Réalisation d'un dirigeable autonome

Culture Sciences de l'Ingénieur

La Revue 3E.I

Salma BAIRAT[1] – Ibrahim EL KASSIMI[1]
Antoine HUET[2] – Anthony JUTON[3]

Édité le 10/11/2025

[1] *Élève en M1 au DER Sciences de l'ingénierie électrique et numérique (SIEN), ENS Paris-Saclay*

[2] *Élève en M1 Mécanique des matériaux et structures au DER Génie mécanique (GM), ENS Paris-Saclay*

[3] *Professeur agrégé de physique appliquée au DER des Sciences de l'ingénierie électrique et numérique (SIEN), ENS Paris-Saclay*

Cette ressource fait partie du N°117 de La Revue 3EI du 4ième trimestre 2025.

Cette ressource présente les solutions techniques retenues lors de la conception et la réalisation, dans le cadre du projet CoBRA de la L3 Saphire de l'ENS Paris Saclay, d'un dirigeable autonome pédagogique se déplaçant en intérieur, transportant et déchargeant un colis dans une cible. Elle traite ainsi de la cinématique de vol, de la localisation en intérieur, de l'informatique embarquée et de l'asservissement du dirigeable.

1 - Introduction

Porté par le succès médiatique des dirigeables de transport de la société Flying Whales (200 m, 60 tonnes de charge utile), le dirigeable est un dispositif volant original, impressionnant par sa taille et peu bruyant, bien adapté à l'enseignement. L'objectif du projet CoBRA (Course de Ballons Réactifs Autonomes) est de rendre autonome un dirigeable (2,4 m de long et 1,3 m diamètre) pour le dépôt de colis. Il est alors nécessaire de pourvoir le dirigeable d'un système de propulsion, d'un système de localisation dans son environnement ainsi que d'un asservissement. Ce projet aborde alors différents domaines des sciences et techniques de l'ingénieur rendant le projet pluridisciplinaire.

Figure 1 : Vue d'ensemble du dirigeable

2 - Propulsion et préhension

Cette partie s'intéresse à la solution de propulsion permettant de mouvoir le dirigeable dans l'espace 3D et au dispositif de livraison que constitue le treuil.

2.1 - Propulsion

La solution technique de propulsion doit permettre le déplacement jusqu'à la cible mais aussi la stabilisation du dirigeable au-dessus de celle-ci durant le temps de livraison. La solution de propulsion retenue permet deux natures de déplacement dans l'espace 2D parallèle au sol. Dans un premier temps l'objectif est de permettre au dirigeable de se déplacer à l'aide d'une rotation sur lui-même et d'un mouvement rectiligne de direction confondue avec l'axe du dirigeable (figure 2).

Nous proposons donc une première solution de propulsion composée de deux propulseurs latéraux permettant à la fois une avance rectiligne selon l'axe du dirigeable et un mouvement de rotation autour de l'axe vertical (figure 3).

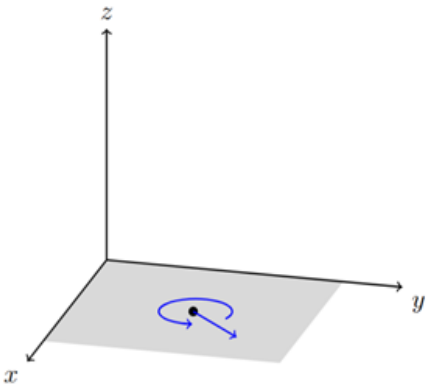

Figure 2 : Repère des premiers mouvements du dirigeable

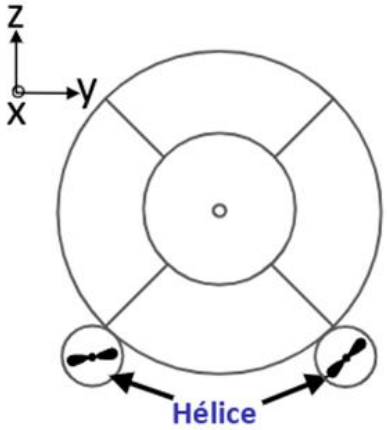

Figure 3 : Emplacements des deux premiers propulseurs - Vue de face

Dans un deuxième temps l'objectif est de stabiliser le dirigeable au-dessus de la cible, celui-ci restant immobile, en position et en rotation (figure 4). Pour ce faire, on souhaite en plus du mouvement rectiligne selon l'axe du dirigeable ajouter un mouvement de translation orthogonal à cette première translation. On ajoute alors deux propulseurs latéraux (figure 5).

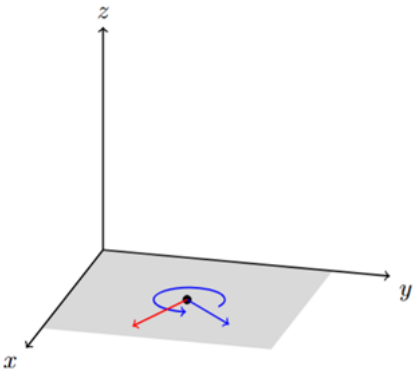

Figure 4 : Nouvelle direction de translation

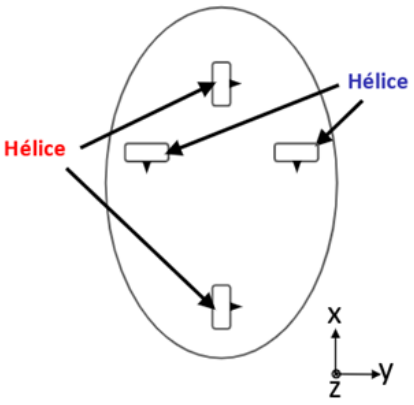

Figure 5 : Emplacements des deux propulseurs suivants - vue du dessous

Ces différentes positions de propulseurs sur le dirigeable permettent à celui-ci de se mouvoir dans un espace 2D selon deux méthodes différentes réalisant deux fonctions distinctes : l'approche et

la stabilisation au-dessus de la cible. On remarque cependant que l'on dispose maintenant de 4 propulseurs qui constituent une contrainte de masse plus importante.

Un dernier moteur de direction de poussée verticale permet de faire varier l'altitude du dirigeable (figure 6). De plus, le dirigeable est lesté afin de perdre en altitude lorsque son propulseur vertical est inactif, de sorte de redescendre naturellement en cas de batterie vide.

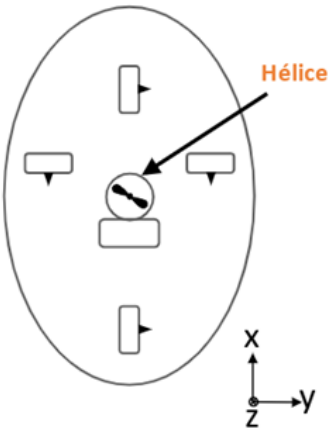

Figure 6 : Emplacement du propulseur vertical - vue du dessous

La principale contrainte dans la réalisation du dirigeable est la limite de masse embarquée. En effet, le dirigeable de 2300 L environ peut embarquer, compte-tenu de la masse de son enveloppe, seulement 650 g.

Notre solution de propulsion gourmande en nombre de propulseurs impose d'autant plus de contraintes sur la réalisation de ces derniers. Pour ce faire, nous avons choisi de réaliser des propulseurs latéraux miniatures et de diminuer dans un premier temps les dimensions des éléments de structure et de prendre des moteurs de drone (moteur brushless) plus adaptés à cette application pour leur bon rapport vitesse de rotation/masse du moteur.

La dernière étape pour créer les propulseurs latéraux est de connaître la poussée souhaitée. Nous avons donc estimé le coefficient de traînée de l'enveloppe via une étude de mécanique des fluides à l'aide du logiciel *Solidworks* qui est cohérente avec les ouvrages (forme classique connue) [1]. Ensuite, avec la vitesse de déplacement souhaitée, nous avons obtenu une force de poussée nous amenant à choisir un couple moteur/hélice adéquat.

Nous avons donc choisi cette combinaison de variateur réversible, moteur (62W, 17g) et hélice :

Figure 7 : ESC CONTROLEUR 16A F3P 3D/4D T-HOBBY

Figure 8 : AS2303 F3P / 3D / 4D 2300KV 17GR T-MOTOR

Figure 9 : Hélice 5x5E

Des structures ont ensuite été réalisées à l'aide de tiges carbone et d'impression 3D et fixées à l'enveloppe (figure 10).

Figure 10 : Structure propulseur latéral

Pour le propulseur vertical le même moteur a été choisi mais une hélice plus grande a été sélectionnée, dû au fait qu'il n'y a qu'un seul moteur.

Le choix de la cinématique de vol s'est révélé judicieuse, permettant les déplacements souhaités et un asservissement indépendant des 4 axes (X,Y,Z, lacet).

En revanche, l'hélice de propulsion verticale de plus grande dimension est moins dynamique que les autres (difficulté au démarrage ou lors de changement de sens de rotation). De surcroît, la structure comprenant le moteur de poussée verticale transmet des vibrations à la caméra, qui ne permet alors pas la localisation. Une solution serait alors d'utiliser deux petits propulseurs verticaux identiques aux propulseurs latéraux et de les fixer sur des structures sur l'enveloppe, indépendantes de la nacelle.

2.2 - Préhension

La seconde partie constituait la réalisation d'un treuil muni d'une pince permettant de déposer un colis dans une cible. La solution présentée ici se compose de trois fils de treuil formant une pyramide à base triangulaire assurant la stabilité de la pince. Une machine à courant continu en haut permet l'enroulage et le déroulage des câbles (câbles mécaniques et électriques). Une autre machine à courant continu positionnée sur la pince permet son ouverture et sa fermeture. Encore une fois la réduction de masse est très importante dans la réalisation de cette partie.

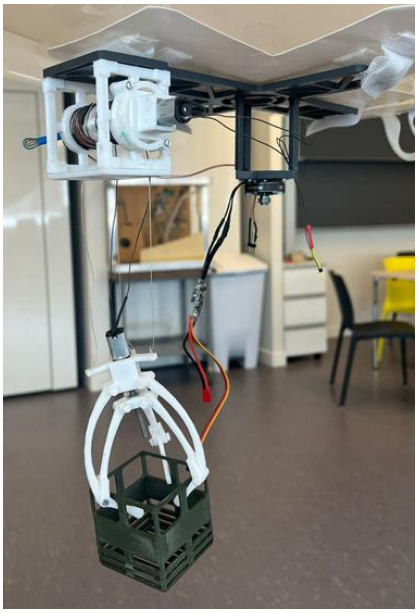

Figure 11 : Dispositif de préhension avec colis

2.3 - Nacelle

Une structure inférieure fixée au dirigeable permet enfin de fixer le treuil, le propulseur vertical, la batterie ainsi que les cartes de commandes et les capteurs qui sont détaillés dans la partie suivante. Cette structure comprend plusieurs éléments assez volumineux mais doit aussi être de masse minimale. L'espacement entre l'hélice verticale et l'enveloppe du dirigeable doit permettre un flux d'air suffisant mais l'hélice ne doit pas cacher le sol à la caméra grand angle.

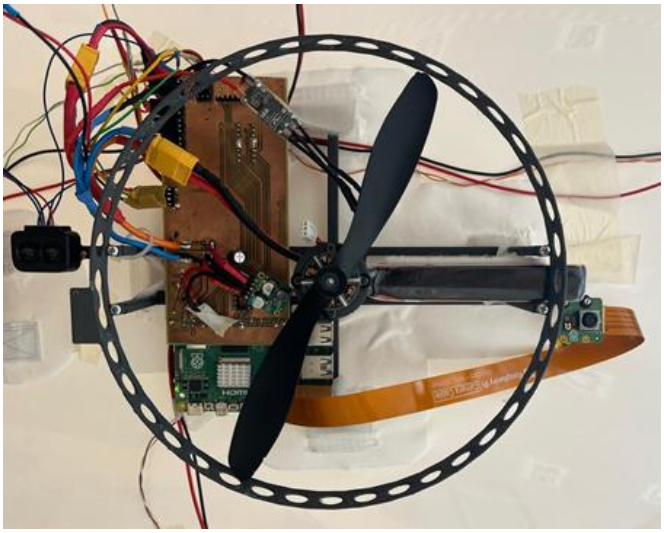

Figure 12 : Nacelle inférieure

3 - Localisation intérieure

Pour assurer la navigation autonome du dirigeable en intérieur (et donc sans solution GNSS possible), plusieurs capteurs sont utilisés afin de collecter des informations sur l'environnement et l'état du système :

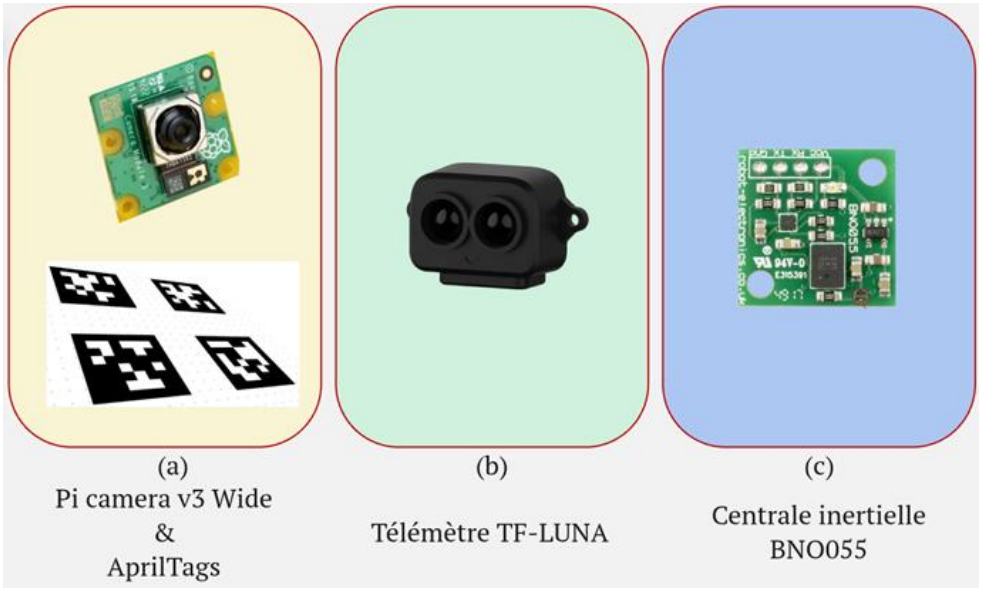

Figure 13 : Capteurs utilisés pour la localisation

La conception du dirigeable, soumise à une contrainte de masse stricte, nous a poussés à limiter le nombre de capteurs embarqués. Nous avons donc commencé avec une configuration minimale basée sur une **caméra** et des marqueurs **AprilTags** (Figure 13-a et Figure 16). Après plusieurs tests visant à évaluer la précision du système (Figure 15), nous avons constaté que l'estimation de l'altitude manquait de fiabilité. Or, ce paramètre est essentiel pour les phases de préhension et de dépôt du

colis. De plus, lorsque le dirigeable se trouvait trop proche du sol, les tags n'étaient plus détectés. Pour pallier ces limitations, nous avons ajouté un **télémètre** TF Luna (figure 13-b) afin d'obtenir une mesure directe de la distance au sol. Cependant, les inclinaisons du dirigeable influençaient la valeur renvoyée par le télémètre. Nous avons donc intégré une **centrale inertielle** BNO055 permettant de corriger la mesure d'altitude en fonction de l'orientation du dirigeable, améliorant ainsi la précision globale de la localisation.

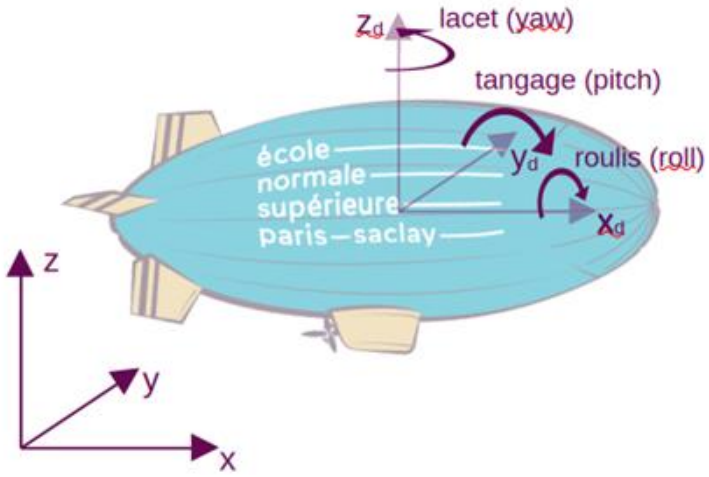

Figure 14 : Repère terrestre (x,y,z), relatif au dirigeable (xd, yd, zd) et angles d'Euler associés

3.1 - Localisation via la vision avec la bibliothèque Apriltags.

Une description détaillée de l'utilisation de la bibliothèque AprilTags est disponible dans la ressource « Localisation via la vision avec la bibliothèque Aprilags » [2].

Nous avons commencé par équiper le dirigeable d'une caméra **Picam v3 Wide** associée à des **AprilTags**, des marqueurs visuels similaires à des codes QR (Figure 16). La Picam v3 Wide présente plusieurs avantages pour notre application : elle offre une haute résolution, un grand angle de vue et une bonne sensibilité à la lumière, ce qui permet une détection fiable des tags même lorsque le dirigeable se déplace ou que les conditions lumineuses varient, tout en couvrant une large zone avec un seul capteur. Les AprilTags, de leur côté, permettent une détection relativement rapide (100 ms) de la position.

Pour évaluer la précision de notre système basé sur la caméra Picam v3 et les AprilTags, nous avons réalisé plusieurs mesures expérimentales à différentes hauteurs et positions du dirigeable, notons que le cahier des charges imposait une erreur de position inférieure à 10 cm.

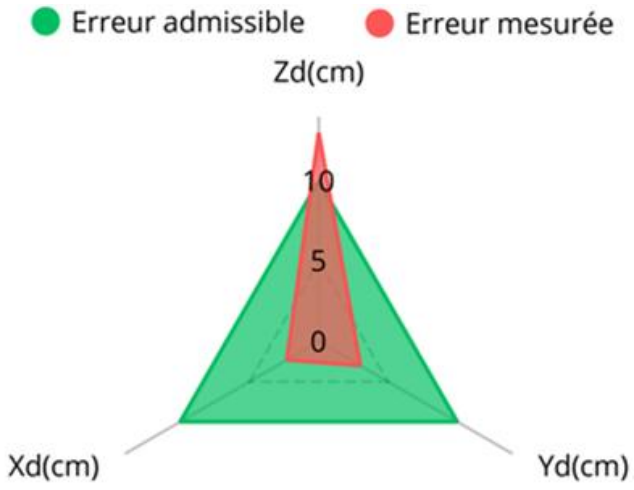

Figure 15 : Erreur sur la position estimée par le système caméra-AprilTags

Ces tests ont montré que, bien que les AprilTags offrent une bonne estimation de la latitude, la longitude, et l'angle de lacet, l'erreur sur l'altitude ne satisfait pas le cahier des charges. En plus, si le dirigeable est trop proche du sol la caméra ne détecte pas les tags. Ces résultats mettent en évidence les limitations de l'usage exclusif de la caméra, et justifient l'intégration d'un télémètre pour assurer un suivi fiable de l'altitude du dirigeable.

Figure 16 : Dirigeable se déplaçant au-dessus des AprilTags

3.2 - Estimation de l'altitude par télémétrie.

Afin de pallier les limites de la localisation par vision, notamment la perte de détection des AprilTags à faible altitude et l'erreur sur l'altitude estimée, nous avons intégré un télémètre TF-Luna pour mesurer directement la distance entre le dirigeable et le sol. Ce capteur repose sur la technologie ToF (Time of Flight), qui consiste à mesurer le temps mis par une onde lumineuse (généralement infrarouge) pour parcourir l'aller-retour entre le capteur et la surface réfléchissante. Cette méthode permet d'obtenir une estimation rapide et précise de l'altitude du dirigeable. Cependant, en cas d'inclinaison, le faisceau n'est plus perpendiculaire au sol, ce qui entraîne une surestimation de la distance mesurée (la figure 17 illustre ce problème). Ainsi, la précision du télémètre dépend fortement de l'orientation du dirigeable, limitant son utilisation seule pour une mesure d'altitude fiable.

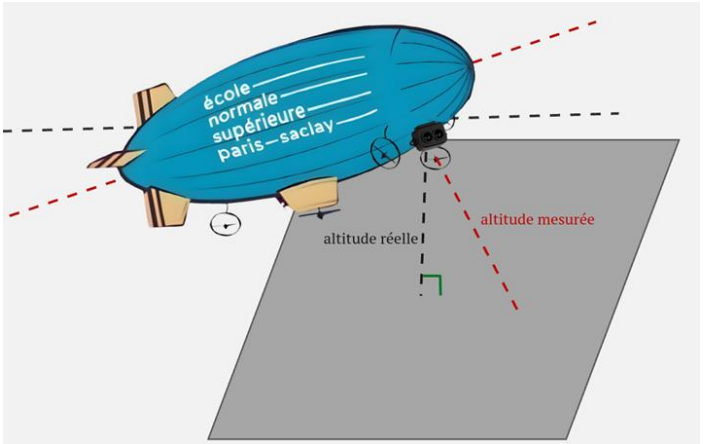

Figure 17 : Erreur d'estimation de l'altitude à cause de l'inclinaison du dirigeable

Afin de compenser l'erreur d'altitude causée par l'inclinaison du dirigeable, une centrale inertielle BNO055 a été intégrée au système.

3.3 - Centrale inertielle BNO055.

Le BNO055 (Bosch Sensortec) est un capteur inertiel de type IMU 9-DoF (Inertial Measurement Unit, 9 degrés de liberté) intégrant un accéléromètre, un gyroscope et un magnétomètre tri-axiaux dans un même circuit intégré (System-in-Package SiP). Il comprend également un microcontrôleur interne (Cortex-M0+) chargé d'exécuter l'algorithme de fusion des capteurs et de fournir directement les sorties d'orientation (quaternions, angles d'Euler, vecteurs magnétiques, etc.).

Pour corriger la mesure de l'altitude, on récupère les valeurs des angles de roulis et de tangage renvoyées par la centrale inertielle et on applique la formule suivante :

$$h = h_{mesuré} \times cos(\Phi) \times cos(\theta)$$

Avec :

h : L'altitude du dirigeable.

$h_{mesuré}$: L'altitude mesurée par le télémètre.

Φ : L'angle de roulis du dirigeable.

θ : L'angle de tangage du dirigeable.

4 - Électronique embarquée

Cette partie s'intéresse à l'électronique embarquée dans le dirigeable permettant la lecture des capteurs, la commande des moteurs et l'asservissement.

4.1 - Informatique embarquée : le nano-ordinateur et le protocole i2c

Le nano-ordinateur est la pièce centrale de notre solution d'électronique embarquée : il traite les informations reçues des capteurs et envoie les signaux de commande aux moteurs. La bibliothèque AprilTag nécessitant des ressources importantes pour un traitement en moins de 100 ms des images, nous avons choisi un nano-ordinateur **RaspberryPi5** bon marché et avec une large communauté partageant son expérience.

Figure 18 : Nano-ordinateur Raspberry Pi 5, source raspberrypi.com

L'utilisateur peut écrire et lancer des programmes informatiques (de localisation ou de commande moteurs) sur la Raspberry en s'y connectant, par Wifi, en **ssh** via un terminal ou VScode sur un pc.

On parle d'ordinateur headless (sans clavier ni écran). La carte Raspberry supportant le bluetooth, il est possible d'y connecter une manette de console Sony PS4.

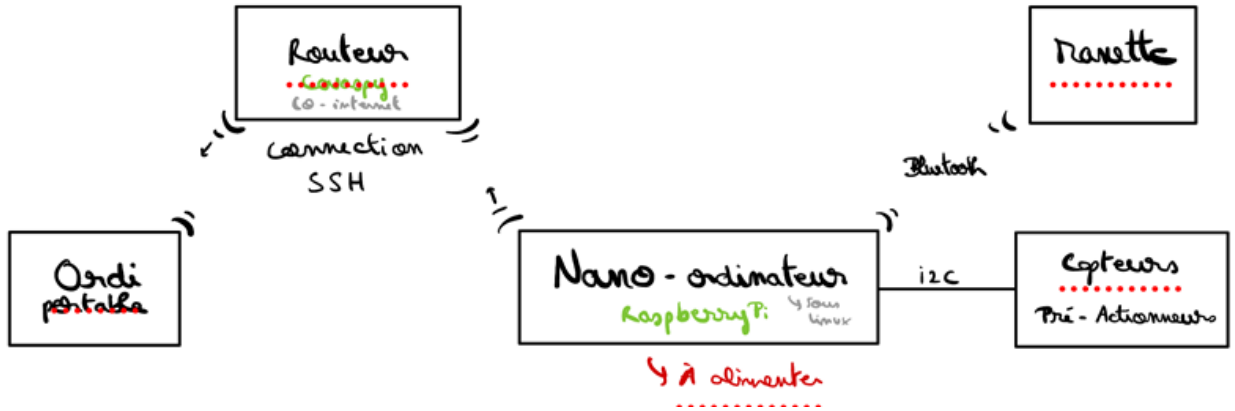

Figure 19 : Éléments communiquant avec le microcontrôleur

Le microcontrôleur communique avec les capteurs et les moteurs via le **protocole i2C**. Le protocole i2C est un bus série permettant de faire communiquer entre eux un composant électronique "**maître**" et des composants électroniques "**esclaves**" grâce à deux signaux : un signal de donnée (**SDA**) et un signal d'horloge (**SCL**).

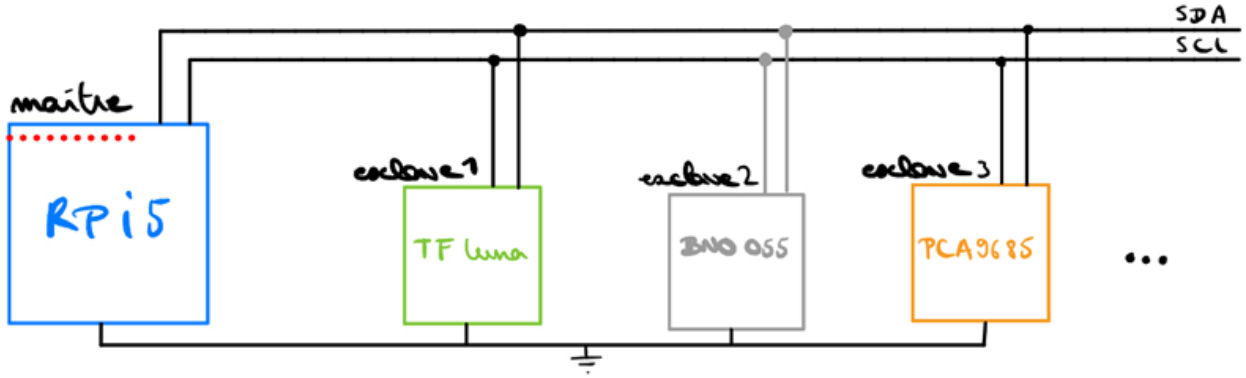

Figure 20 : Protocole i2C

Sur notre dirigeable, le maître est le nano-ordinateur Raspberry Pi5 et les esclaves sont les différents capteurs et moteurs.

Les bibliothèques de communication avec les capteurs et le module PCA9685 sont fournis en annexe de cette ressource [3].

4.2 - Commande des moteurs : le générateur de PWM PCA9685 et les différents moteurs

Le générateur de PWM PCA9685

Pour commander des moteurs, on utilise des signaux PWM (Pulse Width Modulation) dont on modifie la largeur d'impulsion pour commander la vitesse des moteurs. Généralement :

- 1 ms correspond à la vitesse maximale en marche arrière ;
- 1,5 ms correspond à la vitesse nulle ;
- 2 ms correspond à la vitesse maximale.

Les PWM hardware des Raspberry Pi étant peu nombreuses et les PWM software pas suffisamment stables, on utilise un composant externe, le **PCA9685,** pour générer ces signaux.

Figure 21 : Composant PCA9685, source lisleapex.fr

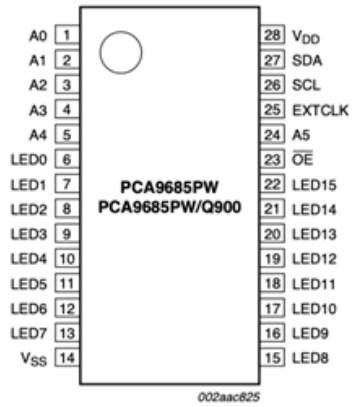

Figure 22 : Configuration des pins, source Datasheet PCA9685

Le PCA9685 est capable de générer **16** signaux PWM à la **même fréquence** (sur les pins LED0 à LED15 sur la figure).

On choisit une fréquence de PWM de **50Hz** (période de 20 ms) comme suggéré dans la documentation technique du variateur du moteur Brushless.

Une fois la période imposée, ainsi que le passage à 1 de la PWM au début de chaque période, pour régler la largeur d'impulsion d'une PWM n, le PCA9585 met à disposition deux registres :

- **LEDn_OFF_L** : octet de poids faible (bits[7 :0]) de l'instant où la PWM passe à 0
- **LEDn_OFF_H** : octet de poids fort (bits[11 :8]) de l'instant où la PWM passe à 0

La bibliothèque fournie en annexe propose alors trois fonctions de haut niveau :

- brushless.cmd_vit_pourcent(vit_pourcent) pour la commande des moteurs de déplacement ;
- MCC_2PWM.cmd_vit_pourcent(vit_pourcent), pour les moteurs à courant continu des treuil et pince ;
- servo.cmd_angle_deg(angle) pour les servomoteurs (non utilisée ici).

Pour tester ces fonctions, les moteurs peuvent être commandés depuis un ordinateur via connexion ssh ou par une manette PS4 via connexion Bluetooth à la Raspberry Pi 5.

Le moteur brushless

Comme mentionné dans la partie 2.1, les cinq moteurs brushless que nous avons choisis pour le déplacement du dirigeable sont :

Figure 23 : AS2303 F3P / 3D / 4D 2300KV 17GR T-MOTOR

La commande de ce moteur se fait via un **variateur** qui peut être paramétré grâce à l'interface logicielle **BLHeliSuite 16.7.14**. On peut notamment y paramétrer la possibilité pour le moteur de tourner dans les deux sens de rotation.

Le variateur reçoit un signal PWM du PCA9685 et met en rotation le moteur brushless à la vitesse correspondant à la largeur d'impulsion de la PWM.

La MCC (Machine à Courant Continu)

Pour le treuil et la pince, nous avions choisis ce modèle de mcc :

Figure 24 : MCC 1210GM-06100, source lextronic.fr

La commande d'une MCC se fait via un **hacheur** (pont en H) qui a besoin de :

- Un signal **PWM** pour la **valeur** absolue de la **vitesse ;**
- Un premier **signal logique** pour le **sens AVANT** de rotation ;
- Un second **signal logique** pour le **sens ARRIÈRE** de rotation.

Le signal PWM est généré par le PCA9685 comme vu précédemment.

Les signaux logiques (prennent les valeurs 0 ou 1) sont générés par la Raspberry Pi 5 :

- Quand la vitesse est positive, le premier signal logique est mis à 1 et le second à 0.
- Quand la vitesse est négative, le premier signal logique est mis à 0 et le second à 1.

4.3 - Carte électronique

Pour alimenter et relier les composants d'électronique embarquée, nous avons conçu une **carte électronique intermédiaire** sur le logiciel Eagle en prenant en compte les entrées pour l'alimentation (conversion des 7,2 Vcc de la batterie en 5 Vcc), les communications avec les différents capteurs, pré-actionneurs et actionneurs, et les pistes les reliant entre eux. Les servomoteurs, indiqués sur le schéma, ne sont pas utilisés par notre équipe.

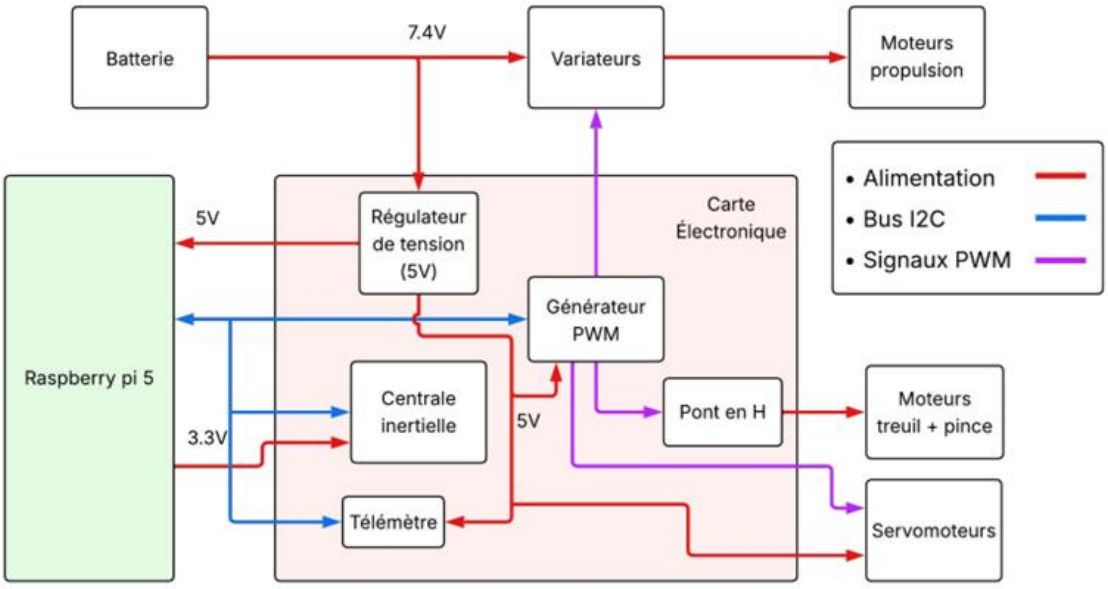

Figure 25 : Schéma synoptique de la carte électronique réalisée sur Eagle

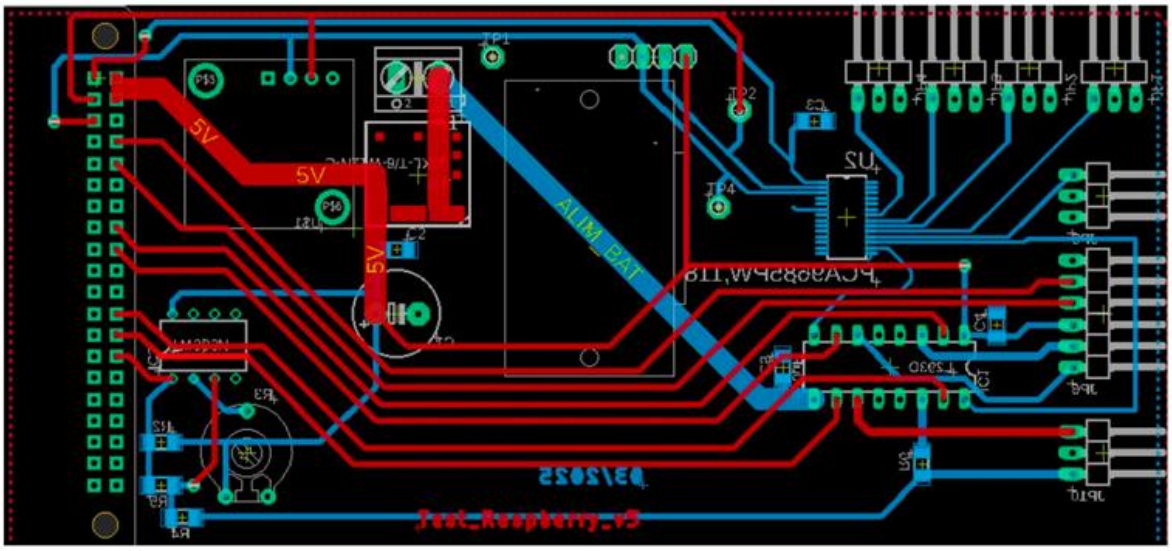

Figure 26 : Carte électronique réalisée sur Eagle

5 - Asservissement

Maintenant que le dirigeable est localisé et les moteurs pilotables, il nous reste à asservir les axes de déplacement du dirigeable pour s'assurer qu'il arrive à suivre une commande imposée. Le dirigeable ayant une inertie importante, son asservissement est un problème complexe mais très intéressant.

Pour faire simple dans cette première approche, la fin du projet étant en vue, nous avons asservi les 4 axes (x, y, z et lacet) indépendamment. Ce sont donc quatre problèmes identiques d'asservissement mono-variable que nous avons eu à résoudre. Est décrit ci-dessous l'asservissement selon l'axe z ; le schéma d'une boucle d'asservissement est le suivant :

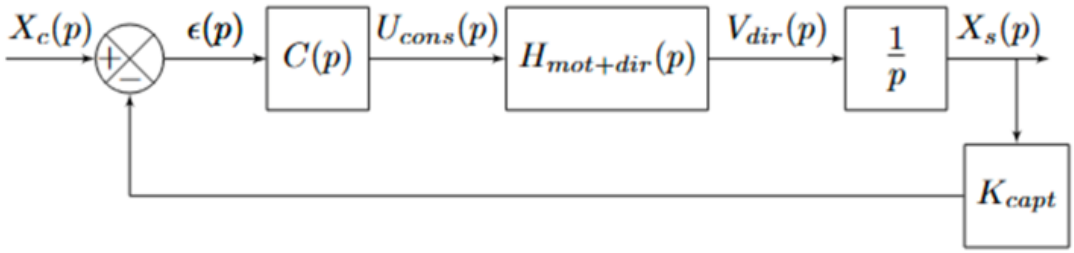

Figure 27 : Schéma de la boucle d'asservissement

5.1 - Première étape : identification

Nous commençons par réaliser un essai indiciel sur le dirigeable afin d'obtenir, à l'aide de Matlab un modèle du comportement du dirigeable le long de l'axe z.

Pour cela, nous avons écrit un programme simple, succession de montée et de descente de 12 s, réalisant l'enregistrement dans un fichier des informations du dirigeable (temps, position, commande) à chaque apparition d'une nouvelle valeur des capteurs.

Une fois les données acquises, nous les transférons sur le PC, nettoyons un peu (remplacement des '.' par des ',', suppression des points aberrants, séparation des données en colonne) et traçons.

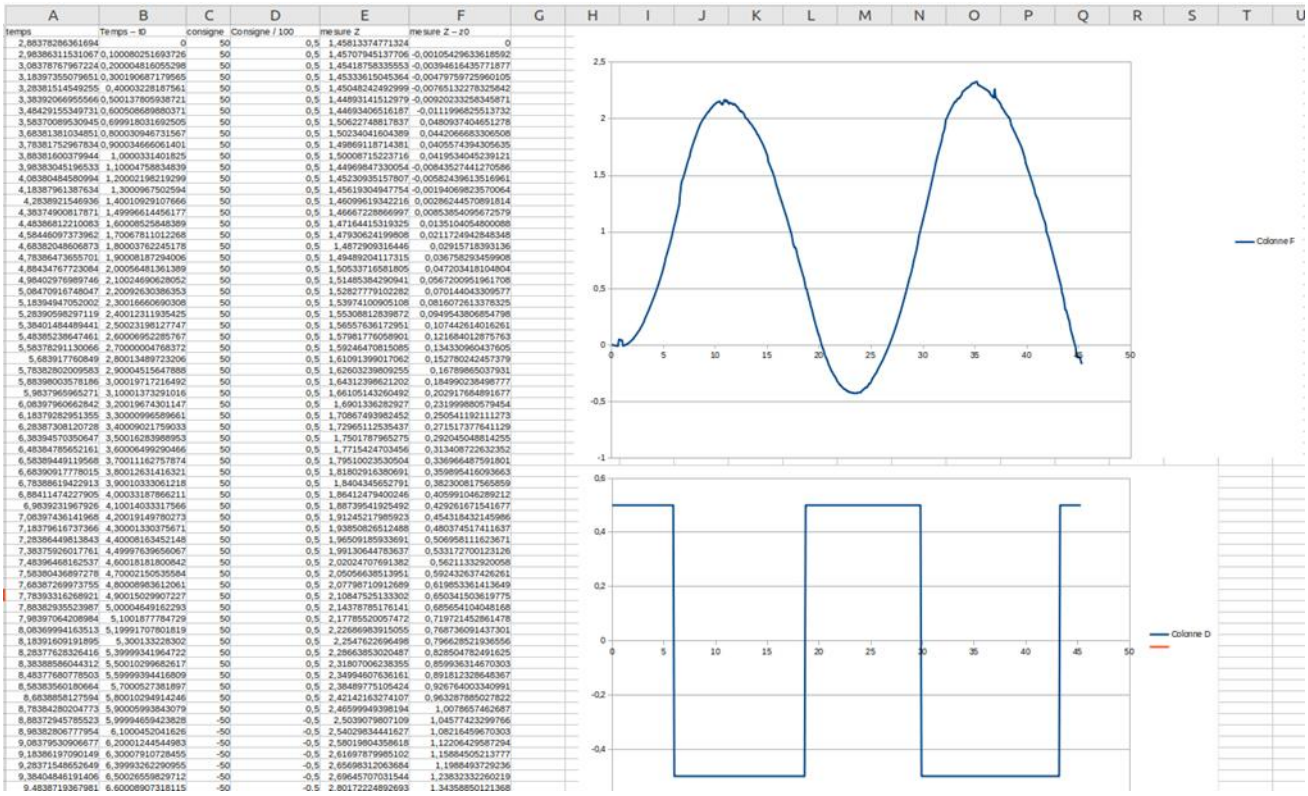

Figure 28 : Position et de la commande acquises lors d'un essai en boucle ouverte

Une fois les données vérifiées, elles sont exportées au format csv et exploitées sous Matlab avec la fonction sys_tf (ou l'outil SystemIdentification) pour obtenir un modèle du système.

```
>> data = readmatrix('/home/ajuton/Documents/a0_Saphire/Dirigeable/asservissement/data_essai_indiciel_3.csv');
>> t = data(:,1);
>> consigne = data(:,2);
```

```
>> z = data(:,3);
>> data2 = iddata(z, consigne, 0.1)

data2 =

Time domain data set with 454 samples.
Sample time: 0.1 seconds

Outputs      Unit (if specified)
   y1

Inputs       Unit (if specified)
   u1
```

```
>> sys_tf = tfest(data2, 3, 2);
>> sys_tf

sys_tf =
  From input "u1" to output "y1":
    0.01645 s^2 - 0.001026 s + 0.01059
  -------------------------------------
  s^3 + 6.152 s^2 + 0.6814 s + 3.634e-12

Continuous-time identified transfer function.

Parameterization:
   Number of poles: 3   Number of zeros: 2
   Number of free coefficients: 6
   Use "tfdata", "getpvec", "getcov" for parameters and their uncertainties.

Status:
Estimated using TFEST on time domain data "data2".
Fit to estimation data: 49.14%
FPE: 0.2219, MSE: 0.2133
```

Figure 29 : Commandes Matlab pour l'identification d'un modèle pour le comportement du dirigeable selon l'axe z

Un essai dans Matlab Simulink du modèle obtenu permet de vérifier sa fidélité au modèle réel :

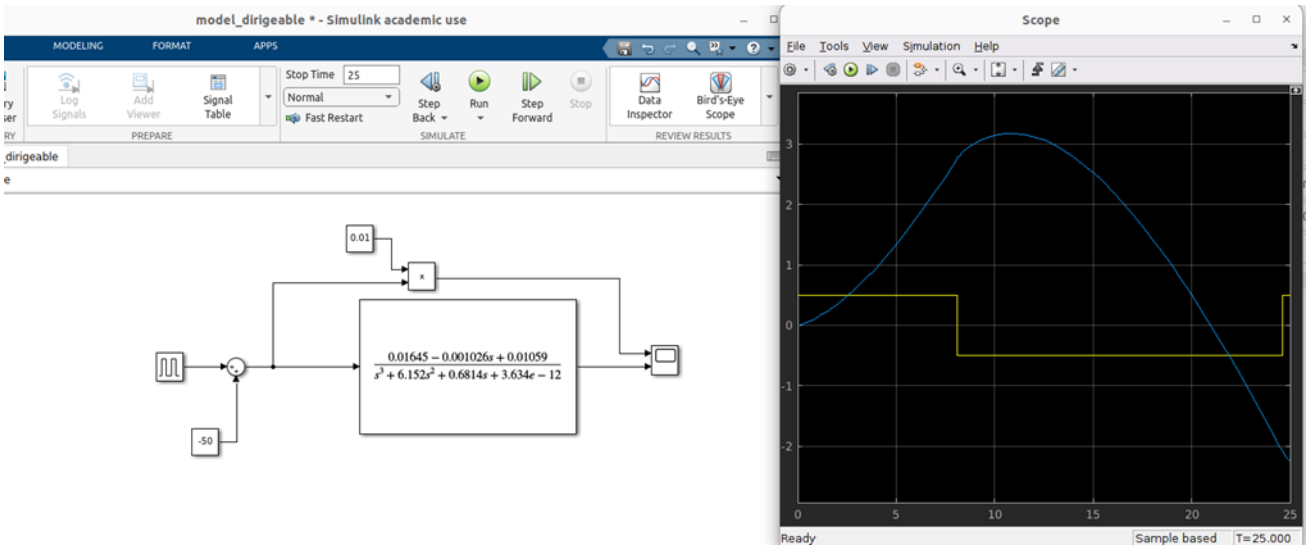

Figure 30 : Simulation en boucle ouverte pour vérifier le modèle obtenu

5.2 - Réglage d'un correcteur PID simulé

Nous avons ensuite ajouté dans Matlab Simulink un correcteur PID et utilisé la fonction auto-tune pour obtenir les premières valeurs de Kp, Ki (I), Kd (D).

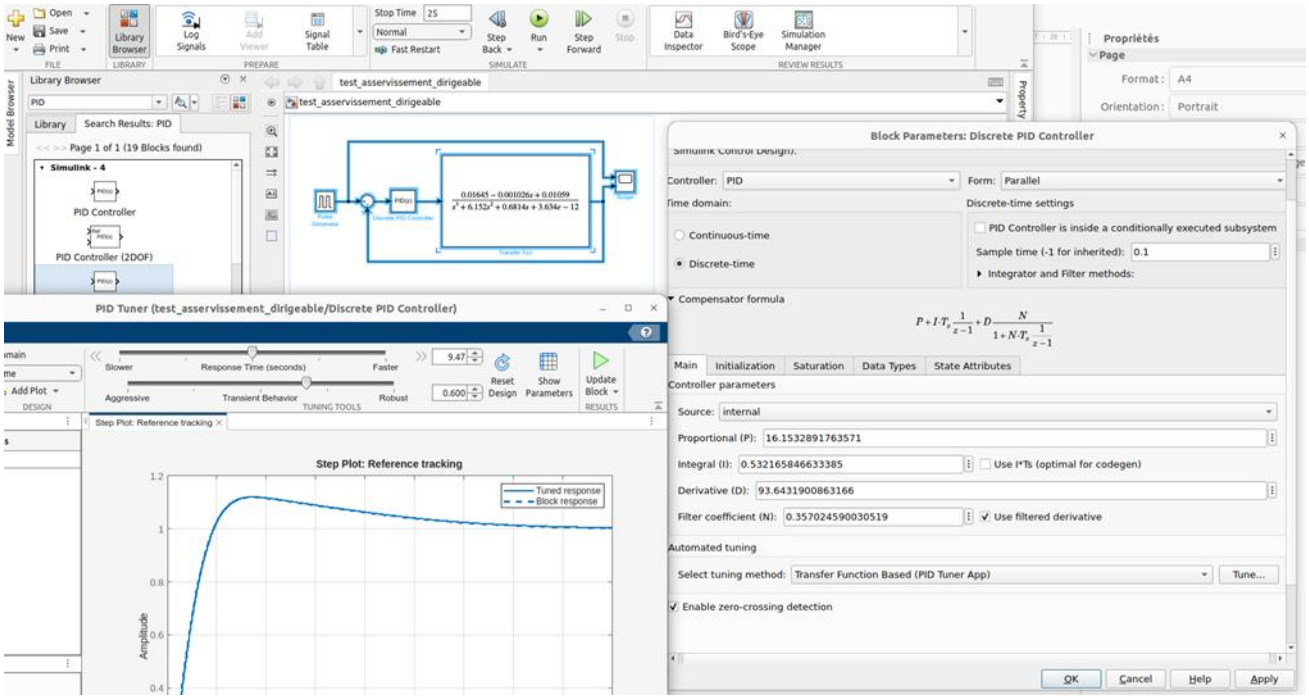

Figure 31 : Réglage du PID à l'aide de la fonction auto-tune de Matlab Simulink

5.3 - Intégration d'un correcteur réel et réglage de ce correcteur

Nous avons ensuite intégré ce correcteur dans le programme python du dirigeable et ajusté manuellement les paramètres du correcteur pour obtenir un comportement stable (mais pas très rapide…).

Nous avons répété cette séquence pour les 4 axes (x, y, z, lacet) avec pour résultat un déplacement lent mais stable et précis du dirigeable.

5.4 - Pistes d'amélioration

Pour que le dirigeable soit autonome, il faut que tous ces axes de déplacement soient correctement asservis. C'est pourquoi une réflexion est en cours pour améliorer le modèle (ordre plus élevé, modèles non linéaires, prise en compte de la portance qui varie en fonction des pertes d'hélium dans la semaine) et pour prendre en compte les 4 axes simultanément pour un asservissement multi-variables.

6 - Conclusion

Lors de ce projet plusieurs domaines de la physique et des sciences de l'ingénieur ont été abordés par les membres d'une équipe regroupée autour des trois problématiques majeures : l'optimisation de masse, la localisation intérieure et l'asservissement. Cela nous a permis d'aboutir à un dirigeable fonctionnel en juin 2025 et de laisser deux pistes d'amélioration principales :

- Deux petits propulseurs verticaux fixés à l'enveloppe au lieu d'un seul gros fixé à la nacelle pour éviter les vibrations de celle-ci et améliorer la dynamique.
- Un travail sur l'asservissement pour améliorer le modèle et mettre en œuvre une correction multi-variables.

Les étudiants de la promotion suivante ont d'ores et déjà repris ces deux thématiques ainsi que la création d'un dock de rechargement en hélium et en électricité pour assurer une réelle autonomie. Un travail est fait également pour limiter le nombre de tags à installer dans le bâtiment de vol du dirigeable.

Références :

[1]: Fundamentals of Aircraft and Airship Design, Volume 2 – Airship Design and Case Studies, G. E. Carichner, L. M. Nicolai, AIAA Education series, february 2013, Coefficient de trainé, p. 69, section 3.3.2)
https://rexresearch1.com/AeroEngineeringLibrary/FundamentalsAircraftAirshipDesignVol2.pdf

[2]: Localisation via la vision avec la bibliothèque Aprilags, G.-A. Fade, A. Juton, novembre 2024, https://sti.eduscol.education.fr/si-ens-paris-saclay/ressources_pedagogiques/localisation-via-la-vision-avecla-bibliotheque-apriltags

[3]: Annexe de la ressource « Réalisation d'un dirigeable autonome », S. Bairat, I. El Kassimi, A. Huet, A. Juton, https://sti.eduscol.education.fr/si-ens-paris-saclay/ressources_pedagogiques/realisation-dun-dirigeable-autonome