modules. 2022.

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

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* j.wan3@aston.ac.uk

Nom du stagiaire accueilli / *Name of intern* Augustin Morge

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 Birmingham, on 26/08/2022

Signature Entreprise _____ Signature stagiaire
Company stamp _____ *Intern's signature*

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