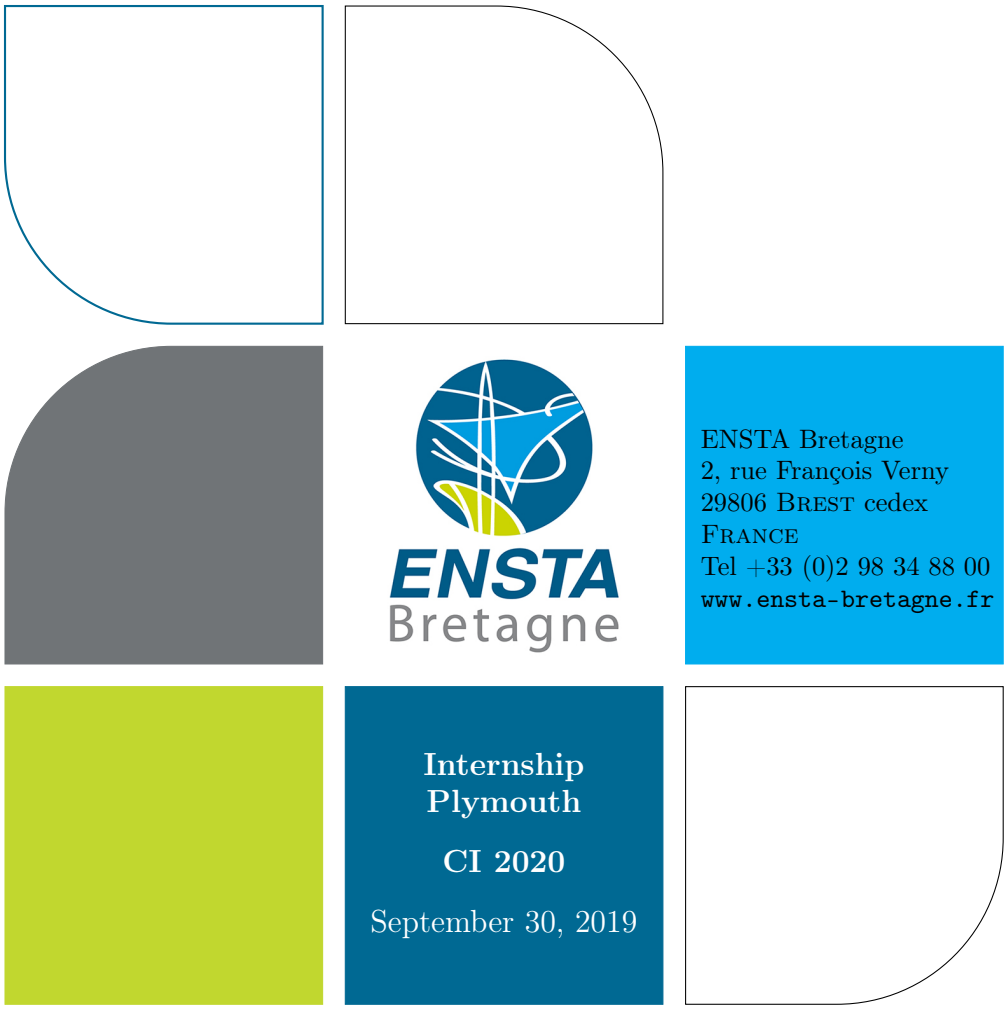

ENSTA Bretagne
2, rue François Verny
29806 Brest cedex
France
Tel +33 (0)2 98 34 88 00
www.ensta-bretagne.fr

Internship
Plymouth

CI 2020

September 30, 2019

Autonomous Sailboat

Alexandre Courjaud
Alexandre.Courjaud@ensta-bretagne.org

Abstract

This report explains the work I did during my 2nd year internship at Plymouth University. The subject of the course is the design of an autonomous sailing robot. This system was tested during the WRSC and can be used for research. During this training I worked on the low level of programming, the controllers of the sailing boat as well as on the simulation. I will introduce the general system before approaching these 3 parts in this report, and finally reflect on the participation in WRSC 2019 in China. I did this internship with Corentin Jegat, Matthieu Bouveron and Alexandre Argento.

Résumé

Ce rapport explique le travail que j'ai réalisé lors de mon stage de 2eme année à l'université de Plymouth. Le sujet du stage est la conception d'un système autonome pour voilier. Ce système a été testé lors de la WRSC et pourra servir pour la recherche. Lors de ce stage j'ai travaillé sur le bas niveau de programmation, les contrôleurs du voilier ainsi que sur la simulation. Je présente le système général avant d'aborder ces 3 parties dans ce rapport, pour finir sur la participation à la WRSC 2019 en Chine. J'ai réalisé ce stage avec Corentin Jegat, Matthieu Bouveron et Alexandre Argento.

Acknowledgements

I would like to begin by thanking Jian Wan, a researcher at the University of Plymouth, for welcoming me, for accompanying me and for making this internship possible. I would also like to thank Ulysse Vautier, a PhD student at the University of Plymouth, for helping me throughout this internship. I would then like to thank Luc Jaulin, researcher at ENSTA Bretagne, for getting me in touch with Jian Wan and allowing me to carry out this internship. Finally I want to thank Corentin Jegat, Matthieu Bouveron, Alexandre Argento with whom I realized this 3-month internship.

Contents

Introduction

In order to validate my 2nd year at the ENSTA Bretagne I realized an internship at the University of Plymouth. The internship was on autonomous sailing boats. More precisely, the main objective of this training course was to design a system for an autonomous sailing boat of one meter. I chose this course because it allowed me to put into practice all the knowledge I had learned during the year in robotics. In addition, I did not have the opportunity to work in marine robotics during the year's projects. This internship was therefore the opportunity to discover and work in this field. This internship was supervised by Jian Wan and Ulysse Vautier.All the codes I have created are in my github repository [2].

1 System engineering

1.1 Objectives

To begin, this 3-month course was divided into several objectives. The main objective, the one for which I was recruited, was to design a robust and reliable system to make an autonomous sailing boat with ROS. The second objective was to participate in the World Robotics Sailboat Cup. Finally the 3rd objective was to show that the system designed could be used to do fleet control for example.

1.2 Starting point

In order to start our work, Jian Wan gave us the following equipment:

- Arduino Uno Board
- Module Grove Base Shield
- 2 Actuators
- Adafruit 16-Channel 12-bit PWM/Servo Shield
- Module IMU 9 DOF Grove
- Wind sensors with a vane and a anemometer
- Anker Battery 20100 mAh
- Raspberry PI 3b+
- Module GPS Grove
- 1 Ragazza Proboat

In addition, we had access to Ulysse Vautier's codes on the same subject [5]. However the system set up was different and did not use the same sensors. These codes were also useful to start.

Figure 1.1: Ragazza proboat

1.3 Constraints

To achieve the sailing boat's autonomous system, the following constraints must be met:

- Use ROS. The use of ROS has been chosen to provide a robust system with standardized messages.
- Do not drill holes in the boat. To ensure the waterproofness of the boat we had to make no hole in it.
- Make a system easily adaptable on another boat. The system had to be simple and standard enough to be bought and equipped on a new boat quickly.

1.4 Hardware architecture

We started this system from scratch. After several tests the final hardware architecture of the system that we realized is drawn in the figure 1.2.

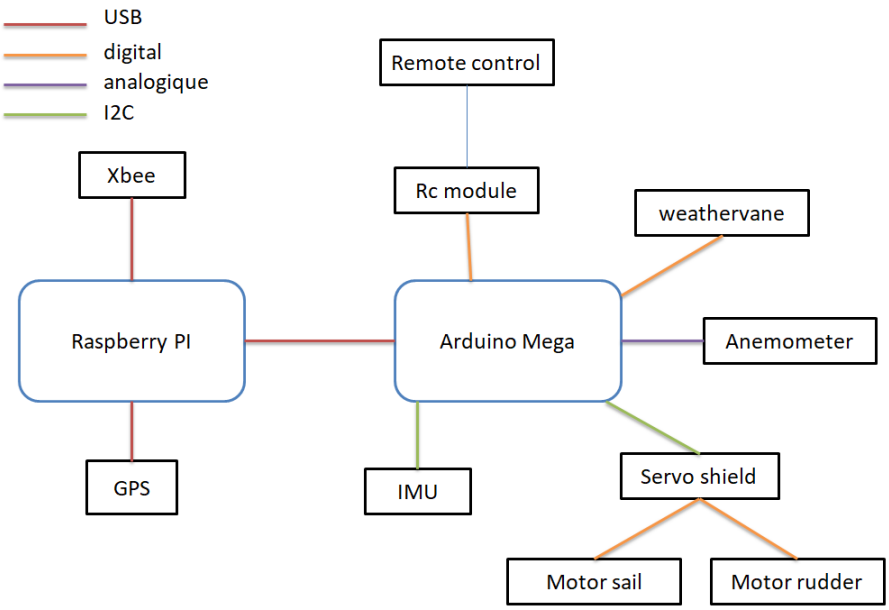

Figure 1.2: Hardware architecture

The choice of sensors and their operation are explained in the next chapter. This diagram shows all the components included in the system: a Raspberry PI board, an

Arduino board, a wind sensor, an imu, 2 servomotors with a shield, a GPS, a remote control and finally an Xbee for outward communications.

1.5 Software architectural design

And here's how we set up our program to make it all work.

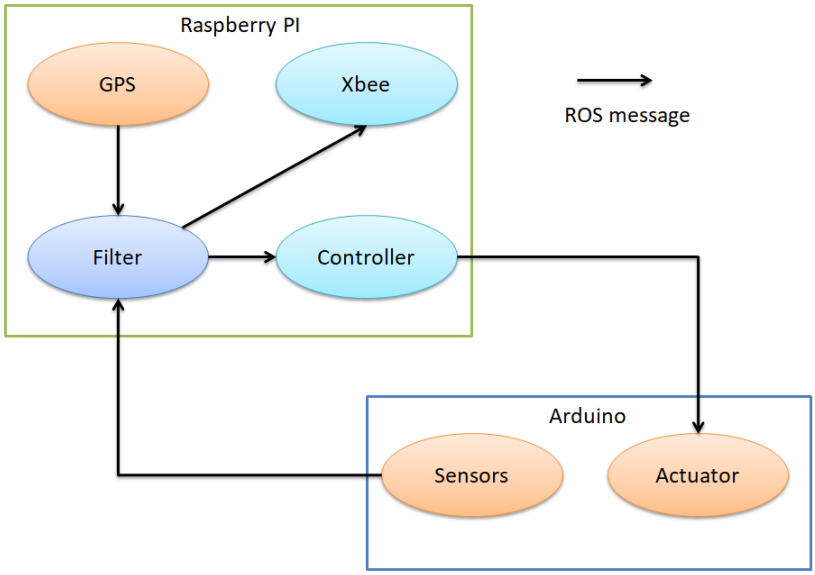

Figure 1.3: Software architecture

We chose to concentrate the low level (using sensors and motors) on the Arduino board. The Raspberry board contains the filters as well as the controllers. We chose not to filter the data directly on the arduino board to lighten the program and get the best possible frequency on the sensors. The 2 cards are connected by a serial connection via usb. You can find explanations about the filters and other controllers in the report of Corentin Jegat. Communication between systems can be found in Matthieu Bouveron's report. The system has been coded on ROS.

2 Low-level programming

In this part I will discuss in detail the work I did on Arduino during this course, as well as my choices in board and sensor.

2.1 Arduino

First, the use of an Arduino board to manage the sensors was required. To facilitate the connections I had at my disposal a Grove shield, and an Adafruit motor shield to control the sail and the rudder.

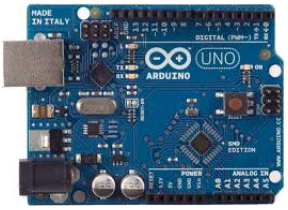

Figure 2.1: Arduino Uno

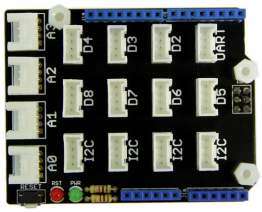

Figure 2.2: Module grove base shield

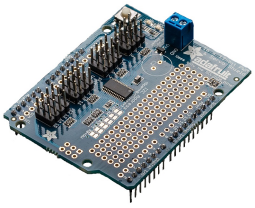

Figure 2.3: Adafruit Servo Shield

However the use of ROS on Arduino requires a lot of dynamic memory. The creation of the node represents 61% of the dynamic memory of an Arduino UNO. This limits the use of topics and other libraries. So I chose to use an Arduino MEGA that offers 4 times more memory.

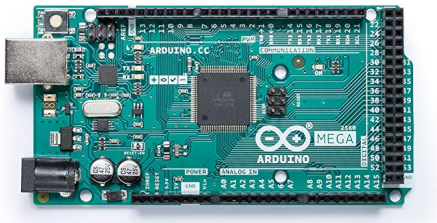

Figure 2.4: Arduino Mega

2.2 Sensors and actuator

2.2.1 Inertial measurement unit

This sensor mainly served us to measure the course of our boat, indispensable for a good regulation. That makes this sensor the most important part of the system. This explain why Jian offered us a second sensor during our tests: the cmps12.

Figure 2.5: Imu 9Dof

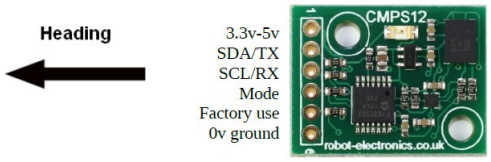

Figure 2.6: Imu cmps12

Unlike the 9Dof module, it can run in series, can be self-calibrated and can read Euler angles directly. However thanks to the filter made by Corentin Jegat, explained in his report, we had more reliable and stable results with the 9Dof module so we chose it.

Data acquisition

The 9Dof module works with i2c so requires the use of the wire library.h to be simple. The data acquisition allows obtaining the 3 components of the acceleration in m/s^2, the 3 components of the angular velocity in degrees/s and finally the 3 components of the μT magnetometer.

Data communication

As previously presented, the system does not process data on the Arduino board in order to optimize the reading frequency. So I chose to use the ROS sensor_msgs Imu message to communicate the 9 raw elements on a single topic, with the acquisition time.

Sensor test

Sensor testing without calibration or filter is not accurate. However it is possible to check roughly that the accelerations and the magnetometer work. For this it is enough to put the sensor on a table, if the table is straight then the acceleration on the axis z is equal to the

acceleration of gravity. You can check x and y the same way. To check the magnetometer it is necessary to try to set 2 elements to 0, after calibration of the sensor the 3rd element is the North for the sensor.

2.2.2 Module GPS grove

The grove GPS module runs in series with the Arduino board. The module communicates multiple NMEA frames at a frequency of 1Hz. However, the time it took to acquire the frames on the Arduino board considerably slowed down the acquisition frequency of the other sensors. That's why we decided to change the GPS to a USB GPS directly connected to the Raspberry PI. We chose the Globalsat BU-353-S4.

Figure 2.7: GPS grove

Figure 2.8: BU-535-S4

This choice allowed us to acquire and process the data directly on the Raspberry.

Data acquisition

In the same way as the grove module, this module communicates frames in series. Then you have to sort the frames so that you only select the one you needed. So I chose the GPGGA and GPRMC frames to keep the time, the latitude, the longitude, the number of satellites used, the horizontal precision, the speed on the bottom and the heading in degrees. It is then necessary to ensure that the frames are valid, for this reason each frame has a parity check sum, so by performing an XOR on the characters of the frame then it is possible to ensure that the frame is valid. After several tests I noticed that the sensor data was stable so I did not make a filter to process them.

Data communication

To communicate the data I chose the ROS GPSfix message from GPS_common. This message is used because it takes into account all the data that can be recovered from the frames I have used, and even more if there is a need to improve the acquisition code. Latitude and longitude are converted and reported to decimal for ease of use by controllers. Moreover I chose to communicate in a message string the complete GPS frame so that it is easier communicable with the Xbee programmed by Mathieu Bouveron.

Sensor test

The test of this sensor is simple. Just read the frames and check them via google map for example.

2.2.3 Wind sensor

The wind sensor is divided into two sensors: an anemometer and a weather vane. I used the Standard Weathervane Anemometer – 7911 – Davis Instruments sensor.

Figure 2.9: Wind sensor

Data acquisition

The weathervane part works in analog and returns a value on 10 bits that can be converted to obtain an angle. I chose to place this angle between $-\pi$ and π. The raw data corresponds to the apparent wind in the ship's coordinate system; the zero indicates a back wind. The anemometer part works with 1 bit, the sensor sends one pulse per turn. By detecting this pulse and measuring the time between 2 pulses it is possible to deduce the sensor speed and then the apparent wind speed. In the same way as for imu, the measured data need to be filtered. These filters were made by Corentin Jegat and are explained in his report.

Data communication

The filters are not made on the Arduino board, so the raw values must first be communicated. I chose to communicate the wind speed at each sensor pulse while the weathervane values are communicated at the program frequency. So the two sensors do not have the same frequency. So I used 2 different ROS messages of double type for these values. After filtering there is only one topic with the real wind in the map coordinate system.

Sensor test

To test the weathervane, the raw values of the analog port must be displayed and the value range returned by the sensor checked. After working with 3 weathervanes from the same manufacturer I noticed that all the sensors did not use the 1024 values, it is possible that the max value of a sensor is less than 1023. In this case the conversion function must be adapted. To test the anemometer it is sufficient to rotate it and verify that one pulse per turn is obtained.

2.2.4 RCmodule

To ensure the safety of the system, I used the RC module with the original remote control to control the boat if the autonomous mode ceases to function.

Data acquisition

I used 2 chanel of the RC module, one for the sail and one for the rudder. When the remote is turned on, each chanel sends pulses out. By measuring the pulse time we obtain the value communicated by the remote control. If no pulse is detected for two seconds then I considered in my program that the remote was off.

Data communication

The remote control acts directly on the motors also in the Arduino board so there is no need to communicate the data. However I still used a message ROS Vector3 to indicate the status of the remote control and make easier the analysis of the data during the tests of the boat.

Sensor test

To ensure the operation of the remote control and the RC module it is sufficient to read the pins of the Arduino board, using the pulseIn() function for example, checking the reception of values when the remote is on and the absence of a signal when it is off.

2.2.5 Actuators

The boat has only 2 actuators, one servomotor for the rudder and one for the sail.

Control

To control the engines I used an adafruit shield (figure 2.3). This shield connects in i2c to the Arduino board and allows having a separate power supply more powerful. For communication with the shield, adafruit provides a ready-to-use library.

Command acquisition

There are 2 ways to acquire orders for actuators. The first is the use of the remote control, directly connected to the Arduino so there is no special message. To receive the controller commands from the Raspberry PI I used 2 ROS messages of double type, one for each servomotor. For the sail the program expects a command between 0 and $\pi/2$ corresponding to the sheet length. For the rudder the program expects a command between $-\pi/2$ and $\pi/2$ corresponding to its angle. I have added safeguards to the playback of messages that make it impossible to request an impossible motion from the actuators.

Test

To test the actuators it is necessary to send several commands, starting with 0 and incrementing. This also allows finding the limit values of the servomotor.

2.3 Main program

To integrate each component into the main program I realized a setup, update and publish function for each sensor, and the setup and update functions for the actuators. The setup function of my Arduino program initializes all the components, and then the loop recovers the commands from the controller, sends them to the motors, updates the sensors and finally publishes the measurements. In addition, the variable initializations necessary for the operation of each component are performed in configuration files, included at the beginning of the main program. Thus, to add a sensor it is necessary to perform these 3 functions, the configuration file and add the function calls in the setup and loop. To change the sensor configuration, the initializations in the associated configuration file must be changed. After optimization, the program is able to perform a loop at a frequency of 20Hz.

3 Simulation

An autonomous sailing ship system is much more complicated to test than a terrestrial robot. That's why I did a simulation with ROS to make sure the codes didn't have any major problem.

3.1 The model

To carry out the simulation I used the following model:

$$
\begin{cases}
\dot{x} &= v\cos\theta + p_1 a\cos\psi \\
\dot{y} &= v\sin\theta + p_1 a\sin\psi \\
\dot{\theta} &= \omega \\
\dot{v} &= \frac{f_s\sin\delta_s - f_r\sin u_1 - p_2 v^2}{p_9} \\
\dot{\omega} &= \frac{f_s(p_6 - p_7\cos\delta_s) - p_8 f_r\cos u_1 - p_3\omega v}{p_{10}} \\
f_s &= p_4||W_{ap}||\sin\delta_s - \psi_{ap} \\
f_r &= p_5 v\sin u_1 \\
\sigma &= \cos\psi_{ap} + \cos u_2 \\
\delta_s &= \begin{cases} \pi + \psi_{ap} & \text{if } \sigma \leq 0 \\ -\text{sign}(\sin\psi_{ap})\cdot u_2 & \text{otherwise} \end{cases} \\
W_{ap} &= \begin{pmatrix} a\cos(\psi - \theta) - v \\ a\sin\psi - \theta \end{pmatrix} \\
\psi_{ap} &= \text{angle } W_{ap}
\end{cases}
$$

Where (x, y, θ) corresponds to the posture of the boat, v is its forward speed, ω is its angular speed, f_s (s for sail) is the force of the wind on the sail, f_r (r for rudder) is the force of the water on the rudder, δ_s is the angle of the sail, a is the true wind speed, ψ is the true wind angle and W_{ap} is the apparent wind vector. The quantity σ is an indicator of the sheet tension. In these equations, the p_i are design parameters of the sailboat [4].

However, not having all the parameters of our boat we used those of another boat provided by Ulysse Vautier. Even if the behaviour of our boat was not perfectly reproduced,

this model allowed us to check the proper functioning of our regulators.

3.2 Display

In order to visualize the results of the simulation I also programmed a display with Ros and Rviz. The operation of the display is quite simple: the node retrieves all the data useful to the position of the boat (GPS, Euler angles, wind, control of the rudder and sail, the objective) then displays them.

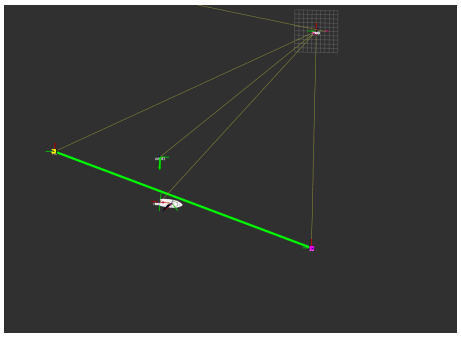

Figure 3.1: Screen Rviz

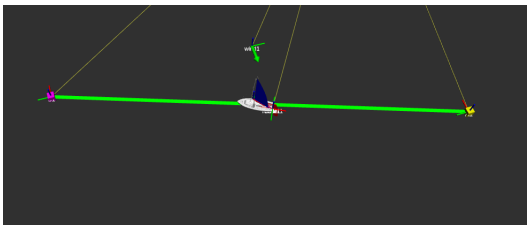

Figure 3.2: Screen Rviz

The 3D models of the hull, sail and rudder are available on my github [2]. This code also allows you to replay a mission. Indeed if the data of a test were recorded in a rosbag then it is possible to replay it with this node in addition, we can then visualize and analyze the behaviour of the boat during this test.

3.3 Use

Simulation is replacing the data that we would give the sensors. To use it you must launch the simulation node instead of the GPS code and arduino code and filters. In addition, controllers should be told to subscribe to simulated topics and not to real value topics. If the user wants to view his test then he can also launch the visualization node in parallel. By using the groups of a roslaunch, it is also possible to view several boats at the same time on Rviz.

4 Sailboat control

After the low level, I worked on the sailboat controllers.

4.1 Cap and line following

The first controller I implemented on the robot is a heading following one. This controller allows to check that the system is working properly. It may be useful to use it before launching to ensure the condition of the sailboat. I then implemented a line-tracking controller that runs the following fonction [4]:

```
def control(x,q):
    zeta = pi/4
    theta = x[2,0]
    m = array([ [x[0,0]],[x[1,0]]])
    e = det(hstack((b-a,m-a)))/norm(b-a)
    phi = arctan2(b[1,0]-a[1,0],b[0,0]-a[0,0])
    if (abs(e)>r):
        q = sign(e)
    thetabar = phi - arctan(e/r)
    if (cos(psi-thetabar)+cos(zeta)) < 0:
        thetabar = pi +psi-zeta*q

    deltar = (2/pi)*arctan(tan(0.5*(theta-thetabar)))
    deltamax = pi/4*(cos(psi-thetabar)+1)
    u = array([ [deltar],[deltamax]])
    return u,q
```

This controller takes in the robot state (position, heading, wind) as well as 2 points (a,b). It then returns a rudder control and a sail control. It is a basic controller but very interesting for our needs if we add a control on the line that the boat must follow.

4.2 Waypoint Following

In order to carry out the tests of the WRSC the line controller can be more than enough. Indeed the 4 events do not require more than reaching points in a precise order.

4.2.1 Functioning

So I decided to code a way to control the line performed by line tracking.

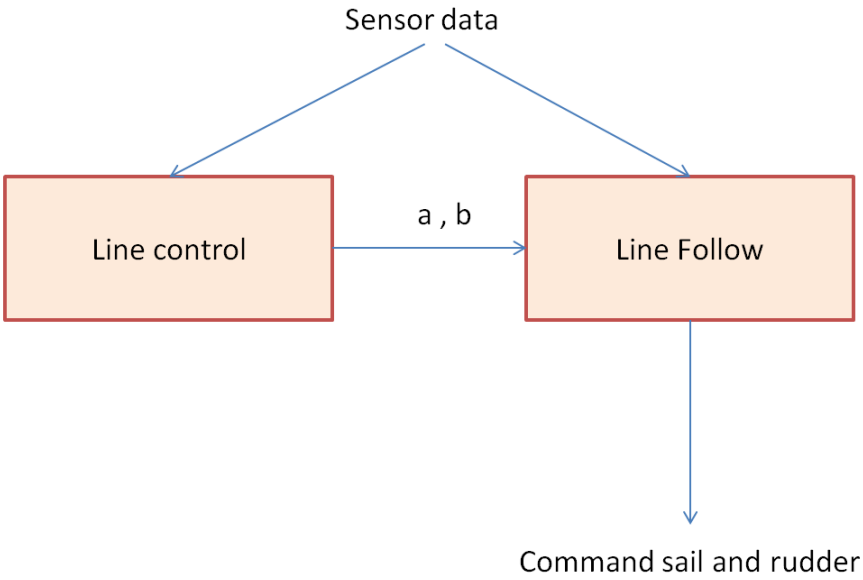

Figure 4.1: Controller

The 2 controllers receive the data from the sensors, then the 1st controller communicates the line to follow to the second. To use this controller, the user must create a mission file in this form.

```
ref:
29.867139,121.538975
############################
29.867139,121.538975,0
29.867054,121.538999,0
29.867208,121.598678,0
29.867295,121,538401,0
```

This is the file we used for the race on the first day of the competition.

The line control works in cartesian system, so the reference position is useful for the conversion of longitude and latitudes into cartesian system locally. For our tests we have taken the habit of fixing the reference position in the place where we put the boat in the water, so the display is easier to look at. But it can be anywhere as long as it's near the test area. Then the user enters the positions of the waypoints that he wishes to join in the order "latitude, longitude, time". The 3rd value, time, is used for the Station Keeping, which I would discuss in the next part.

The line controller will then retrieve these values and function as follows:

- Initially it creates a line between the starting point of the boat and the 1st waypoint.
- When it has reached its objective it creates a line between this point and the 2nd, and so on up to the last point.
- When the last point is reached it creates a line between the position of the boat and the last point. So the user easily notices when the boat completed the mission. One possible improvement would be to return the boat to the user; it depends on the needs of the user and is already possible if the user adds this waypoint in the mission file.

4.2.2 Interest of this solution

As I said earlier, this is an easy-to-use solution. The objective of my internship was to achieve a robust low-level complete system. So I spent less time on the controller, this solution can be implemented without having a lot of test: if the line following works then the mission works. In addition, this solution is very effective for the events that the WRSC offers. In fact, most of the trials require a line on a lake. The weak point of this technique is the realization of complicated trajectories, like circles for example, since it requires breaking it down into several lines, it is therefore less optimized.

4.3 Station keeping

The most complicated technical test for the sailing boat is the station keeping. This is the reason why I added in my mission file a 3rd column corresponding to the time in second that the boat must stay close to this point before moving on to the next one. If this time is different from 0 then the program perform a Keeping station around this point. I tested 5 different solutions with the constraint of making lines between 2 points in order to stay as close to the point as possible.

Solution 1

The first solution is to make a line between the positions of the boat whatever its course and the objective. This solution works well at the beginning of the manoeuvre. However after some change of tack the boat tends to converge in a position facing the wind without speed, which makes it uncontrollable, and drifty.

Solution 2

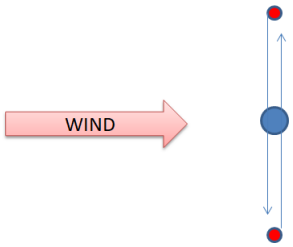

Figure 4.2: Station keeping 1

In the second solution, I make a line perpendicular to the wind. This solution allows for good results: for example, it is the one we retained for the competition and helped us achieving ninth over 23 teams. This solution may have been effective for our boat because we had good manoeuvrability and needed little speed to make a successful tack. Thus we were able to reduce the size of the line around 3.5 m without having any problem.

Solution 3

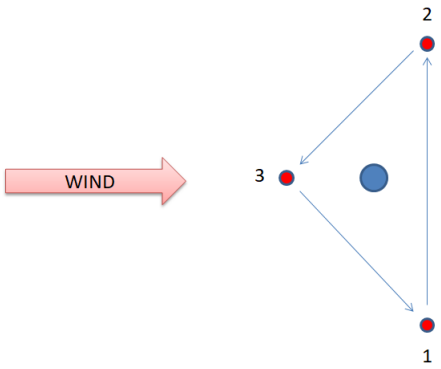

Figure 4.3: Station keeping 2

The 3rd solution consists in making a triangle around the point. I thought the distance between the boat and the point would be more consistent. This method has the advantage of not passing on point, and therefore can be used to turn around a real buoy. In the case of a virtual buoy the method is not interesting because it is the average of the distance that is retained by the referees, so crossing by as close as possible to the virtual buoy is more advantageous.

Solution 4

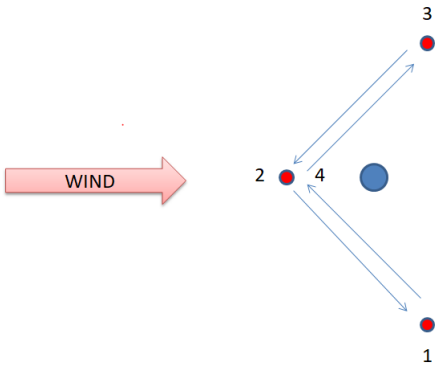

Figure 4.4: Station keeping 3

The fourth solution is close to the previous one. However, in the case of significant wind I wanted to find a way to break the speed of the boat to stay close to the point. That's why I removed the side perpendicular to the wind of the triangle, so I force the boat to turn almost in front of the wind and considerably reduces its speed. This also reduces the difference that could exist in the passage of line at the point 1 (figure 4.3) in the case of strong wind. Having very little wind on the day of the competition, we did not use this solution.

Solution 5

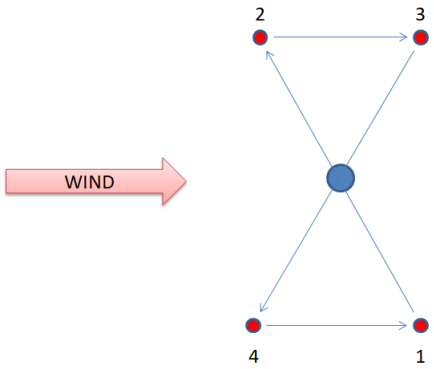

Figure 4.5: Station keeping 4

I thought of the last solution in case our boat has a lower manoeuvrability. This makes it possible to make wider turns and never risked to face the wind when changing the line. The idea is to achieve an «8», however it might be better to achieve this solution without making lines but with a controller allowing to make circles or ellipses.

4.4 Others idea

Outside the context of the competition, we also wanted to prove that our system could be used, for example, to control a fleet of boats thanks to Matthieu Bouveron's communications. That is why I also realized a simple controller allowing to follow another boat that would communicate its position to us. For this I have imagined 2 solutions:

- Draw a line between the 2 boats.
- Save the boat positions tracked at regular intervals and try to copy his trajectory.

These 2 solutions work on simulator however we did not have time to perform real tests. Plus we have no obstacle avoidance so the tests are risky.

5 Competition

The WRSC 2019 in China allowed us to conclude our internship with a real-life validation of our system. It also allowed us to compare our performance to other teams.

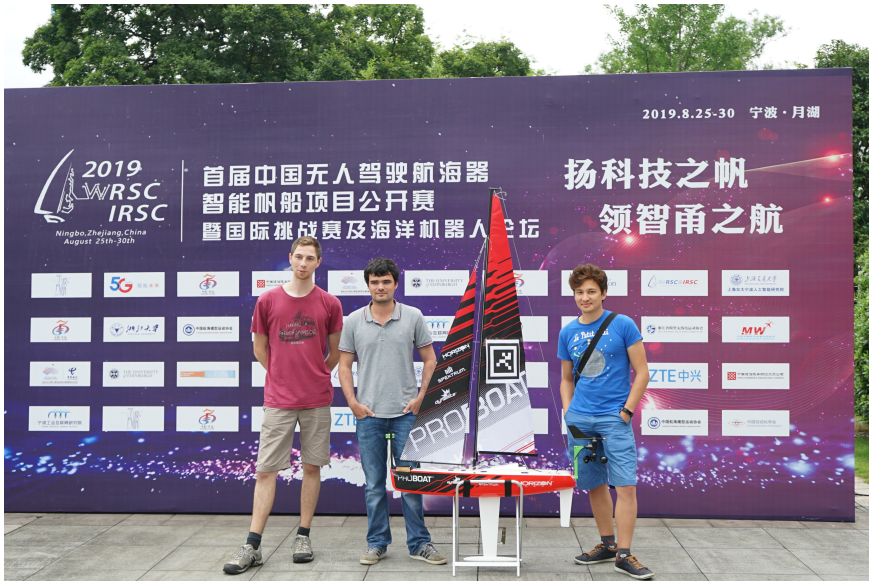

Figure 5.1: Plymouth team

I competed with Corentin Jegat and Ulysse Vautier for the plymouth team.

5.1 Strategy

The competition is divided into 4 events. We used the controller presented in the previous chapter for each of the tests. Our main objective was to validate each test and thus prove that our system is usable and robust.

5.1.1 Fleet race

The race is the simplest event, the aim is to reach 4 waypoints as fast as possible. We used the coordinates provided by the organisers to launch the mission. This allowed us to validate a time of 3 minutes and 28 seconds, knowing that the best time is 1 minute and

26 seconds and the least good time validated is 9 minutes and 56 seconds. 13 teams failed to validate the test.

5.1.2 Station keeping

The station keeping test is divided into 2 parts. The first part is to stay close to a given gps point. The second is to detect a real buoy and stay close. We used strategies 2 and 3 explained in the previous chapter and obtained the 9th place on the test.

5.1.3 Area scanning

Area scanning is an exploration test of the lake. Each team has 15 minutes to explore a defined area separated down into brick. In each group of 7 teams, the first to explore a brick wins 100% of the points, the second 50% etc... To carry out this mission we prepared the mission file to scan as much area as possible in 15 mins, starting in the 1st. We got the 10th place on this test. Here is a plan of our strategy

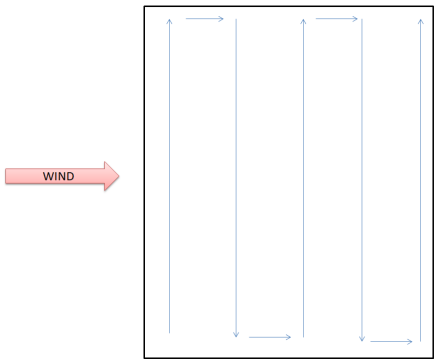

Figure 5.2: strategy area scanning

5.1.4 Hide and seek

Hide and seek is a paired event. Each team has an April tag on each side of their sail and makes lines between 2 points.

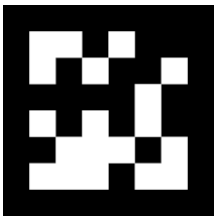

Figure 5.3: April tag

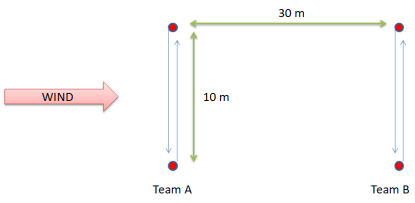

Figure 5.4: Hide and Seek

Each boat marks 1 point per line made and 3 more if it photographs and identifies the code of the opposing team. Despite a working code, our camera did not have a high enough resolution to identify the opposing code from far enough away. We finished 10th on the test.

5.2 Feedback

We finally obtained the 7th place in the overall ranking. In addition, we have validated each event, which is not the case for many teams. This was mainly due to the fact that our system seemed simpler to put in place and more robust, even if the price was much lower. So we were able to prove that our system could be used for research on controllers, for example, therefore we are proud of that. It is nevertheless possible to make improvements, especially with a better wind sensor. The one we had was not sensitive enough for very weak winds and induced undesirable behaviours such as unnecessary tacking during trials.

Conclusion

To conclude, this internship was very rewarding. It allowed me to put in place in a very concrete case all that we had studied in robotics during this 2nd year, and to end up with a competition in China to validate our work.

At the end of this course I managed to program reliable and adaptable system with a set of sensors and actuator. In fact to equip a new boat it is enough to update the parameters of the servomotors and the sensors. I also created a simulation and a controller that could be used for a competition like the WRSC. Generally speaking, we have proven that the system we have designed is robust and usable for research. Moreover with the communications in place it can be usable to control a fleet of ships for example.

Working on a topic of marine robotics has allowed me to become aware of specific constraints such as the essential waterproof side. This is also an area where real tests are complicated to implement, hence the importance of being able to build efficient simulations. I advise this internship to people showing motivation about work on a sailboat.

Bibliography

[1] Matthieu Bouveron. *Github repository.* URL: https://github.com/Matthix7/plymouth_internship_2019.

[2] Alexandre Courjaud. *Github repository.* URL: https://github.com/AlexandreCourjaud/Stage2APlymouth.

[3] Corentin Jegat. *Github repository.* URL: https://github.com/corentin-j/WRSC_plymouth_JEGAT.

[4] Jaulin L. "Robmooc". In: (Mar. 2019). URL: https://www.ensta-bretagne.fr/jaulin/robmooc.pdf.

[5] Ulysse Vautier. *Github Plymouth Sailboat.* URL: https://github.com/Plymouth-Sailboat.

A

Appendix

ENSTA Bretagne

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* University of plymouth

Adresse / *Address* Drake Circus, plymouth, Devon PL48AA UK

Tél / *Phone (including country and area code)* +44 01752 586157

Nom du superviseur / *Name of internship supervisor* Jian Wan

Fonction / *Function* Lecturer in control systems Engineering

Adresse e-mail / *E-mail address* jian.wan@plymouth.ac.uk

Nom du stagiaire accueilli / *Name of intern* COURJAUD Alexandre

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 ? *Was the initial contract carried out to your satisfaction?* **A** B C D E F
- Manquait-il au stagiaire des connaissances ? *Was the intern lacking skills?* ☐ oui/*yes* ☑ non/*no*

  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 Plymouth, on 30/08/2019

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

School of Engineering
University of Plymouth
Drake Circus
Plymouth
Devon
PL4 8AA

Courjaud

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