Rapport de stage

Développement d'un voilier autonome

Catherine Rizk

ROB 24

Je tiens à remercier mon tuteur de stage Dr. Jian Wan pour son soutien et sa bienveillance pendant la totalité du stage ainsi que mes deux camarades de stage, Ludovic Mustiere et Harendra Rangaradjou pour leur aide et leur bonne humeur au quotidien.

Table des matières

Table des figures

Abstract

I did my internship at Aston University in Birmingham, England. The goal was to make a sailing boat autonomous by using the various sensors available. The sailboat had to be able of carrying out several tasks, such as moving in a straight line or reaching a waypoint and staying there.

The first part of the course consisted of discovering and familiarising ourselves with a set of sensors in order to develop various computer programs for calibrate and then use them and, if necessary, filter them. The rest of the course was based on the implementation of control algorithms to enable the boat to carry out its missions. These algorithms were then tested using a simulation that initially ran on a simple terminal, and for which we developed a more complex graphical interface to make it easier for the user to use and simplify access to the various simulation parameters. Finally, we were able to carry out real-life tests on a lake in the south of Birmingham, after which we adjusted the code and various parameters.

Résumé

J'ai réalisé mon stage dans l'Universté d'Aston à Birmingham en Angleterre. Le but de ce dernier était d'autonomiser un voilier en utilisant les différents capteurs mis à notre disposition. Le voilier devait ainsi être capable d'effectuer plusieurs missions telles qu'un déplacement en ligne droite ou atteindre un point de repère et y rester.

La première partie de ce stage consistait à découvrir et à se familiariser avec un ensemble de capteurs afin de développer divers programmes permettant leur calibration puis leur usage et si nécessaire leur filtrage. La suite du stage reposait sur l'implémentation d'algorithmes de contrôle pour permettre au bateau de réaliser ses missions. Ces algorithmes ont ensuite été testés à l'aide d'une simulation qui fonctionnait tout d'abord dans un simple terminal puis pour laquelle on développait une interface graphique plus complexe pour permettre à l'utilisateur un usage plus facile et un accès simplifié aux différents paramètres de la simulation. Enfin, nous avons pu réaliser des tests en conditions réelles sur un lac au sud de Birmingham à la suite desquels nous ajustions le code et divers paramètres.

Mots-Clés

Voilier, contrôle, autonomie, simulation, capteurs

1 Introduction

1.1 Présentation de l'Université d'Aston

J'ai réalisé mon stage dans la ville de Birmingham en Angleterre. Il s'agit d'une ville de plus d'un million d'habitants située dans les Midlands de l'ouest [1]. Les plus grandes universités de Birmingham sont l'Université de Birmingham et l'Université d'Aston. C'est dans cette dernière que j'ai effectué mon stage.

Figure 1 – Carte situant Birmingham en Angleterre

En 2022, elle décroche la 25ème place du classement des meilleurs universités britanniques publié par The Guardian et regroupant un total de 121 universités. [2]

Le campus s'étend sur 24 hectares et compte de nombreux bâtiments consacrés à différents domaines d'études (technologie, business, médical, art...). J'ai travaillé pendant l'intégralité de mon stage dans le bâtiment principal ("main builiding") qui compte de multiples salle de cours et espaces de travail. [3]

Figure 2 – Bâtiment Principal de l'Université d'Aston

Figure 3 – Campus de l'Université d'Aston, photographie provenant de [4]

1.2 Les bateaux à voiles

Le voilier constitue l'un des premiers moyens de locomotion permettant de voyager sur d'importantes distances. Il est utilisé dès l'Antiquité notamment pour le commerce et se verra peu à peu remplacé par le bateau à vapeur lors de la Révolution industrielle au XIXème siècle.

L'architecture du voilier repose sur une coque, un mât et des voiles qui servent à la propulsion. Il s'agit donc d'un modèle initialement assez simple qu'il est de nos jours possible de sophistiquer à l'aide de l'informatique embarquée. On dispose d'un voilier accompagné de différents capteurs. Le but de ce stage est de rendre ce voilier autonome afin qu'il soit en mesure d'effectuer diverses missions. Nous verrons en quoi ce simple modèle de voilier constitue un support pédagogique pour le développement de codes informatiques et pour la compréhension de ses composants mécaniques ainsi que de différents capteurs dans le but de rendre ce voilier autonome.

2 Le voilier, ses composants et ses capteurs

2.1 Le voilier

Je disposais d'un voilier de la marque ProBoat mesurant 195,5cm de haut et 1m de large. Celui-ci est composé d'une coque contenant un espace dédié à la carte raspberry que je rangeais à l'intérieur, de deux voiles pouvant être pilotées par un servomoteur, d'un gouvernail, lui aussi piloté par un servomoteur et d'une quille.

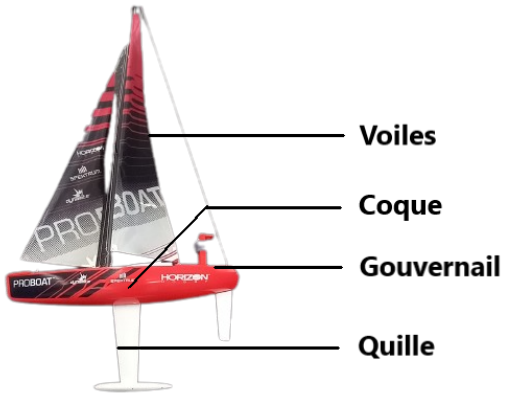

FIGURE 4 – Voilier utilisé

Plusieurs structures avaient été préalablement fabriquées à l'aide d'une imprimante 3d et permettaient de poser la carte à l'intérieur de la coque de manière stable et de fixer différents composants sur le bateau tels qu'un GNSS et un anénomètre. Nous détaillons l'usage de ces capteurs dans les paragraphes ci-dessous.

FIGURE 5 – Structure utilisée pour poser la carte raspberry

FIGURE 6 – Structure utilisée pour fixer l'anénomètre et le GNSS

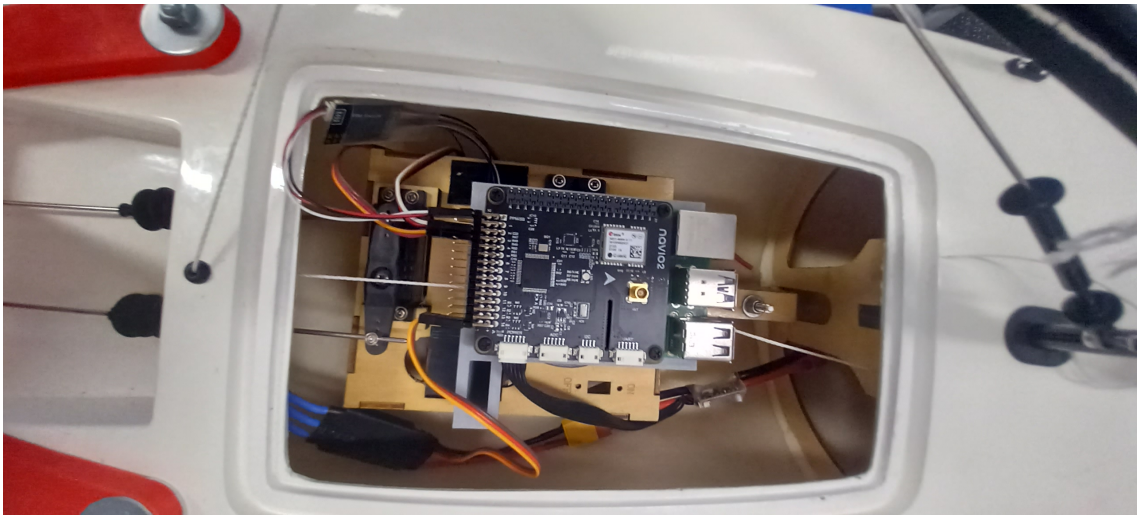

FIGURE 7 – Photographie de la carte Raspberry sur son support dans la coque du bateau

2.2 La carte

Je disposais aussi d'une carte Raspberry Pi 4 à laquelle venait se superposer une carte Navio 2 qui se fixait à l'aide de petites vis. L'ajout de cette carte permet d'étendre les capacités de la Raspberry Pi notamment grâce au port MCX de la Navio auquel il est possible de brancher une antenne GNSS et grâce ses deux IMU intégrés [5] dont nous décrivons l'utilisation par la suite. J'ai codé en python et les codes écrits ont pu être copiés sur la carte via une connexion en ssh. Tout mes codes sont disponibles sur le git en collaboration avec mes camarades de stage. [6]

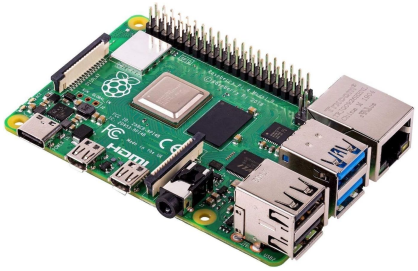

FIGURE 8 – Carte Raspberry Pi 4

FIGURE 9 – Carte Navio fixée sur la Raspberry Pi

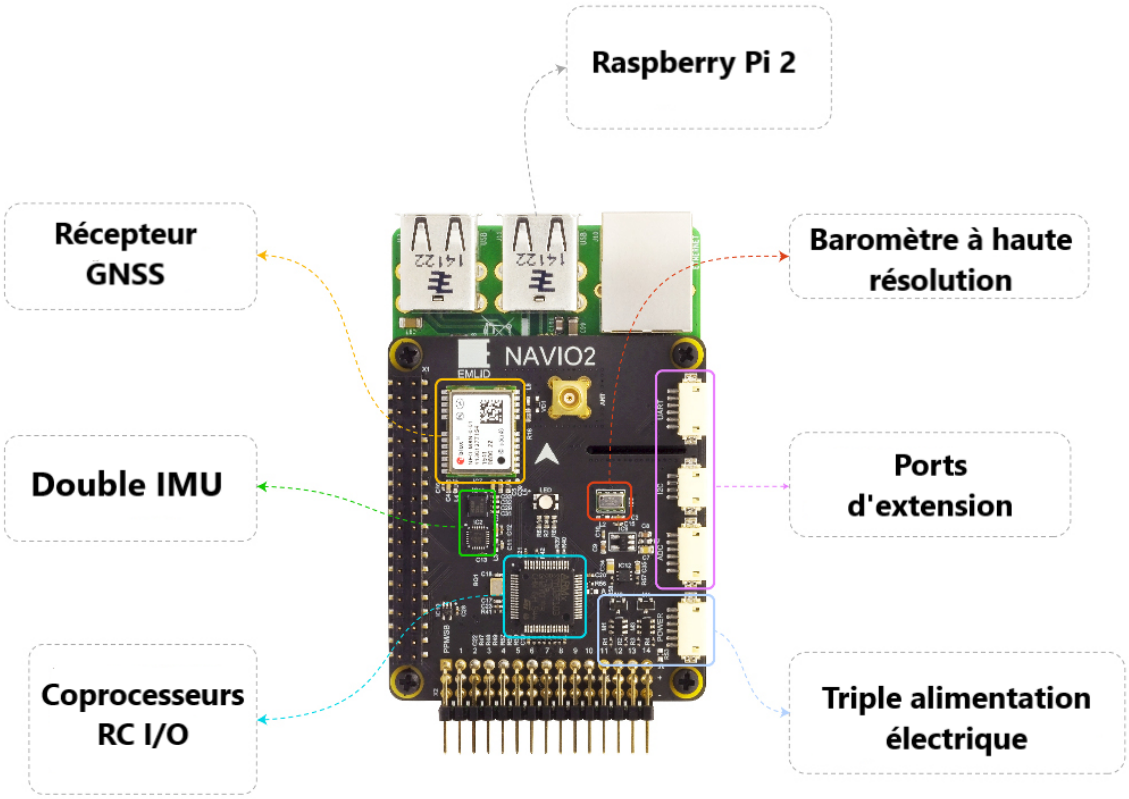

FIGURE 10 – Détail de l'architecture d'une carte Navio, image provenant de [7]

2.3 L'anémomètre

J'avais également un anémomètre qui est un capteur détectant la vitesse et la direction du vent. Le modèle dont je disposais a été fabriqué par la marque Calypso Instruments et est un modèle sans fil Bluetooth alimenté à l'aide de panneaux solaires.[8]

FIGURE 11 – Anémomètre Calypso Instruments

L'anémomètre était installé à l'aide d'une vis qui se fixait dans le support 3d présenté précédemment figurant à l'arrière du bateau.

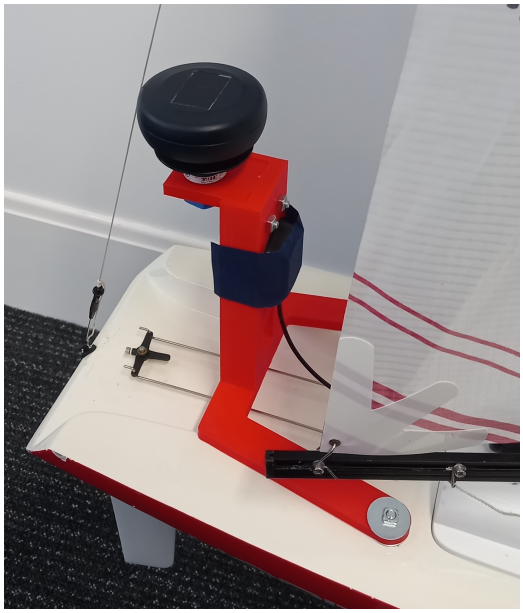

FIGURE 12 – Anémomètre fixé sur son support

Une fois fixé, l'anémomètre était connecté à la carte Raspberry par Bluetooth. J'ai utilisé une librairie python nommée "calypso-anemometer" proposant de nombreuses fonctions permettant la lecture des données captées par l'anémomètre. J'utilisais ensuite les valeurs lues pour obtenir la vitesse du vent et sa direction.

Le vent capté par l'anémomètre ne correspond cependant pas au vent réel, c'est ce qu'on appelle le "vent apparent". Par opposition au vent réel qui correspond au vent météorologique qui serait ressenti par un objet à l'arrêt, le vent apparent est le vent ressenti par un corps en mouvement. Lors de son déplacement, celui-ci est en effet soumis à un vent dit "relatif" engendré par la vitesse de son mouvement. Le vent apparent correspond donc à la somme vectorielle du vent réel et du vent relatif.

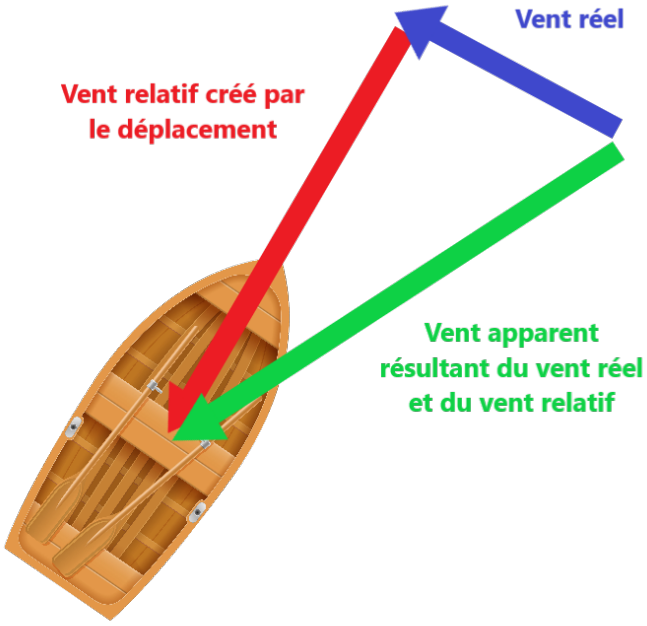

FIGURE 13 – Schéma représentant les vecteurs du vent réel, du vent relatif et du vent apparent soufflant sur un bateau

Pour déterminer le vent réel à partir des données captées par l'anémomètre, on doit donc effectuer le calcul suivant : (expression établie en s'appuyant sur la référence [9])

$$\vec{\mathbf{wind}} = \vec{\mathbf{w_{ap}}} - \vec{\mathbf{v_{relatif}}} \tag{1}$$

Avec :
— $\vec{\mathbf{wind}}$, le vecteur du vent réel,
— $\vec{\mathbf{w_{ap}}}$, le vecteur du vent apparent,
— $\vec{\mathbf{v}}_{\text{relatif}}$, le vecteur du vent relatif ($= -\vec{\mathbf{vitesse}}_{\text{bateau}}$),

Il reste encore à construire le vecteur $\vec{\mathbf{w_{ap}}}$ à partir des données dont on dispose. On capte le heading θ qui correspond au cap du bateau, la direction du vent ψ_{ap} et la vitesse du vent a_{ap}, on a donc : (expression établie en s'appuyant sur la référence [10])

$$\vec{\mathbf{w}}_{\text{ap}} = a_{\text{ap}} \cdot \begin{bmatrix} \cos(\psi_{\text{ap}} + \theta) \\ \sin(\psi_{\text{ap}} + \theta) \end{bmatrix} \tag{2}$$

2.4 L'IMU

Je disposais de 2 IMU directement intégrés dans la Navio : l'IMU MPU9250 et l'IMU LSM9DS1. Ces deux IMU permettent d'obtenir des valeurs du gyroscope, du magnétomètre et de l'accélèromètre. Ces valeurs sont ensuite utilisées pour déterminer le "picth", le "roll" et le "heading" qui représentent en français le tangage, le roulis et le cap.

J'obtenais alors avec chaque IMU un vecteur pour chacun de ces capteurs et réalisais une moyenne des valeurs

obtenues avec chaque IMU pour avoir un vecteur accélération, un vecteur du champ magnétique et un vecteur décrivant la vitesse angulaire de la forme :

$$\vec{\mathbf{Acc}} = \begin{bmatrix} Acc_1 \\ Acc_2 \\ Acc_3 \end{bmatrix} \qquad \vec{\mathbf{Mag}} = \begin{bmatrix} Mag_1 \\ Mag_2 \\ Mag_3 \end{bmatrix} \qquad \vec{\mathbf{Gyr}} = \begin{bmatrix} Gyr_1 \\ Gyr_2 \\ Gyr_3 \end{bmatrix}$$

Je n'utilisais cependant pas les valeurs du gyroscopes car elles n'interviennent pas dans le calcul du pitch, du roll et du cap. Avant de pouvoir utiliser ces valeurs, j'ai du procéder à une calibration de l'IMU.

Pour calibrer l'accéléromètre, on utilise une ressource fournie [11] pendant notre stage proposant une méthode de calibration. On cherche donc à déterminer une matrice de calibration X telle que :

$$Y = w \cdot X \tag{3}$$

Avec Y le vecteur accélération calibré, w le vecteur accélération non calibré et X la matrice de calibration. Pour se faire, on place l'IMU dans des positions pour lesquelles le vecteur Y est connu (des vecteurs unitaires Y_1, Y_2, Y_3, Y_4, Y_5, Y_6) et on note les valeurs lues Acc_{x1}, Acc_{y1} et Acc_{z1} pour chaque Y qu'on stocke ensuite dans un vecteur de la forme :

$$\vec{\mathbf{w1}} = \begin{bmatrix} Acc_{x1} & Acc_{y1} & Acc_{z1} & 1 \end{bmatrix}$$

On stocke ensuite tous ces vecteurs dans des matrices :

$$\mathbf{Y} = \begin{bmatrix} Y1 \\ Y2 \\ Y3 \\ Y4 \\ Y5 \\ Y6 \end{bmatrix} \qquad \mathbf{w} = \begin{bmatrix} w1 \\ w2 \\ w3 \\ w4 \\ w5 \\ w6 \end{bmatrix}$$

On obtient alors la matrice X en utilisant la méthode des moindres-carrés :

$$X = [w^T \cdot w]^{-1} \cdot w^T \cdot Y \tag{4}$$

Il faut ensuite calibrer le magnétomètre. Avant calibration, en représentant les données captées par le magnétomètre en le faisant tourner sur lui-même, on obtient une ellipse ce qui signifie que les données sont déformées. Remédions à cette incohérence en s'appuyant encore sur la ressource [11]. On a l'équation suivante :

$$\frac{(x - x_0)^2}{a^2} + \frac{(y - y_0)^2}{b^2} + \frac{(z - z_0)^2}{c^2} = R^2 \tag{5}$$

Avec :

— x0, y0, z0 les décalages causés par les déformations sur les 3 axes (x, y, z),
— x, y, z les données mesurées par le capteur avant calibration,
— a, b, c les longueurs des demi-axes des ellipses,
— R une constante de l'intensité du champ magnétique terrestre,

Ce qui peut être réécrit sous la forme de l'équation matricielle suivante :

$$x^2 = \begin{bmatrix} x & y & z & -y^2 & -z^2 & 1 \end{bmatrix} \cdot \begin{bmatrix} 2x_0 \\ \frac{a^2}{b^2} \cdot 2y_0 \\ \frac{a^2}{c^2} \cdot 2z_0 \\ \frac{a^2}{b^2} \\ \frac{a^2}{c^2} \\ a^2R^2 - x_0^2 - \frac{a^2}{b^2}y_0^2 - \frac{a^2}{c^2}z_0^2 \end{bmatrix}$$

Ce qui se ramène à une équation matricielle du type :

$$w = H \cdot X \tag{6}$$

A laquelle on peut appliquer la méthode des moindres-carrés comme précedemment ce qui donne :

$$X = [H^T \cdot H]^{-1} \cdot H^T \cdot w \tag{7}$$

A partir du vecteur X obtenu, on a alors :

$$x_0 = \frac{X[1]}{2} \tag{8}$$
$$y_0 = \frac{X[2]}{2 \cdot X[4]} \tag{9}$$
$$z_0 = \frac{X[3]}{2 \cdot X[5]} \tag{10}$$
$$A = X[6] + x_0^2 + X[4] \cdot y_0^2 + X[5] \cdot z_0^2 \tag{11}$$
$$B = \frac{A}{X[4]} \tag{12}$$
$$C = \frac{A}{X[5]} \tag{13}$$

Avec X[1], X[2], X[3], X[4], X[5], X[6] les 6 coordonnées du vecteur X.

On pose ainsi :

$$MSC_x = \sqrt{A} \tag{14}$$
$$MSC_y = \sqrt{B} \tag{15}$$
$$MSC_z = \sqrt{C} \tag{16}$$
$$\tag{17}$$

On a alors les matrices :

$$Offsets = \begin{bmatrix} x_0 & y_0 & z_0 \end{bmatrix}$$

$$Scales = \begin{bmatrix} MSC_x & MSC_y & MSC_z \end{bmatrix}$$

On mesure ensuite les valeurs non calibrées du magnétomètre qu'on nomme $Mag_{raw}[1]$, $Mag_{raw}[2]$ et $Mag_{raw}[3]$ stockées dans le vecteur Mag_{raw}. Les valeurs calibrées sont donc obtenues en soustrayant l'offset à chaque coordonnée du vecteur Mag_{raw} et en divisant par la coordonnée de la matrice Scales comme ci-dessous :

$$\mathbf{Mag} = \begin{bmatrix} \dfrac{Mag_{\text{raw}}[1] - \text{Offsets}[1]}{\text{Scales}[1]} \\ \dfrac{Mag_{\text{raw}}[2] - \text{Offsets}[2]}{\text{Scales}[2]} \\ \dfrac{Mag_{\text{raw}}[3] - \text{Offsets}[3]}{\text{Scales}[3]} \end{bmatrix}$$

Afin de vérifier que cette calibration est effective, nous avons recueilli et représenté graphiquement les données lues par le magnétomètre avant et après calibration en le faisant tourner autour d'un premier axe (données en rouge), puis d'un second axe (données en vert) et enfin autour d'un dernier axe (données en jaune). On fait ensuite effectuer des rotations à l'IMU autour de tous les axes (données en bleues). Après calibration, les ellipses doivent être devenues des cercles centrés en 0 et l'ellipsoïde en bleu doit être devenue une sphère elle aussi centrée en 0.

(a) Données recueillies par le magnétomètre avant calibration

(b) Données recueillies par le magnétomètre après calibration

FIGURE 14 – Représentation graphique des données lues par le magnétomètre en effectuant des rotations dans l'espace

Les valeurs obtenues ne sont pas parfaites mais on peut constater que les ellipses et l'ellipsoïde ressemblent davatange à des cercles et une sphère et sont davantage centrés en 0 avec un rayon de 1. Cette calibration, bien qu'imparfaite, permet donc d'obtenir des valeurs beaucoup plus cohérentes qu'initialement.

On peut désormais calculer le pitch, le roll et le heading :

$$pitch = arcsin(Acc_x) \tag{18}$$

Si $cos(pitch) <= 0$

$$roll = arcsin(-Acc_x/cos(pitch)) \tag{19}$$

Sinon :

$$roll = arcsin(Acc_x/cos(pitch)) \tag{20}$$

Pour calculer le heading on calcule m_x et m_y tels que :

$$m_x = Mag_x * cos(pitch) + Mag_z * sin(pitch) \tag{21}$$

$$m_y = Mag_x * sin(roll) * sin(pitch) + Mag_y * cos(roll) - Mag_z * sin(roll) * cos(pitch) \tag{22}$$

On a alors,
Si $m_x > 0$ et $m_y > 0$:

$$heading = arctan(m_y/m_x) \tag{23}$$

Si $m_x < 0$:

$$heading = \pi + arctan(m_y/m_x) \tag{24}$$

Si $m_x > 0$ et $m_y <= 0$:

$$heading = 2 * \pi + arctan(m_y/m_x) \tag{25}$$

Si $m_x = 0$ et $m_y <= 0$:

$$heading = \pi/2 \tag{26}$$

Si $m_x = 0$ et $m_y > 0$:

$$heading = 3 * \pi/2 \tag{27}$$

Avec Mag_x, Mag_y, Mag_z, Acc_x, Acc_y et Acc_z les composantes des vecteurs du champ magnétique et de l'accélération. On nomme ce heading θ dans la suite du rapport.

2.5 Le GNSS

Les valeurs du GNSS sont lues à l'aide du capteur UBLOX intégré à la carte Navio auquel on relie notre GNSS. On lit à travers le message "NAV POSLLH" la latitude, la longitude et l'altitude, à travers le message "NAV VELNED" le vecteur vitesse et le heading et à travers le message "NAV TIMEUTC" la date et l'heure.

Les valeurs de latitude et longitude sont ensuite projetées selon une projection UTM (Universal Transverse Mercator) pour obtenir des coordonnées cartésiennes.

2.6 Les servomoteurs

Le bateau est équipé de deux servomoteurs, l'un qui commande le gouvernail et un second commandant les voiles (les deux voiles reçoivent la même commande et bougent simultanément du même angle). Les servomoteurs fonctionnenent à l'aide de signaux PWM qui leur sont envoyés générant ainsi un mouvement d'un certain angle. Le servomoteur contrôlant le gouvernail peut bouger de 180° tandis que celui qui contrôle les voiles peut bouger de 90°. Il était nécessaire d'effectuer une calibration afin que le gouvernail puisse se mouvoir d'un angle entre -45° et 45+ et les voiles d'un angle de 0° à 90°.

Pour effectuer cette calibration, il faut déterminer la largeur d'impulsion pour un angle de 0° et la largeur d'impulsion pour un angle de 90°. On a alors la relation suivante pour atteindre un angle α désiré :

$$Commande(\alpha) = SERVO_{min} + \frac{SERVO_{max} - SERVO_{min}}{ANGLE_{max} - ANGLE_{min}} \cdot (\alpha - ANGLE_{min}) \tag{28}$$

2.7 La télécommande

Afin de s'assurer de pouvoir récupérer le bateau en cas de soucis, je disposais également d'une télécommande grâce à laquelle il était possible de commander le bateau.

FIGURE 15 – Télécommande utilisée

3 Algorithmes de contrôle

Le but final de ce projet est d'autonomiser notre voilier. Les capteurs présentés précedemment nous permettent de connaître les valeurs de nombreuses données à prendre en compte : vitesse et direction du vent, orientation du bateau, position et vitesse du bateau... J'ai par la suite implémenté des algorithmes afin de définir à partir de ces données les commandes (angle de la voile et angle du gouvernail) à donner au bateau pour qu'il parvienne à atteindre son objectif. On définit 3 objectifs pour le bateau :

— Effectuer une ligne droite
— Atteindre un point de repère
— Naviguer entre les différents points de repère (en atteindre un premier puis se diriger vers le suivant, etc.)

Pour atteindre ces différents objectifs, j'ai implémenté trois types d'algorithmes :

— Algorithmes de supervision : algorithmes qui visent à établir le but à atteindre, ils supervisent de manière globale le bateau.
— Algorithmes de controle : algorithmes qui visent à réaliser le but fixé par l'algorithme de supervision en agissant concrètement et précisement sur les actionneurs du bateau c'est-à-dire les servomoteurs.
— Algorithmes d'observation : algorithmes qui visent à prédire les variables d'état du bateau à l'aide des valeurs mesurées par les capteurs.

3.1 Algorithmes de supervision

J'ai défini trois algorithmes de supervision pour chacun des trois objectifs qu'on désire atteindre :

— "Supervisor Line Following Default" : a pour but de faire réaliser une ligne droite au bateau. Pour cela, on utilise deux points d'une liste de points de repère préalablement définie. Le bateau devra suivre la ligne qui relie ces deux points.
— "Supervisor Station Keeping Default" : a pour but de faire parvenir le bateau à un point de la liste de points de repère et de faire en sorte qu'il y reste. Pour se faire, on trace deux cercles de deux diamètres différents autour du point à atteindre. Ces cercles seront utiles pour le contrôlleur. On considère que le bateau a atteint le point lorsqu'il se situe à l'intérieur du cercle de plus petit diamètre.
— "Supervisor Path Following Default" : a pour but de faire parvenir le bateau d'un point de repère à un autre. Le fonctionnement de ce superviseur est similaire au précédent mais le bateau doit cette fois-ci continuer de se déplacer vers un nouveau point une fois que le premier point a été atteint et continue ainsi avec les différents points de la liste.

3.2 Algorithmes de controle

J'ai implémenté deux algorithmes de controle : un algorithme permettant de suivre une ligne et un algorithme permettant d'atteindre un point de repère. Ces algorithmes permettent de réaliser les tâches requises par les superviseurs. Ils utilisent différentes variables relatives au bateau définies ci-dessous :

— $\vec{\mathbf{X}} = \begin{bmatrix} x \\ y \\ \theta \\ v \\ w \end{bmatrix}$ le vecteur d'état avec x et y la position du bateau, θ le heading (ou cap), v la vitesse, w la vitesse angulaire,

— $\vec{\mathbf{pos}} = \begin{bmatrix} x \\ y \end{bmatrix}$ le vecteur contenant les coordonnées du bateau,

— $\vec{\mathbf{P_a}}$, le vecteur contenant les coordonnées du point à atteindre,

— $\vec{\mathbf{P_d}}$, le vecteur contenant les coordonnées du point de départ,

— $\vec{\mathbf{DA}} = \vec{\mathbf{P_a}} - \vec{\mathbf{P_d}}$, le vecteur entre le point de départ et le point à atteindre,

— $\vec{\mathbf{b_m}} = \vec{\mathbf{P_a}} - \vec{\mathbf{pos}}$, le vecteur entre la position du bateau et le point à atteindre,

— $\mathbf{d} = \left\| \vec{\mathbf{b_m}} \right\|$, la distance entre le bateau et le point de repère,

— $\alpha = arg(\vec{\mathbf{b_m}})$, l'angle du vecteur $\vec{\mathbf{b_m}}$,

— $\mathbf{a} = \left\| \vec{\mathbf{DA}} \right\|$, la distance entre le point de départ et le point à atteindre,

— $\vec{\omega}$, le vecteur vent,

— $\beta = arg(\vec{\omega})$, l'angle du vecteur vent,

— $\omega_\mathbf{n} = \|\vec{\omega}|$, la norme du vecteur vent,

— $\vec{\mathbf{w_{ap}}} = \begin{bmatrix} \omega_\mathbf{n} - cos(\beta - \theta) - v \\ \omega_\mathbf{n} - sin(\beta - \theta) \end{bmatrix}$ le vecteur du vent apparent,

— $\psi_\mathbf{ap} = arg(\vec{\mathbf{w_{ap}}})$, l'angle du vent apparent,

— ξ, le "close-hauled angle", angle entre la trajectoire du voilier et la direction du vent lorsque le voilier navigue au plus près du vent,

— γ, angle d'incidence,

— $\mathbf{r}$, distance de coupure, distance maximale entre le voilier et la ligne,

— $\mathbf{d_o}$, le diamètre du cercle extérieur autour du point à atteindre,

— $\mathbf{d_i}$, le diamètre du cercle intérieur autour du point à atteindre,

— $\mathbf{delta_r}$, l'angle du gouvernail,

— $\mathbf{delta_s}$, l'angle de la voile,

— $\mathbf{delta_{rmax}}$, l'angle maximal du gouvernail,

— $\mathbf{delta_{smax}}$, l'angle maximal de la voile,

3.2.1 Algorithme "Line Following"

L'algorithme de "Line Following" consiste à suivre la ligne droite formée par le point de départ du bateau et un des points de repère. Il repose sur l'algorithme de L. Jaulin [12] :

Algorithme de "Line Following" : déplacement en ligne droite
Variables d'entrée : $\vec{\mathbf{X}}$ (x, y, θ, v et w), $\vec{\omega}$, $\vec{\mathbf{P}}$ (on dispose donc aussi de $\mathbf{p\vec{o}s}$, de $\mathbf{\vec{bm}}$, de α, de $\mathbf{\vec{w_{ap}}}$, de $\vec{\psi_{\mathbf{ap}}}$) Variables de sortie : $\mathbf{delta_r}$, $\mathbf{delta_s}$
On détermine la distance du robot à la ligne : $e = \det\left(\text{concatenate}\left(\frac{\mathbf{\vec{DA}}}{a}, \mathbf{p\vec{o}s} - \mathbf{\vec{P_d}}\right)\right)$ *Et on détermine l'angle à suivre :* $angle_{nom} = \alpha - 2 \cdot \gamma \cdot \frac{\arctan(e,r)}{\pi}$
Si la valeur absolue de e est plus grande que r/25, le bateau est éloigné de la ligne. On change alors la valeur d'une variable qu'on nomme "hysteresis" qui servira dans les calculs ci-dessous. Si abs(e) $> r/2$: hysteresis = sign(e) Si $\cos(\beta - angle_{\text{nom}}) + \cos(\xi) < 0$ ou ($\lvert e\rvert < r$ et $\cos(\beta - \alpha) + \cos(\xi) < 0$) : *L'angle nominal à suivre correspond à une direction bien trop près du vent ce qui rend difficile pour le bateau de la suivre, on passe donc en mode "close-hauled angle", mode du voilier qui navigue près du vent :* $angle_{\text{actuel}} = \pi + \beta - hysteresis \cdot \xi$ Sinon : $angle_{\text{actuel}} = angle_{\text{nom}}$ Si $\cos(\theta - \text{angle}_{\text{actuel}}) \geq 0$: $\delta_r = \delta_{\text{rmax}} \cdot \sin(\theta - \text{angle}_{\text{actuel}})$ Sinon : *(Si la direction du robot n'est pas cohérente, on ouvre le gouvernail au maximum.)* $\delta_r = \delta_{\text{rmax}} \cdot \text{sign}(\sin(\theta - \text{angle}_{\text{actuel}}))$ *On obtient les angles suivant pour les voiles :* $\delta_{smax} = \frac{\pi}{2} \cdot \left(\frac{\cos(\beta - \text{angle}_{\text{actuel}}) + 1}{2}\right)$ $\delta_s = -sign(\psi_{ap}) * min(abs(\pi - abs(\psi_{ap})), \delta_{smax})$

3.2.2 Algorithme "Station Keeping"

L'algorithme de "Station Keeping" consiste à atteindre un point de repère et à rester au niveau de celui-ci. Pour cela, on trace deux cercles autour du point de repère et on construit notre algorithme en fonction de la distance du bateau à ces cercles. Le programme se divise en trois parties, la première a lieu lorsque le bateau est trop éloigné du point de repère, il suit alors une ligne droite allant de son point de départ au point de repère d'après l'algorithme précédent. La deuxième partie est enclenché lorsque le bateau pénètre dans la zone délimitée par le cercle de plus grand diamètre. La dernière partie a lieu lorsque le bateau est dans la zone du cercle intérieur. Il tente alors de rester dans cette zone. Cet algorithme a été écrit à partir d'un code en MatLab dont disposait déjà notre tuteur, le code implémenté est donc une retranscription en python d'un code déjà existant.

Algorithme de "Station Keeping" : déplacement jusqu'à un point de repère

Variables d'entrée : $\vec{\mathbf{X}}$ (x, y, θ, v et w), $\vec{\omega}$, $\vec{\mathbf{P}}$ (on dispose donc aussi de $\vec{\mathbf{pos}}$, de $\vec{\mathbf{bm}}$, de α, de $\vec{\mathbf{w_{ap}}}$, de $\vec{\psi_{\mathbf{ap}}}$)
Variables de sortie : $\mathbf{delta_r}$, $\mathbf{delta_s}$

Si d $> d_o$: *Si le bateau, n'est pas dans un des cercles autour du point,*
    Appliquer l'algorithme "Line Following"

Sinon :
    Si d $> d_i$: *Si le bateau est dans le cercle de plus grand diamètre mais pas dans celui de plus petit diamètre,*
        Si $cos(\alpha - \beta) > 0$:
            Si $cos(\pi + \alpha - \pi/2 - \beta) > cos(\pi + \alpha + \pi/2 - \beta)$:
                $\alpha = \alpha + \pi/2$
            Sinon :
                $\alpha = \alpha - \pi/2$
        Si $cos(\beta - \alpha) + cos(\xi) < 0$:
            Si $sin(\beta - \alpha) > 0$:
                hysteresis = 1
            Sinon :
                hysteresis = -1
            $\text{angle}_{\text{actuel}} = \beta + \pi + \text{hysteresis} \cdot \xi$
        Sinon :
            $\text{angle}_{\text{actuel}} = \alpha$
        Si $cos(\theta - \text{angle}_{\text{actuel}}) > 0$:
            $\delta_r = \delta_{r_{\max}} \cdot \sin(\theta - \text{angle}_{\text{actuel}})$
        Sinon :
            $\delta_r = \delta_{r_{\max}} \cdot \text{sign}\left(\sin(\theta - \text{angle}_{\text{actuel}})\right)$
        $\delta_{smax} = \pi/2 \cdot ((cos((\beta - \text{angle}_{\text{actuel}})) + 1)/2)$
        $\delta_s = -sign(\psi_{ap}) \cdot min(abs(\pi - abs(\psi_{ap})), \delta_{smax})$

    Sinon : *Si le bateau est dans le cercle de plus petit diamètre,*
        Si $\cos(\beta - \alpha) + \cos(\xi) > 0$:
            Si $\cos(\alpha - \beta - \pi + \xi) > \cos(\alpha - \beta - \pi - \xi)$
                hysteresis = -1
            Sinon :
                hysteresis = 1
            $\text{angle}_{\text{actuel}} = \beta + \pi + \text{hysteresis} \cdot \xi$
        Sinon :
            $\text{angle}_{\text{actuel}} = \alpha$
        Si $cos(\theta - \text{angle}_{\text{actuel}}) > 0$:
            $\delta_r = \delta_{rmax} \cdot sin(\theta - \text{angle}_{\text{actuel}})$
        Sinon :
            $\delta_r = \delta_{rmax} \cdot sign(sin(\theta - \text{angle}_{\text{actuel}}))$
        $\delta_{smax} = \pi/2 \cdot (cos((\beta - \text{angle}_{\text{actuel}})) + 1)/2)$
        $\delta_s = -sign(\psi_{ap}) * min(abs(\pi - abs(\psi_{ap})), \delta_{smax})$

3.2.3 Algorithme "Path Following"

L'algorithme "Path Following" fait réaliser tout un parcours au bateau en lui faisant atteindre un premier point de repère puis un deuxième une fois le premier atteint et poursuit ainsi avec tous les points de repère de la liste de points. En disposant des deux algorithmes précédents, son fonctionnement est simple. Il applique l'algorithme "Station Keeping" jusqu'à parvenir dans la zone délimitée par le cercle de plus petit diamètre autour du point visé puis applique l'algorithme "Line Following" pour sortir de cette zone et utilise de nouveau l'algorithme "Station Keeping" avec un nouveau point de repère. Il continue de la sorte avec tous les points suivants.

Algorithme de "Path Following" : parcours entre les différents points de repère
Variables d'entrée : $\vec{\mathbf{X}}$ (x, y, θ, v et w), $\vec{\omega}$, $\vec{\mathbf{P}}$ (on dispose donc aussi de $\vec{\mathbf{pos}}$, de $\vec{\mathbf{bm}}$, de α, de $\vec{\mathbf{w_{ap}}}$, de $\vec{\mathbf{psi_{ap}}}$)
Si d $> d_i$: *Si le bateau n'est pas dans le cercle de plus petit diamètre autour du point, c'est-à-dire s'il n'a pas atteint le point,* Appliquer l'algorithme "Station Keeping"
Sinon : *Si le bateau a atteint le point,* Appliquer l'algorithme "Line Following"

3.3 Algorithme d'observation

Pour améliorer la précision du système, on implémente un filtre de Kalman. Celui-ci a pour but d'estimer les valeurs futures de notre vecteur d'état à partir des données mesurées. Notre système n'est cependant pas linéaire en raison de la non linéarité des angles. En effet, dans un système faisant intervenir des angles (angle du vent, angle de la voile), les relations entre ces derniers impliquant de nombreuses fonctions trigonométriques sont non linéaires. Pour faire face à ce problème, on utilise un filtre de Kalman étendu ("Extended Kalman Filter" aussi appelé EKF) qui a l'avantage de parer à ce problème en linéarisant les relations autour de l'estimation courante de l'état de ce système.

3.4 Filtres

Certains capteurs renvoient des valeurs qui présentent parfois des variations brutales notamment dues à la présence de bruits. On implémente donc un filtre pour remédier à l'erreur engendrée par ces possibles variations. Il s'agit d'un filtre médian qu'on applique aux valeurs de l'IMU et de l'anénmomètre. Celui-ci stocke les 15 dernières valeurs lues par ces capteurs puis en renvoit la valeur médiane.

4 Simulation

4.1 Implémentation d'une simulation

Pour tester le bon fonctionnement des algorithmes de contrôle, j'ai créé une simulation donnant un premier aperçu de nos résultats.

4.1.1 Evolution du bateau virtuel

Pour faire évoluer le vecteur d'état du bateau virtuel , on utilise une référence fournie par notre tuteur [13] sur l'évolution du vecteur d'état d'un voilier. Pour cela, on introduit les constantes suivantes :

- $p0 = 0.03$, coefficient de dérive,
- $p1 = 40$, frottement tangentiel,
- $p2 = 6000$, frottement angulaire,
- $p3 = 200$, portance de la voile,
- $p4 = 1500$, portance du gouvernail,
- $p5 = 0.5$, distance au centre d'effort de la voile,
- $p6 = 0.5$, distance au mât,
- $p7 = 2$, distance au gouvernail,
- $p8 = 300$, masse du bateau,
- $p9 = 400$, moment d'inertie,
- $p10 = 0.2$, coefficient de rupture du gouvernail

On implémente ainsi l'algorithme ci-dessous donné dans la même référence [13] :

Algorithme d'évolution du vecteur d'état
Variables d'entrée : $\vec{\mathbf{X}}$ (x, y, θ, v et w), $\vec{\omega}$ (on dispose donc aussi de $\mathbf{\vec{pos}}$, de $\mathbf{\vec{w_{ap}}}$, de $\vec{\psi_{\mathbf{ap}}}$) Variables de sortie : $\dot{\vec{\mathbf{X}}}$
On introduit gr et gs, les forces sur le gouvernail et sur la voile : $gr = p4 \cdot v^2 \cdot \sin(\delta_r)$ $gs = p3 \cdot a_{ap} \cdot \sin(\delta_s - \psi_{ap})$
$\dot{x} = v \cdot \cos(\theta) + p0 \cdot \omega_{\mathbf{n}} \cdot \cos(\beta)$ $\dot{y} = v \cdot \sin(\theta) + p0 \cdot \omega_{\mathbf{n}} \cdot \sin(\beta)$ $\dot{\theta} = w$ $\dot{v} = \frac{gs \cdot \sin(\delta_s) - gr \cdot p10 \cdot \sin(\delta_r) - p1 \cdot v^2}{p8}$ $\dot{w} = \frac{gs \cdot (p5 - p6 \cdot \cos(\delta_s)) - p7 \cdot gr \cdot \cos(\delta_r) - p2 \cdot w \cdot v}{p9}$ $\dot{\vec{\mathbf{X}}} = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \\ \dot{v} \\ \dot{w} \end{bmatrix}$

On notera que le vent a été simulé à l'aide d'un vecteur aléatoire dans la simulation. Il s'agit là d'une grandeur assez imprévisible qu'il semblait complexe de modéliser de manière fidèle à la réalité à l'aide d'une loi quelconque.

4.1.2 Sauvegarde des données

Lorsqu'on effectue un test en conditions réelles (test de l'algorithme "Line Following", "Station Keeping" ou "Path Following" sur un lac), on stocke l'ensemble de nos données à plusieurs instants de la mission dans un fichier .csv (on nomme ce type de fichiers des "logs"). Ces données peuvent ensuite être rejouées dans la simulation. Le bateau évolue en prenant les mêmes paramètres (positions, ouverture des voiles, du gouvernail...) que lors de l'essai sur le lac ce qui nous permet d'étudier à nouveau et plus précisément les tests effectués.

4.2 Création d'une interface graphique

Afin de faciliter l'utilisation de la simulation, j'ai créé un affichage graphique permettant de lancer la simulation ainsi que de choisir différents réglages. J'ai pour cela utilisé le logiciel Pygubu. Il permet de dessiner une fenêtre assez simple qui peut se composer de divers éléments interactifs tels que des boutons, des zones de texte à remplir par l'utilisateur, des listes déroulantes et bien d'autres.

Une fois la fenêtre créée, il est possible de générer le code python associé. Deux options étaient alors envisageables : générer un code utilisant le fichier pygubu créé ou générer un code indépendant de celui-ci ne faisant qu'appel à la librairie Tkinter. Deux des fenêtres que j'ai créées faisaient appel à un fichier pygubu mais j'ai par la suite opté pour la seconde option pour le reste des fenêtres car la modification d'une fenêtre qui utilise un fichier pygubu nécessite la modification de ce même fichier ce qui est particulièrement fastidieux.

Le code ainsi généré constituait une base que j'utilisais pour la création du reste de ma fenêtre.

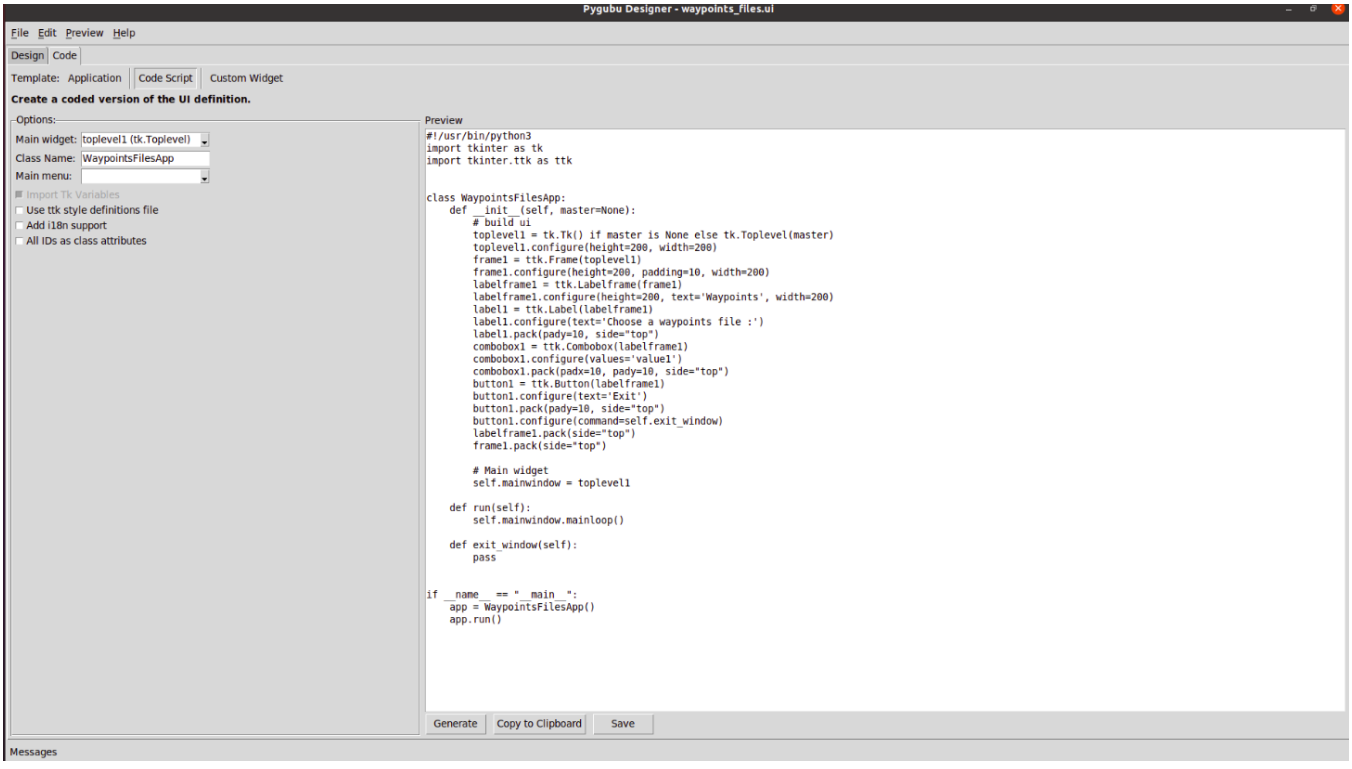

FIGURE 16 – Code généré lors de la création d'une fenêtre sur Pygubu

Chaque fenêtre est décrite par une classe et est reliée aux autres fenêtres bien souvent à travers l'utilisation de boutons qui sont décrits par des méthodes au sein du code. On compte au total 20 fenêtres liées les unes aux autres comme représenté sur la figure ci-dessous :

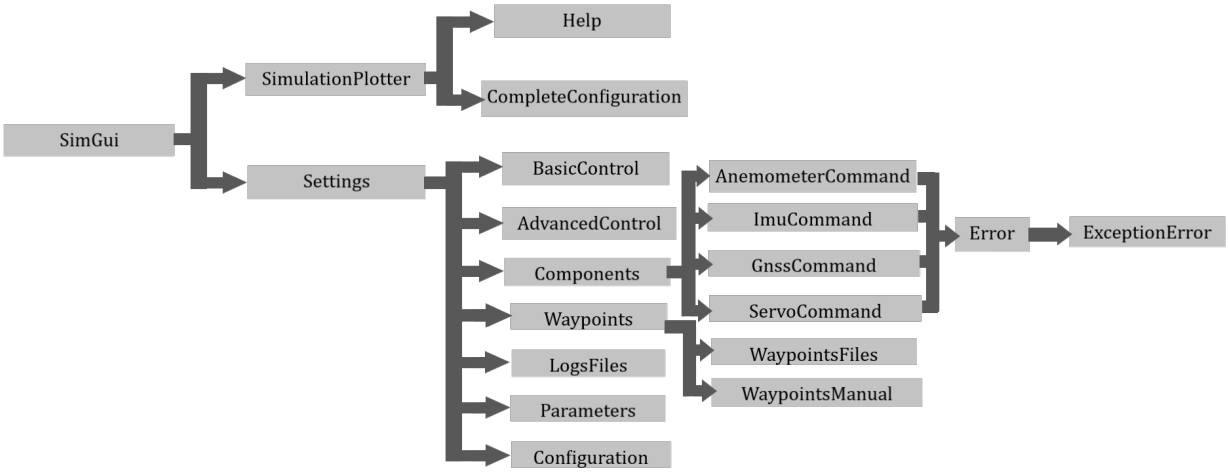

FIGURE 17 – Architecture des différentes fenêtres représentées par leur classe correspondante

La fenêtre générée par la classe "SimulationPlotter" lance la simulation et son affichage. Celle-ci montre le voilier représenté par un bateau noir, le vent représenté par une flèche rouge, le bateau et le vent observés par le filtre de Kalman représentés par un bateau et une flèche bleus ainsi que la ligne à suivre ou les points à atteindre selon l'objectif fixé pour le bateau. Cet objectif peut être choisi dans les fenêtres "Basic Control" (permet de choisir un superviseur) ou "Advanvced Control" (permet de chosir un superviseur, un controlleur et un observateur).

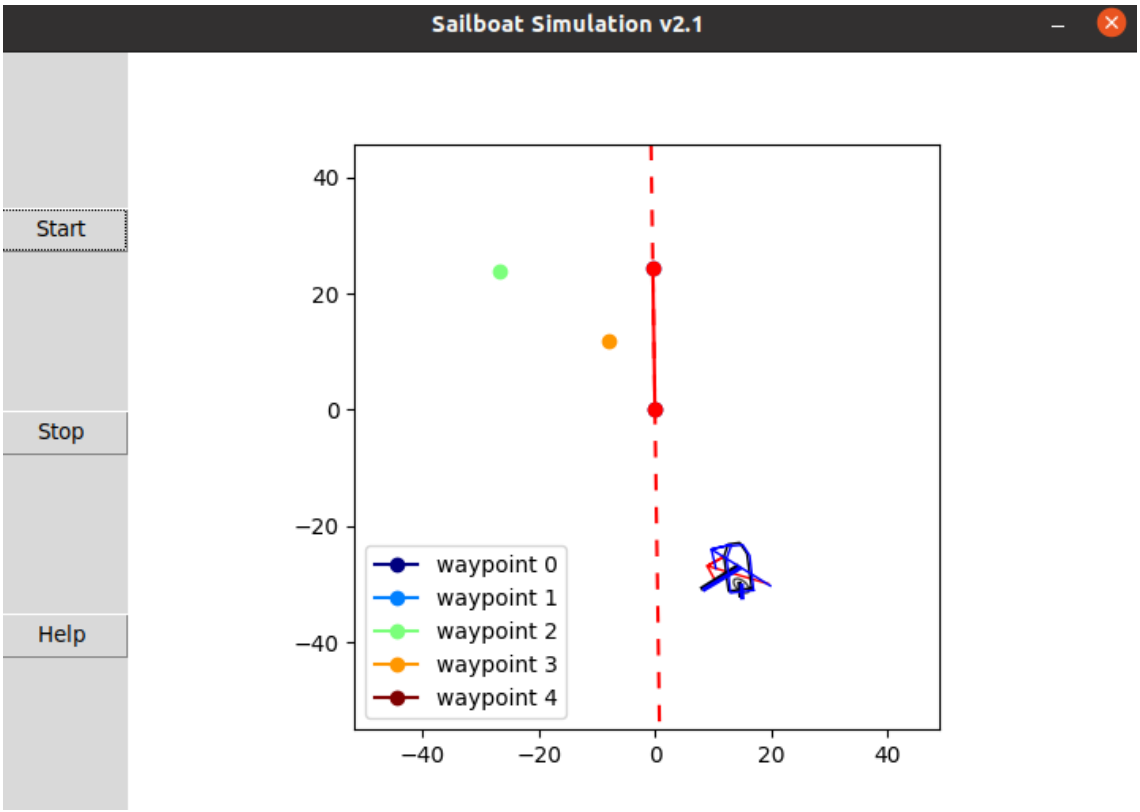

FIGURE 18 – Affichage graphique de la simulation pour un superviseur de type "Station Keeping"

De nombreux paramètres tels que la liste de points à atteindre ou encore la vitesse du vent peuvent être modifiés dans les différentes fenêtres.

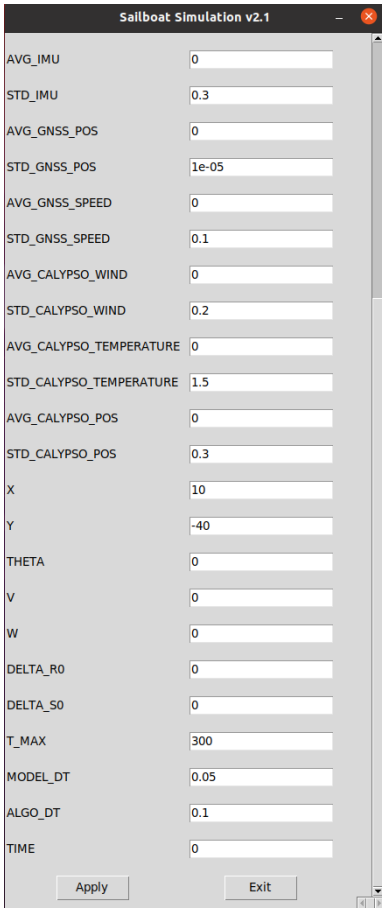

FIGURE 19 – Fenêtre de modification des paramètres

5 Résultats et problèmes rencontrés

5.1 Tests sur le lac

Une fois le programme fonctionnel, plusieurs tests ont été effectués sur le lac Bournville situé au sud de Birmingham.

FIGURE 20 – Bateau naviguant sur le lac

Les différents algorithmes ont alors été testés (”Line Following”, ”Station Keeping” et ”Path Following”) en conditions réelles. De nombreux tests ont été interrompus à cause des conditions météorologiques notamment de fortes averses qui survenaient brutalement mais nous avons bénéficié d'une météo clémente à quelques reprises au mois d'août. Lors des tests qui ont pu être menés dans de meilleures conditions, nous étions bien souvent confrontés à un bateau qui agissait de manière surprenante et assez aléatoire, loin d'atteindre les objectifs fixés. Nous avons cependant pu observer quelques réussites bien qu'assez rares.

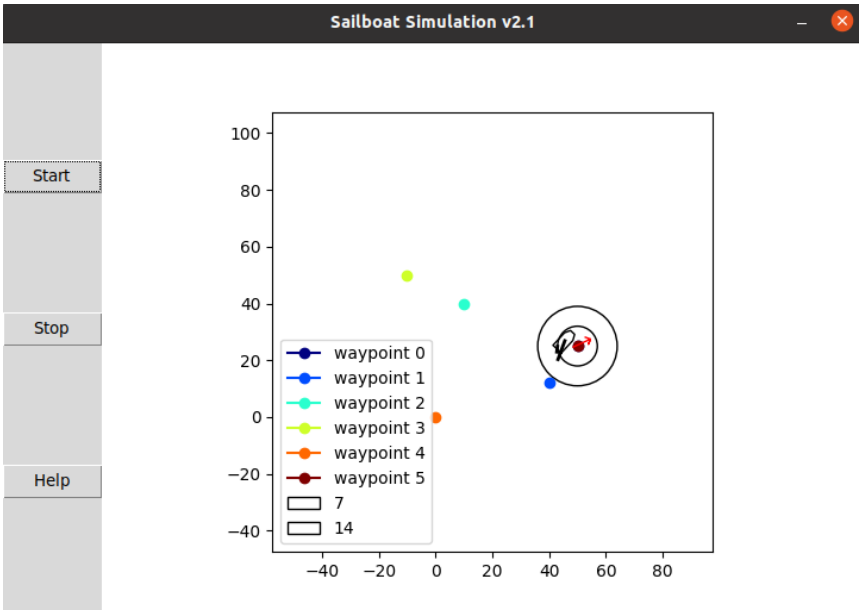

FIGURE 21 – Simulation rejouant une des réussites du bateau sur le lac pour la mission ”Station Keeping”

Bien qu'il nous était parfois difficile de mettre le doigt sur les causes de nos échecs, nous avons déterminé quelques pistes qui peuvent être à l'origine de ces erreurs. Celles-ci sont détaillées dans le paragraphe suivant.

5.2 Problèmes rencontrés

Voici quelques éléments pouvant possiblement être à l'origine des échecs rencontrés :

— Batterie du Calypso : l'anémomètre Calypso est un appareil se rechargeant à l'aide d'un panneau solaire. Il ne présente aucune autre alternative permettant de le recharger de manière filaire. Il s'est avéré que pour fonctionner correctemment, il nécessitait un niveau de batterie important que nous avions du mal à atteindre. Pour être chargé correctement, le Calypso devait être exposé à une forte lumière solaire pendant plusieurs jours ce que la météo ne permettait pas toujours. Cette information assez surprenante peut s'expliquer par le fait que l'appareil a été acheté il y a maintenant plusieurs années et qu'il avait passé plusieurs mois à l'abri du soleil dans une boîte avant le stage ce qui a possiblement détérioré sa batterie.

— Connexion au Calypso : dans le cas assez fréquent où l'anémomètre n'était pas complètement chargé, il se déconnectait assez fréquemment de la carte Navio avec laquelle il était connecté en bluetooth. Les dysfonctionnements du Calypso ont été la source de nombreux échecs puisqu'une fois la connexion à celui-ci perdue, notre bateau se retrouvait livré à lui-même, la vitesse et la direction du vent étant des informations cruciales pour déterminer l'angle d'ouverture des voiles et du gouvernail.

— Incohérence des mesures de l'IMU : en observant nos fichiers logs après certains échecs lors des missions, nous avons été surpris de constater que les valeurs mesurées par l'IMU étaient complètemenet incohérentes et variaient parfois de manière très brutales. En effectuant plusieurs tests sur terre, nous avons remarqué que malgré nos nombreuses tentatives de calibration de l'IMU nous ne parvenions à obtenir des valeurs cohérentes pour celui-ci que tant que le bateau ne tanguait pas. Lors d'une rotation autour d'un axe vertical par exemple, les valeurs mesurées sont tout à fait pertinentes mais si on effectue cette rotation tout en faisant tanguer le bateau d'une manière assez importante, les mesures deviennent complètement incohérentes. Le bateau était malheureusement bien souvent soumis à des vagues assez importantes ce qui faussait possiblement les données mesurées par l'IMU.

— Arrêt du GNSS : Il arrivait parfois que le GNSS cesse soudainement de fonctionner et reste bloqué sur les mêmes cordonnées bien que le bateau continue d'évoluer dans l'espace, nous ne sommes malheureusement pas parvenus à élucider les causes de ce problème.

— Emmêlement des cordes des voiles : les voiles sont liées par des cordes qui sont enroulées autout d'un cylindre en dessous du mât. En constatant qu'une des voiles ne bougeait plus, nous avons décidé de démonter le mât pour inspecter les cordes enroulées en dessous. Celles-ci étaient complètement emmêlées. Les démêler prend un certain temps car cela nécessite de retirer toutes les cordes puis de les réinserer dans le bateau. Il est nénamoins arrivé à plusieurs reprises que cette corde s'emmêle lors de tests et qu'une des voiles cesse donc de bouger.

5.3 Connaissances acquises

Malgré certains échecs lors de la phase expérimentale, ce stage a été riche en enrichissements et m'a permis d'améliorer mes connaissances en robotique ainsi que ma compréhension de nombreux outils. J'ai notamment développé une meilleure maîtrise des algorithmes de contrôle, des méthodes de calibration ou encore de l'utilisation des cartes Raspberry et Navio. J'ai aussi pu me familiariser davantage avec Git que j'ai du utiliser tous les jours pour partager mon code avec mes camarades de travail et sur lequel j'ai découvert de nouvelles fonctionnalités. J'ai également eu l'occasion de créer une interface graphique, ce que je n'avais pas réalisé depuis un projet d'informatique de première année et qui m'a particulièrement plu. Enfin, j'ai amélioré mes aptitudes en programmation en utilisant de nombreux outils ou procédés avec lesquels je n'étais pas à l'aise ou que je n'avais pas utilisé depuis un moment tels que les threads ou encore les classes python.

6 Conclusion

Le but initial de ce stage était d'autonomiser un voilier à l'aide des capteurs à ma disposition pour qu'il puisse réaliser des missions par lui-même. Même si le voilier ne parvenait pas toujours à atteindre ses objectifs, j'ai tout de même rencontré quelques succès lors des essais et je suis parvenue à mettre au point une simulation parfaitement fonctionnelle qui illustre la validité des différents algorithmes de contrôle implémenté.

Ce stage a aussi été particulièrement riche puisqu'il m'a permis de manipuler de nombreux composants et procédés très couramment utilisés en robotique ce qui m'a notamment permis de réaliser ce que j'appréciais le plus dans le domaine de la robotique et vers quel secteur je pourrai potentiellement me tourner pour mon PFE. J'ai notamment grandement apprécié la réalisation de l'interface graphique à laquelle j'ai consacré beaucoup de temps.

Ce projet comprend néanmoins de nombreuses possibles voies d'amélioration vis-à-vis des problèmes évoqués en 5.2. L'usage d'un anémomètre qu'on pourrait recharger de manière filaire ou d'un IMU plus performant pourraient par exemple contribuer à rendre le bateau plus précis et ainsi améliorer les performances de ce dernier. Au niveau du code, on pourrait également envisager l'utilisation d'algorithmes de machine learning qui permettraient au bateau de s'améliorer après chaque essai.

Bibliographie

— [1] : « Birmingham — History, Population, Map, & Facts — Britannica », 30 septembre 2023. Histoire de Birmingham

— [2] : the Guardian. « The Guardian University Guide 2022 – the Rankings ». Consulté le 30 septembre 2023. Classement de l'Université d'Aston d'après the Guardian

— [3] : « Aston University ». In Wikipedia, 29 septembre 2023. Informations sur l'Université d'Aston

— [4] : Team, Clearing Courses UK Editorial. « Aston University Clearing 2023 — Optometry & Pharmacy ». Clearing Courses UK, 4 juin 2023. Image de l'Université d'Aston

— [5] : « Hardware Setup — Navio2 ». Consulté le 30 septembre 2023. Introduction aux cartes Navio

— [6] : « Harendra RANGARADJOU / Aston Autonomous Sailboat 2023 · GitLab », GitLab, 18 août 2023, Gitlab du stage en collaboration avec Ludovic Mustiere et Harendra Rangaradjou

— [7] : « NAVIO2 Overview — Copter documentation ». Consulté le 30 septembre 2023. Image de l'architecture d'une carte Navio

— [8] : Prodeo Ingeniería y Consultoría, S.L. « Calypso Instruments — Precise Ultrasonic Wind Meters ». Consulté le 30 septembre 2023. Site vendeur de l'anémomètre Calypso

— [9] : Stelzer, Roland. Autonomous Sailboat Navigation : Novel Algorithms and Experimental Demonstration. Thèse de doctorat, Centre for Computational Intelligence, De Montfort University, Leicester, 2012.

— [10] : Melin, Jon. Modeling, control and state-estimation for an autonomous sailboat. Mémoire (Examensarbete) présenté à Teknisk- naturvetenskaplig fakultet, Uppsala universitet, septembre 2015.

— [11] : « Using LSM303DLH for a Tilt Compensated Electronic Compass », s. d.,Formules pour la calibration d'un accélèromètre et d'un magnétomètre

— [12] : Jaulin, Luc, and Fabrice Le Bars. A simple controller for line following of sailboats. Cardiff, Wales, England : Springer, 5th International Robotic Sailing Conference, 2012.

— [13] : Viel Christophe, Ulysse Vautier, Jian Wan, Luc Jaulin, "Position keeping control of an autonomous sailboat." IFAC PapersOnLine 51-29 (2018) : 14–19.

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* wanj3@aston.ac.uk

Nom du stagiaire accueilli / *Name of intern* Catherine Rizk

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

_____

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 ☑ 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 Jiarwan Signature stagiaire
Company stamp _____ *Intern's signature*

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