

ÉCOLE NORMALE SUPÉRIEURE PARIS-SACLAY

DÉFI SAPHIRE

RAPPORT DE PROJET CoBRA :

Équipe CoBRAPID

Étudiants : Raphaël Barrabes – Emile Bonneville – Antoine Gallissian – Enzo Henry – Louis Poste – Victor Petrovic – Tancrede Wlodyka

Encadrants : Fabien Adam – Bruce Anglade – Bastien Durand – Anthony Juton

Année 2024 – 2025



FIGURE 1 – Dirigeable lors de premiers tests

Table des matières

1	Introduction	3
2	Propulsion du dirigeable	3
2.1	Mesure autour du dirigeable	3
2.1.1	Mesures relatives à l'enveloppe du dirigeable	3
2.1.2	Évaluation des anciennes méthodes de propulsion	3
2.2	Calculs théoriques autour du dirigeable	3
2.2.1	Dimensionnement des moteurs	3
2.2.2	Évaluation du besoin en propulsion verticale	4
2.2.3	Analyse de la stabilité	4
2.3	Choix de la configuration et intégration des moteurs	4
2.4	Choix des moteurs et des variateurs	4
2.5	Validation expérimentale	5
3	Nacelle et supports	5
3.1	Supports des moteurs latéraux	5
3.2	Supports des moteurs d'élévation	6
3.3	Nacelle principale	6
4	Modélisation, contrôle et commande	7
4.1	Contrôle	7
4.1.1	Asservissement en hauteur	7
4.1.2	Asservissement dans le plan	8
4.2	Électronique embarquée	8
4.2.1	Justification de la conception d'une carte dédiée	8
4.2.2	Quelles fonctions doit avoir la carte ?	9
4.2.3	Réalisation et validation de la carte	10
5	Localisation	11
5.1	Introduction	11
5.2	Localisation par vision : AprilTag	11
5.2.1	Principe général	11
5.2.2	Problème PnP	11
5.2.3	Choix de la caméra et de la résolution	12
5.2.4	Robustesse et limites	13
5.3	Hauteur par télémètre TF-Luna	13
5.4	Intégration sous ROS	14
5.5	Conclusion	14
6	Système de préhension	15
6.1	Pince	15
6.2	Treuil	16

6.2.1	Aspect général	16
6.2.2	Modèle 3D	16
6.2.3	Capteur de fin de course du treuil	16
7	Commande numérique	16
7.1	Contrôle de la préhension	17
8	Conclusion	17
9	Annexes	18

1 Introduction

Le défi SAPHIRE propose aux étudiants de concevoir un système autonome répondant à une mission réaliste, dans un environnement contraint et partiellement connu. Cette année, le défi "CoBRA – Course de Ballons Rapides et Automatiques" a pour objectif de concevoir un dirigeable autonome capable d'évoluer en intérieur pour récupérer un colis et le déposer à un emplacement donné, le tout en un temps minimal.

La complexité de ce défi réside dans la maîtrise simultanée de plusieurs domaines : la propulsion d'un dirigeable, la modélisation dynamique, l'asservissement et la localisation, mais aussi la conception mécanique d'un système de préhension et l'intégration électronique et logicielle. Le dirigeable doit se déplacer précisément, saisir un colis, le transporter et le déposer en autonomie complète.

Notre équipe, CoBRAPID, a réparti les différentes responsabilités parmi les membres. Ce rapport détaille notre démarche, nos choix techniques et les résultats obtenus. Chaque grande fonction du dirigeable est présentée dans une section dédiée.

2 Propulsion du dirigeable

2.1 Mesure autour du dirigeable

2.1.1 Mesures relatives à l'enveloppe du dirigeable

Avant toute conception de la propulsion, il était essentiel de caractériser les propriétés physiques de l'enveloppe du dirigeable. Plusieurs mesures ont été réalisées :

- **Volume de l'enveloppe** : mesuré par remplissage à l'hélium, le volume est estimé à $0,17 \text{ m}^3$.
- **Masse totale de l'enveloppe vide** (film + armature) : environ 130 g.
- **Poussée d'Archimède maximale théorique** :

$$F_A = \rho_{\text{air}} \cdot g \cdot V \approx 1,2 \text{ kg/m}^3 \cdot 9,81 \text{ m/s}^2 \cdot 0,17 \text{ m}^3 \approx 2 \text{ N}$$

Cette force permet de supporter environ 200 g de charge utile (propulsion, carte, moteurs, pince). Ces données nous ont permis de définir une **masse cible maximale** de l'ensemble dirigeable à environ 300 g pour assurer une flottabilité quasi-neutre.

2.1.2 Évaluation des anciennes méthodes de propulsion

Dans les éditions précédentes du défi CoBRA, des dirigeables avaient été propulsés :

- soit par deux moteurs horizontaux montés en croix, avec peu de maîtrise sur la rotation,
- soit par propulsion unique centrée, avec orientation par ailerons.

Ces approches souffraient de plusieurs limitations :

- *manque de réactivité* dans les virages,
- *faible manœuvrabilité* en vol stationnaire,
- *forte sensibilité* à la répartition des masses.

Nous avons donc choisi une configuration **quadrimoteur orthogonale** pour améliorer la contrôlabilité dans tous les axes.

2.2 Calculs théoriques autour du dirigeable

2.2.1 Dimensionnement des moteurs

Pour chaque moteur, nous avons utilisé une formule de poussée simplifiée :

$$F = \frac{1}{2} \cdot \rho \cdot S \cdot C_T \cdot \omega^2 \cdot R^2$$

où :

- ρ est la densité de l'air ($\approx 1,2 \text{ kg/m}^3$),
- S est la surface de l'hélice balayée,
- C_T est un coefficient de traction (typ. 0,1 à 0,3),
- ω est la vitesse angulaire,
- R est le rayon de l'hélice.

Chaque moteur, une fois testé avec un dynamomètre, fournit environ 0,6 N de poussée.

2.2.2 Évaluation du besoin en propulsion verticale

En situation de charge maximale, la poussée verticale nécessaire pour la sustentation est donnée par :

$$F_{\text{suppl.}} = (m_{\text{dirigeable}} - m_{\text{Archimède}}) \cdot g \approx 0,1 \text{ kg} \cdot 9,81 \text{ m/s}^2 \approx 1 \text{ N}$$

Soit 0,5 N par moteur vertical. Nous avons respecté ce critère par la suite.

2.2.3 Analyse de la stabilité

Les moteurs verticaux sont positionnés légèrement décalés du centre de gravité pour générer un couple de rappel. L'espacement des moteurs horizontaux assure un couple suffisant pour contrôler le lacet.

2.3 Choix de la configuration et intégration des moteurs

L'architecture retenue repose sur **quatre moteurs brushless** :

- Deux moteurs verticaux pour le contrôle de l'altitude,
- Deux moteurs horizontaux à l'arrière pour le déplacement dans le plan.

Les moteurs brushless sont légers, efficaces, et facilement commandables via PWM. Les moteurs synchrones ont l'avantage de tourner à une vitesse fixe imposée par le variateur indépendamment de la charge (jusqu'au décrochage), donc sont particulièrement adaptés au modélisme et au pilotage automatisé de notre dirigeable.

L'architecture choisie du dirigeable nous force à utiliser des hélices pas trop imposantes, d'une part pour réduire l'encombrement, mais aussi la masse et l'ampérage dans les câbles et la carte. Nous choisissons alors des hélices de type 5x5 de 5 pouces de diamètre et de pas.

Afin d'éviter le roulis et le tangage induits lors des déplacements du dirigeables, nous choisissons de commander les paires de moteurs dans des sens opposés ; on choisit alors de prendre deux moteurs à pas négatif sur les quatre. Chaque hélice pèse 5g. Cette disposition offre un bon compromis entre masse embarquée, stabilité et manœuvrabilité.

2.4 Choix des moteurs et des variateurs

Une fois le type de moteurs et leur architecture choisie, nous devions choisir le modèle exact de moteurs et des variateurs. Nous avons une contrainte de masse importante, il faut donc prendre des moteurs plus légers que ceux de la solution précédente, sans compromettre la poussée.

Grâce à un essai réalisé sur l'ancien moteur, on remarque que le couple que l'on peut générer est bien supérieur au couple équivalent généré par la force de poussée que permettent les hélices 5x5 à leur vitesse de rotation max. On choisit donc de prendre des moteurs moins puissants mais en augmentant le KV : on choisit alors un KV de 2300. Ce moteur satisfait notre cahier des charges, pour une masse de seulement 17 grammes.

On accompagne ce moteur d'un variateur capable d'encaisser la puissance requise (60W, 8,4A max) et capable d'inverser le sens de rotation. On opte pour un modèle de 8,5g qui permet d'avoir alors une solution de propulsion à 30g par convertisseur, soit 120g au total.

Type	Propeller	Throttle	Voltage (V)	Current (A)	Power (W)	RPM	Torque (N*m)	Thrust (g)	Efficiency (g/W)	Operating Temperature (°C)
AS2303 Short Shaft KV2300	GWS 7035	40%	7.52	1.64	12.30	7064	0.011	100	8.17	74 (Ambient Temperature)
		45%	7.54	2.06	15.56	7611	0.013	119	7.61	
		50%	7.51	2.45	18.43	8135	0.014	134	7.29	
		55%	7.46	2.84	21.17	8504	0.016	151	7.13	
		60%	7.47	3.23	24.10	8873	0.017	167	6.93	
		65%	7.48	3.77	28.21	9382	0.019	189	6.69	
		70%	7.43	4.41	32.77	9869	0.021	212	6.47	
		75%	7.40	5.28	39.09	10479	0.024	243	6.21	
		80%	7.33	6.09	44.60	11032	0.026	268	6.01	
		90%	7.19	8.10	58.25	12023	0.031	325	5.58	
		100%	7.18	8.40	60.32	12186	0.032	331	5.49	

Note: Motor temperature is motor surface temperature @100% throttle running 3mins.
(Data above based on benchtest are for reference only, comparison with that of other motor types is not recommended.)

FIGURE 2 – Documentation moteur

2.5 Validation expérimentale

Des essais en salle ont permis de :

- Mesurer la poussée réelle par moteur,
- Vérifier l'équilibre statique du dirigeable (flottabilité à vide et en charge),
- Ajuster le positionnement des moteurs,
- Tester la réponse du système à des commandes impulsionales.

Ces tests ont validé les choix de conception et confirmé que la configuration permettait une manœuvrabilité suffisante pour la mission.

3 Nacelle et supports

3.1 Supports des moteurs latéraux

Les moteurs latéraux étant soumis à des efforts verticaux (poids), horizontaux (poussée) et en couple, il faut une structure assez rigide pour tenir le moteur sans qu'il tombe ou ne bouge lors de l'actionnement, aussi brutale que puisse être la commande. En outre, pour des raisons évidentes de sécurité ainsi que pour ne pas endommager l'hélice en cas de collision, il faut y ajouter un bumper de protection. Par des essais sur la solution constructeurs, nous avons conclu précédemment que l'effet d'un carter était négligeable sur le guidage de l'air, donc on choisit de réaliser un bumper très creux.

La solution retenue à imprimer en 3D est celle-ci :

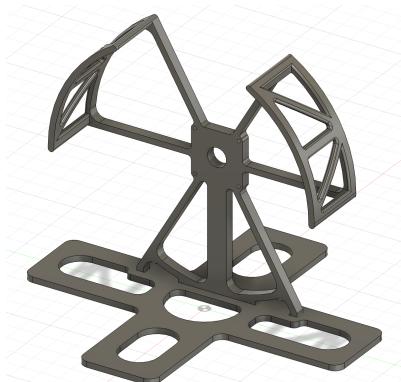


FIGURE 3 – Supports moteurs latéraux

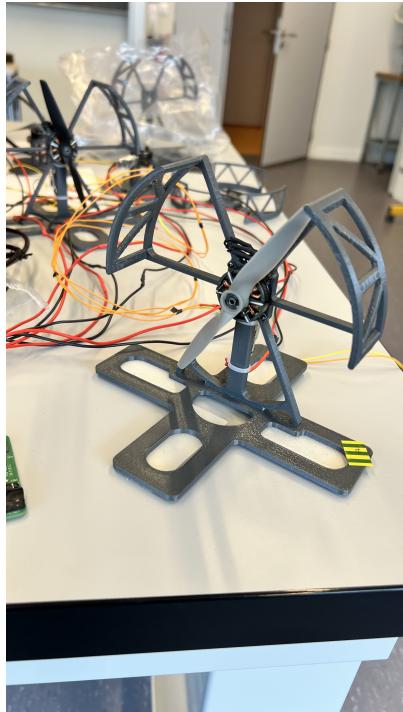


FIGURE 4 – Photo du support monté avec le moteur

3.2 Supports des moteurs d’élévation

On fait de même pour les moteurs du bas, mais ceux-là ne sont pas soumis à autant de forces. On peut donc simplifier la structure en les aplaniissant.

Expérimentalement, on se rend compte que la trop grande proximité du moteur avec l’enveloppe du blimp, entraîne des effets aérodynamiques néfastes en raison de la couche limite qui interfère avec le flux d’air. Lorsque le moteur est censé faire monter le dirigeable, il crée en fait une dépression qui le fait chuter. On envisage alors une solution à piliers, où le moteur est surélevé de 4cm pour faire passer le flux d’air.

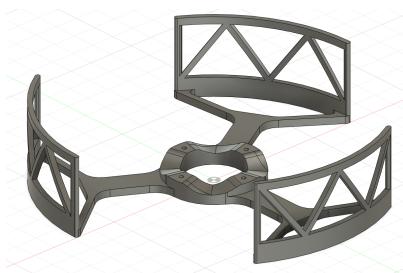


FIGURE 5 – Supports moteurs du bas non corrigés

3.3 Nacelle principale

Enfin, on passe au montage complet de toutes les parties du dirigeable. On rappelle qu’il faut intégrer :

- La carte Raspberry Pi,
- Le ventilateur pour refroidir la Raspberry,
- La carte PCB imprimée contenant les drivers pour les moteurs,
- Le treuil,
- La batterie,
- La caméra.

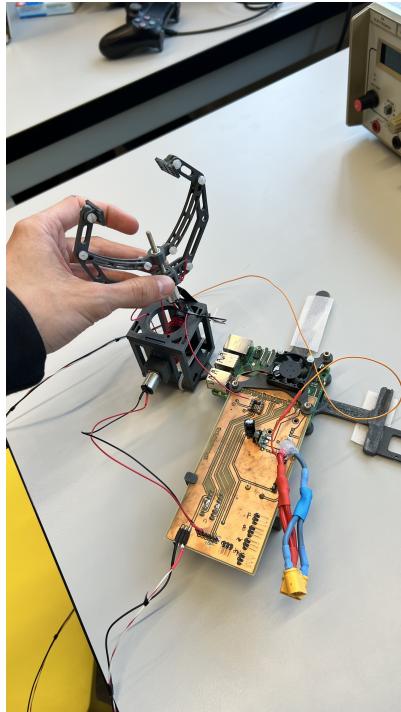


FIGURE 6 – Nacelle principale

On impose quelques contraintes :

- le ballon doit être équilibré donc on doit rendre à peu près symétrique la répartition des masses,
- la caméra doit être centrée par rapport au ballon pour faciliter le positionnement,
- le treuil doit être centré par rapport au ballon pour ne pas le déséquilibrer en tractant la charge.

On choisit donc cette solution imprimable en 3D :

4 Modélisation, contrôle et commande

4.1 Contrôle

Sans asservissement, le contrôle précis du dirigeable est impossible en raison de son inertie importante. En vu de son automatisation, l'établissement de lois de commande est indispensable pour que le dirigeable puisse atteindre son objectif et puisse maintenir une position constante le temps de récupérer le colis.

Le modèle physique du dirigeable est relativement complexe et établir l'expression numérique de certaines forces qui dépendent de la géométrie peut s'avérer fastidieux et nécessiter l'identification de nombreux paramètres. Pour obtenir un modèle simplifié pour permettre la mise en place d'une loi de commande, nous allons faire quelques hypothèses.

L'hypothèse principale qui peut être faite en raison de l'architecture de propulsion retenue est que le dirigeable peut être mis en mouvement de manière indépendante selon chacun de ses degrés de liberté, ce qui permet d'établir une loi de commande simplifiée pour chaque degré de liberté. On suppose également le temps de réponse du moteur négligeable vis à vis de la dynamique du dirigeable.

4.1.1 Asservissement en hauteur

La dynamique dans la direction verticale peut donc être considérée indépendante des autres directions. On prend en compte les forces suivantes :

- \vec{F}_p : force de propulsion des deux moteurs orientée selon l'axe vertical
- \vec{P} : forces de gravitations
- Les forces de frottement de l'air proportionnelles à la vitesse verticale

— La poussée d'Achimède supposé constante et opposée au poids.
On obtient donc l'équation suivante dans la direction \vec{z} :

$$\ddot{z} = az + bu$$

Avec u la tension aux bornes des moteurs et a, b des constantes à identifier. La fonction de transfert du dirigeable dans le domaine de Laplace est donc :

$$H(p) = \frac{Z(p)}{U(p)} = \frac{b}{p(p - a)}$$

Les paramètres a et b sont donc identifiables en réalisant des essais indicuels.

4.1.2 Asservissement dans le plan

Dans cet article, [1] l'auteur utilise un modèle simplifié pour modélisé le comportement d'un dirigeable d'intérieur. Pour obtenir les équations dynamiques du dirigeable dans le plan (\vec{x}, \vec{y}) on applique le PFD dans ce plan et on obtient 3 équations. En notant ψ l'angle de lacet du dirigeable, on obtient les équations

$$\begin{cases} \ddot{x} = c_\psi bu + \kappa_1(a_x, a_y, \psi)\dot{x} + \kappa_3(a_x, a_y, \psi)\dot{y} \\ \ddot{y} = s_\psi bu + \kappa_2(a_x, a_y, \psi)\dot{y} + \kappa_3(a_x, a_y, \psi)\dot{x} \\ \ddot{\psi} = b_\psi v + a_\psi \dot{\psi} \end{cases} \quad (1)$$

Avec $u = u_{\text{gauche}} + u_{\text{droite}}$ et $v = u_{\text{gauche}} - u_{\text{droite}}$

$$\begin{aligned} \kappa_1(a_x, a_y, \psi) &= a_x \cos^2 \psi + a_y \sin^2 \psi, \\ \kappa_2(a_x, a_y, \psi) &= a_y \cos^2 \psi + a_x \sin^2 \psi, \\ \kappa_3(a_x, a_y, \psi) &= a_x \cos \psi \sin \psi - a_y \cos \psi \sin \psi, \end{aligned} \quad (2)$$

Avec a_x, a_y, a_ψ, b_ψ et b des coefficients à identifier.

Les coefficients a_x et a_y sont liés aux frottements pour les mouvements en translations, les paramètres a_ψ et b_ψ sont liés aux frottements lors des mouvements de translation. Le coefficient b est lui identifié lors de la réponse à un échelon des moteurs latéraux.

4.2 Électronique embarquée

4.2.1 Justification de la conception d'une carte dédiée

La carte Raspberry Pi 5, choisie comme ordinateur embarqué, ne permet pas de contrôler directement l'ensemble des moteurs ni de recevoir toutes les données des capteurs. Il a donc été nécessaire de concevoir et réaliser une carte électronique dédiée, capable de gérer l'acquisition des données capteurs et le pilotage des moteurs.

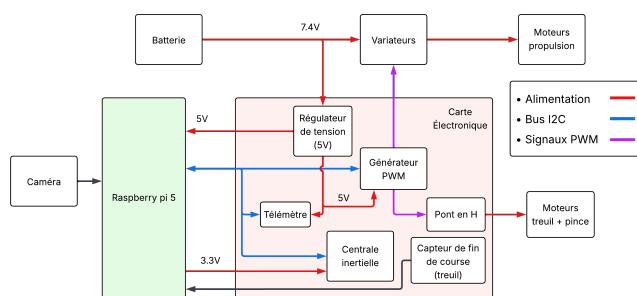


FIGURE 7 – Schéma de principe de la carte électronique

4.2.2 Quelles fonctions doit avoir la carte ?

Dans la solution que nous proposons pour répondre à l'objectif du projet COBRA, notre dirigeable sera propulsé avec 4 moteurs, et il sera équipé de 2 moteurs pour s'occuper de la partie préhension. Il devra aussi être équipé de différents capteurs afin de se repérer dans l'espace, mais aussi pour être apte à récupérer le colis. Dans cette partie, nous nous intéresserons donc aux moyens que nous avons mis en place pour que notre carte valide ces différentes fonctions.

Moteurs de propulsion Les moteurs utilisés pour la propulsion sont des moteurs brushless, leurs commande sera donc assurée par l'envoie d'un signal PWM. La Raspberry pi 5 seule ne pouvant pas fournir des signaux assez stable, il sera donc nécessaire d'ajouter à notre carte un générateur de signaux PWM afin de pouvoir commander tous les moteurs.

L'alimentation des moteurs de propulsion nécessitant une tension de 7,4V, il ne sera pas nécessaire pour eux d'utiliser un régulateur de tension. Ils seront donc "directement" branché à la batterie qui fournit une tension de cet ordre de grandeur.

Moteur de la pince Le moteur de la pince est un MCC, il est donc nécessaire de pouvoir inverser le courant en ses bornes afin de pouvoir choisir son sens de rotation (ouverture ou fermeture des pinces). Pour ce faire, nous utilisons un pont en H qui permet le contrôle de la polarité aux bornes d'un dipôle grâce à une ouverture/fermeture astucieuse d'interrupteurs, comme nous pouvons le voir sur la figure suivante.

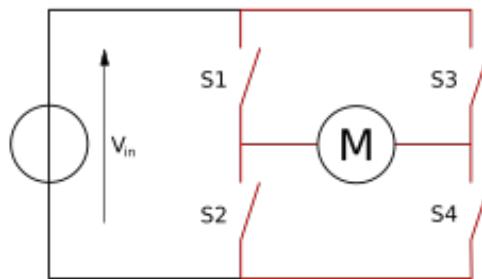


FIGURE 8 – Contrôle d'une MCC avec un pont en H

En plus de nécessiter une commande différente des moteurs de propulsions, le moteur servant au contrôle de la pince fonctionne sous une plage de tension comprise entre 3V et 6V, il était donc nécessaire d'ajouter un régulateur de tension afin de ne pas lui imposer une tension trop importante.

Finalement, la dernière fonction que doit remplir notre pince est celle de détecter quand elle a attrapé notre colis. Pour ce faire, nous envisageons d'utiliser le fait que, lorsque le colis sera attrapé, le moteur va forcer pour continuer à avancer et devra donc utiliser un courant plus important. Pour détecter le moment où le courant est plus important, nous utiliserons un montage comparateur avec un amplificateur-opérationnel. Dans la version de la carte utilisée, le comparateur de courant n'est pas fonctionnel donc la détection de la préhension du colis sera faite avec un chronomètre qui mesure le temps de fermeture.

Moteur du treuil De la même manière que le moteur de la pince, le moteur servant à piloter le treuil est une MCC, il nécessitera donc lui aussi un pont en H pour contrôler son sens de rotation (monter ou descendre le treuil). Il nécessitera lui aussi un régulateur de tension car sa tension de fonctionnement est inférieure à celle fournie par la batterie.

Afin de pouvoir détecter quand le treuil est assez bas pour attraper le colis, il est aussi nécessaire qu'il dispose d'un encodeur lui permettant de savoir son sens de rotation ainsi que sa vitesse de rotation afin de pouvoir savoir où il en est de sa descente/montée. Le moteur choisi étant directement fourni avec cet encodeur, nous devrons ajouter un connecteur à notre carte permettant de :

- piloter le sens de rotation du moteur
- alimenter l'encodeur
- envoyer à la carte Raspberry les informations sur la vitesse de rotation
- envoyer à la carte Raspberry les informations sur le sens de rotation.

TF lunna Afin de pouvoir savoir à quelle hauteur se situe notre dirigeable, nous utiliserons un télémètre, directement branché sur notre carte afin d'envoyer ses données à la Raspberry.

BNO055 Il sera aussi nécessaire de savoir l'orientation et l'accélération de notre dirigeable. Cette fonction sera assurée par le BNO055 qui a comme fonction celle d'accelerometre, de gyroscope et de magnetometre. Ce capteur sera lui aussi branché à notre carte et communiquera avec la Raspberry par bus I²C.

Régulateur de tension Le système étant autonome, tous les composants sont alimentés par une batterie de 7,4 V. Un régulateur de tension (OKL-T/6-W12N-C) génère une tension stable de 5 V pour alimenter la Raspberry Pi et les autres composants alimentés en 5V. Les composants alimentés en 3,3V sont alimentés directement par la Raspberry.

Branchement

Une fois les branchements de chaque composants effectués, nous pouvons nous occuper de définir les pistes qui permettront de faire la jonction avec notre Raspberry, comme nous pouvons le voir sur la figure ci dessous. Les pistes qui supportent des courants de plusieurs Ampères pour alimenter la carte Raspberry et le pont en H font 2,54 millimètres de large tandis que les pistes qui supportent de faibles puissances font 0.6096 mm de large. Les pistes nommées SDA et SCL supportent le transport d'informations entre les composants via le bus I²C. Le signal SCL est le signal d'horloge qui synchronise les données transférés par le signal SDA. Des condensateurs de découplage ont été ajoutés à proximité des ports d'alimentation afin de lisser la tension délivrée.

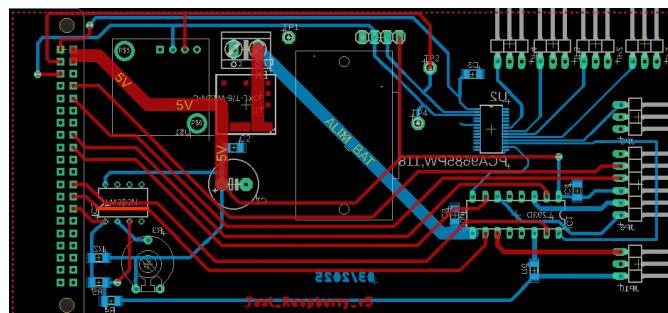


FIGURE 9 – Schéma du PCB avec matérialisation des connexions



FIGURE 10 – Visualisation 3D du PCB

4.2.3 Réalisation et validation de la carte

On réalise le circuit imprimé par un procédé de gravure chimique. Certaines zones d'une carte recouverte de cuivre et d'une couche de résine photosensible sont recouverte par un typon obtenu avec le logiciel de conception.

On expose ensuite la carte à la lumière UV : seules les zones non protégées par le typon sont insolées. Après développement, la résine est retirée des zones exposées. Ces parties seront attaquées par un bain de perchlorure de fer, qui dissout le cuivre non protégé. Les composants traversants, les composants montés en surface (CMS) et les connecteurs ont ensuite été installés.

Pour valider la carte, des tests de continuité ont été réalisés afin de vérifier l'ensemble des soudures et de s'assurer de l'absence de courts-circuits. La tension sur les pistes d'alimentation a ensuite été mesurée, puis l'allure des signaux sur le bus I²C ainsi que celle des signaux PWM générés a pu être vérifiée à l'aide des points de sonde prévus à cet effet.

5 Localisation

5.1 Introduction

Dans le cadre de notre projet de dirigeable autonome évoluant en intérieur, la problématique de la **localisation** s'est avérée centrale. Nous avons opté pour une solution reposant sur des marqueurs visuels (*AprilTag*) associés à une estimation de hauteur fournie par un capteur de distance TF-Luna. L'ensemble est intégré dans l'environnement **ROS**, permettant l'observation en direct et l'ajustement dynamique des paramètres de vol. Ce rapport se concentre sur les aspects techniques de la localisation.

5.2 Localisation par vision : AprilTag

5.2.1 Principe général

Le système repose sur des balises visuelles nommées **AprilTags**, disposées à des positions connues dans un repère global. Une caméra embarquée sur le dirigeable observe ces tags. Leur détection permet, via un solveur PnP (*Perspective-n-Point*), de déterminer la pose de la caméra (et donc du dirigeable).

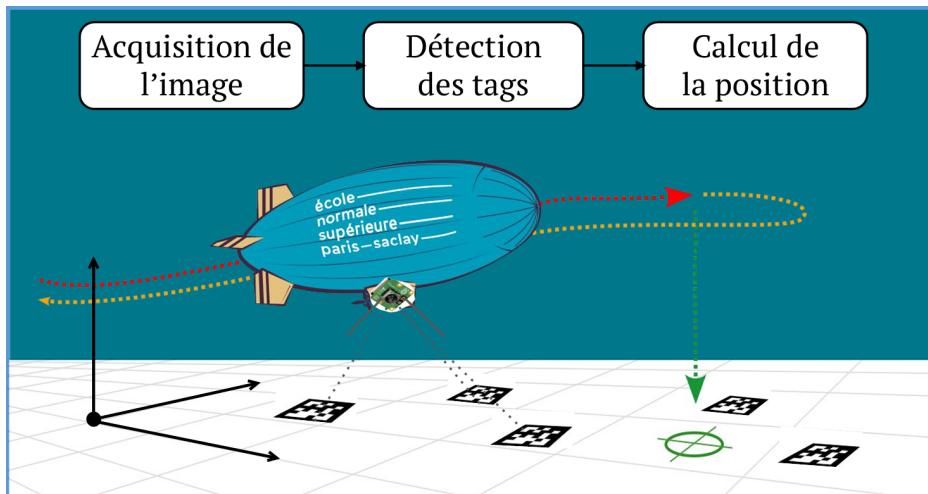


FIGURE 11 – Principe de localisation avec AprilTags

Nous utilisons la bibliothèque **RobotPy-AprilTag**, un wrapper Python autour de la librairie C **AprilTag**. L'identification de tags fournit les coordonnées des coins dans l'image, que l'on associe aux positions 3D connues. La fonction **solvePnP** d'OpenCV permet ensuite d'estimer la pose de la caméra.

5.2.2 Problème PnP

Le problème PnP, pour Perspective-n-Point, est une modélisation de la projection des points des tags dans le repère 3D du dirigeable sur l'image 2D captée par la caméra. Tout l'enjeu consiste à faire le chemin inverse plusieurs fois par seconde, c'est-à-dire à, à partir de la position des tags dans l'image capturée par la caméra, et à partir de

la position des tags dans le repère 3D dans lequel évolue le dirigeable, remonter à la position du dirigeable dans ce même repère 3D. Le problème peut se formuler ainsi :

$$s \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \quad (3)$$

avec :

- u, v , les coordonnées d'un point du tag sur l'image 2D ;
- x, y, z , les coordonnées du même point dans le repère 3D ;
- f_x, f_y, c_x, c_y , les paramètres optiques de la caméra, et s un facteur d'échelle ;
- t_x, t_y, t_z, r_{ij} , la position de la caméra dans le repère 3D, en translation et rotation.

La résolution du problème Pnp se fait grâce à l'algorithme EPnP [2], implémenté dans la bibliothèque OpenCV.

5.2.3 Choix de la caméra et de la résolution

Nous disposons de deux nano-ordinateurs et de 3 caméras pour l'implémentation de la localisation :

- Raspberry Pi Zero 2 W
- Raspberry Pi 5
- Pi Camera module V2
- Pi Camera module V3
- Pi Camera module V3 Wide

Afin de maximiser le fréquence de rafraîchissement de la position, tout en conservant une bonne précision, nous avons opté pour un Raspberry Pi 5 avec à une résolution d'image relativement faible de 640 par 480 pixels.

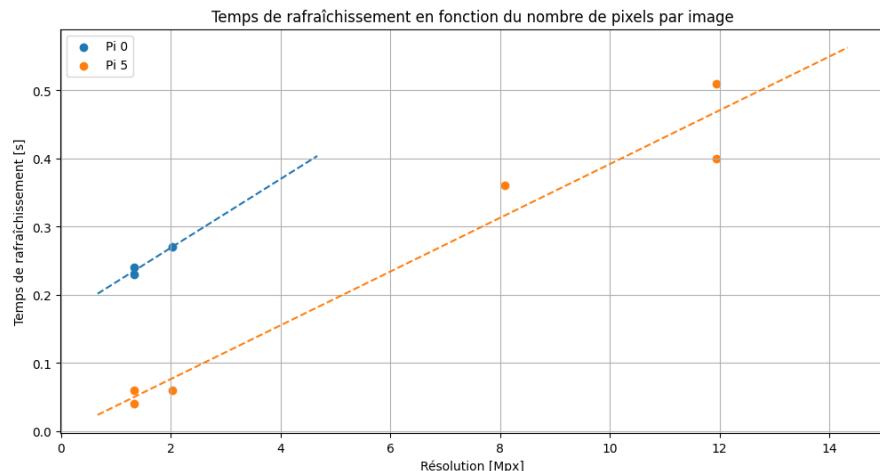


FIGURE 12 – Temps de rafraîchissement en fonction du nombre de pixels par image

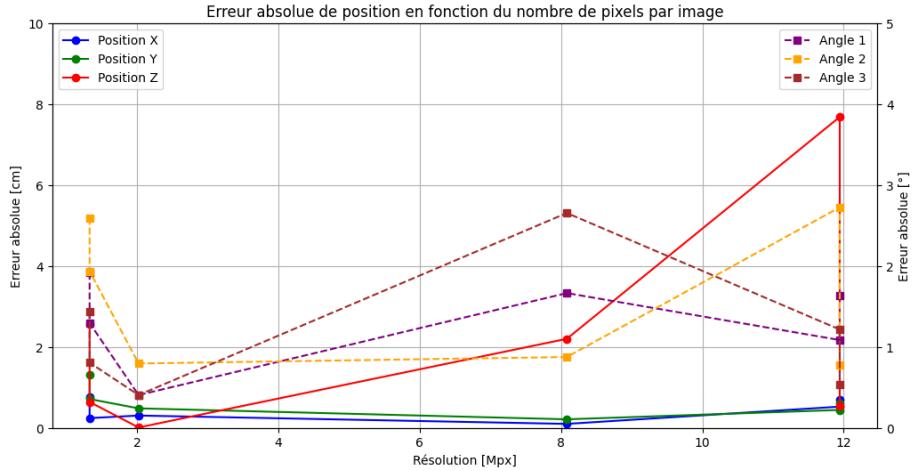


FIGURE 13 – Erreur absolue en fonction du nombre de pixels par image

Nous avons fait le choix de la Pi Camera module V2, car elle offre une précision légèrement meilleure que les autres modules, et que nous avons constaté qualitativement une présence moins importante de bruit numérique sur les images.

5.2.4 Robustesse et limites

La précision dépend fortement de :

- la taille apparente du tag dans l'image
- l'angle d'observation
- la qualité de la calibration
- la densité de tags au sol

Nous avons constaté que des tags trop petits (inférieurs à 4 cm) ou observés sous de trop grands angles mènent à des erreurs importantes. Par ailleurs, la projection des tags dans l'espace image devient plus sensible aux erreurs de calibration à courte distance.

De plus, on remarque lors de nos tests qu'une trop faible densité de tags entraîne une perte de signal à faible altitude, d'où la nécessité d'avoir une certaine densité pour toujours avoir un deux tags minimum dans le champ de vision de la caméra.

5.3 Hauteur par télémètre TF-Luna

Afin de compléter la localisation 3D, un capteur TF-Luna est utilisé pour mesurer la hauteur au sol. Ce capteur Lidar à temps de vol fournit une mesure directe de distance avec une fréquence suffisante pour le contrôle du vol.



FIGURE 14 – Image du capteur TF-Luna

La fusion des données caméra + lidar permet une localisation complète (x, y, z) du dirigeable.

Cependant, après plusieurs essais, nous nous sommes rendu compte que le système *AprilTag* était suffisamment précis dans son estimation de la hauteur pour se suffire à lui-même. Nous nous limiterons donc dans un premier temps à l'utilisation des Tags seuls.

5.4 Intégration sous ROS

L'ensemble du système est intégré dans l'environnement **Robot Operating System (ROS)**. ROS est un environnement complet de bibliothèques conçu pour la robotique et permettant la création de noeuds fonctionnant en parallèle et s'échangeant des informations appelées **topics**.

- Les images de la caméra sont traitées par deux noeuds ROS appelant directement du code compilé en C des bibliothèques OpenCV et AprilTags pour de meilleures performances
- La position estimée est publiée sur un topic ROS
- L'estimation des positions provenant des différents tags sont ensuite fusionnées dans un noeud dédié, en appliquant une moyenne
- Un noeud d'asservissement permet la commande des moteurs à partir des données de localisation
- Enfin, un noeud "maître", à implémenter, permettra de fournir les consignes de trajectoire et gérera les étapes de la mission : prise du colis, montée du treuil, vol vers la zone de dépôt du colis...

Nous avons volontairement plafonné la fréquence de rafraîchissement à 10 Hz, alors qu'il serait possible d'avoir une fréquence plus élevée, pour être sûrs d'avoir une fréquence stable pour l'asservissement.

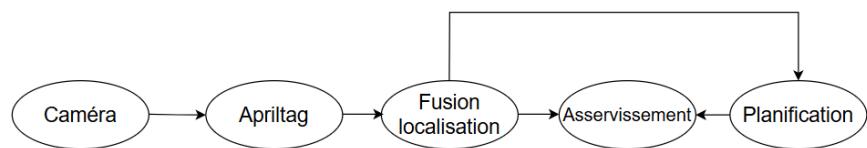


FIGURE 15 – Schéma des noeuds ROS

Il est possible de visualiser les données des différents topics, sur un logiciel dédié nommé Rviz. La visualisation de la position en temps réel est facilité grâce à un environnement 3D.

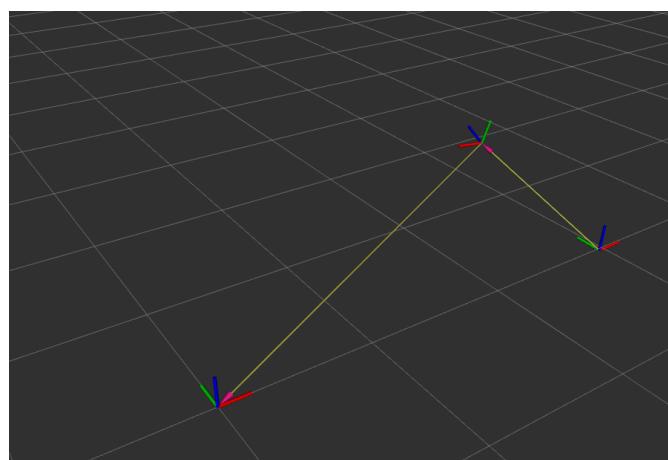


FIGURE 16 – Visualisation de la position du dirigeable en temps réel dans l'environnement 3D de Rviz

5.5 Conclusion

Nous avons finalement retenu la solution des Apriltags, pilotée par ROS. Cela constitue une solution efficace de localisation en intérieur. Malgré certaines limitations (sensibilité à l'éclairage, densité importante des tags au

sol), cette solution est suffisante pour du guidage et de la stabilisation dans un espace contrôlé. Comme piste d'amélioration, il pourrait être intéressant d'utiliser deux caméra dans des orientations différentes, afin de maximiser le nombre de tags visibles, et ainsi de pouvoir améliorer la robustesse de la localisation tout en réduisant la densité de tags au sol.

6 Système de préhension

6.1 Pince

Des différents modèles de pinces, nous avons retenu le modèle de la pince à deux bras, qui permet de limiter la masse tout en assurant le transport du colis.

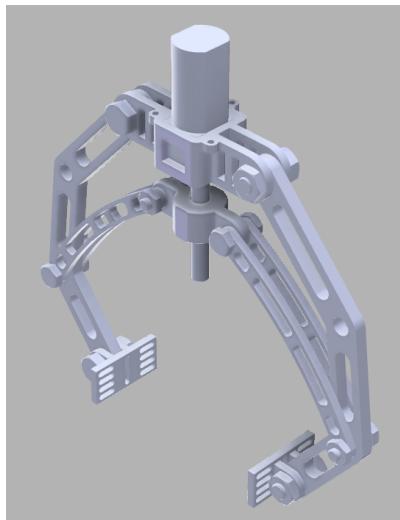


FIGURE 17 – Modèle 3D de la pince

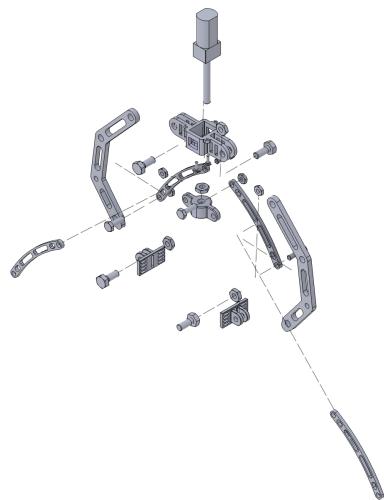


FIGURE 18 – Modèle éclaté de la pince

On souhaite utiliser un moteur MCC qui va faire tourner un axe taraudé, qui va faire translater un écrou bloqué en rotation. Les bras intérieurs vont rétracter les bras extérieurs et refermer la pince.

La possibilité de mesurer le courant a été amenée, mais au vu des difficultés des autres groupes à mettre en place cette solution pour asservir l'ouverture de la pince, nous avons décidé de mesurer le temps de fermeture avec le cube à délivrer pour l'incorporer dans le programme.

Nous avons choisis pour le matériau du plastique imprimé 3D dans le but de minimiser la masse et les coûts, et les vis écrous utilisés pour maintenir la pince sont également en plastique.

Pour garantir une certaine stabilité lors de la descente de la pince, 3 fils en corde attachés en triangle permettent d'équilibrer la descente et la montée, mais également d'améliorer l'enroulement. Cependant, le problème des câbles électriques du moteur de la pince s'est posé : ceux-ci ne peuvent pas être enroulés au vu du risque de rupture interne du fil lié à la torsion. Cependant, puisque notre solution de propulsion n'utilise pas de moteur directement sous le dirigeable, nous avons choisi, au vu du critère de minimisation de la masse, de simplement laisser pendre le câble.

6.2 Treuil

6.2.1 Aspect général

Le treuil sera également piloté par un MCC. Celui-ci sera directement relié à un arbre enrouleur, dans une structure cubique. On souhaite limiter le couple nécessaire pour le moteur, tout en faisant en sorte que l'arbre ne soit pas de rayon trop faible pour éviter les recouvrements de fils. On cherche donc à minimiser R_{arbre} tout en respectant un R_{min} lié au nombre de tours de l'arbre.

Le cahier des charges impose une descente de minimum 1 m. On a la relation :

$$L_{descende} = N_{b\ tours} * 2\pi R$$

On limite le nombre de tours à 10, ce qui impose $R > 15\ mm$. Dans l'optique de prendre une marge pour la conception 3D, on choisit finalement $R = 20\ mm$, que l'on pourra poncer pour diminuer les frottements.

Le modèle technique du moteur est fourni en annexe.

6.2.2 Modèle 3D

Cube de 50 mm, Arbre creux de rayon $R = 20\ mm$ et de longueur $L = 40\ mm$ On choisit de poncer l'arbre pour diminuer les frottements. Une des bases est fixée au support global, avec des vis, et la base opposée est munie de 3 trous pour les 3 fils de la pince.

On a choisi de ne pas utiliser de collecteur pour les câbles d'alimentation de la pince dans le but d'économiser de la masse, sachant que notre solution de propulsion ne possède pas de moteurs dans cette zone, ce qui empêche des problèmes de câbles flottants.

Le moteur est maintenu en position par les dimensions du boîtier du moteur.

6.2.3 Capteur de fin de course du treuil

Nous avons fait le choix de déterminer le début et la fin de course à l'aide d'un interrupteur fabriqué avec 2 fils : lorsque les bouts métalliques passent dans l'un des trous au niveau du treuil, ils ferment un circuit qui permet de déterminer l'arrêt du treuil.

7 Commande numérique

Pour tester les principales fonctions du dirigeables, nous utilisons python pour attribuer les commandes d'une manette de PS4 aux actions des moteurs. Pour cela nous utilisons la bibliothèque pyPS4controller, qui permet d'assigner la manette à des actions. La manette est appareillée en bluetooth à la raspberry.

On utilise également une classe PCA9685, pour associer les différents moteurs aux pins de la PCB, et pouvoir définir comment générer les signaux PWM, et une classe pour chaque type de moteur.

Ainsi pour associer les boutons de la manette à des actions, on utilise la forme suivante :

```

def on_L3_right(self,value): #tourner à droite

    if value<0 :
        value = 0
    value = value*10/3200
    if value >100 :
        value = 100
    print("La valeur de L3 est: ",value)
    self.vitesse_diff = value
    brushless["g"].cmd_vit_pourcent(self.vitesse_moy-self.vitesse_diff)
    brushless["d"].cmd_vit_pourcent(self.vitesse_moy+self.vitesse_diff)

```

FIGURE 19 – Élément de code pour contrôler le dirigeable.

"On L3 right" définit le bouton utilisé, le premier if sert à contrôler la zone morte, et le second à saturer la commande pour éviter que les moteurs reçoivent une commande non traitable. Enfin, les commandes "Brushless" servent à commander les moteurs.

On utilise un système de vitesse moyenne et différentielle pour commander le dirigeable : la vitesse moyenne prends de 0 à 50 % de la puissance des moteurs latéraux, et les 50 % restants servent pour la vitesse différentielle. Lorsqu'on avance, les moteurs sont à 100 % en vitesse moyenne, et lorsqu'on veut tourner, la vitesse différentielle vient créer une différence de vitesse qui fait tourner le dirigeable dans le sens désirer.

Nous avons également mis un système d'arrêt d'urgence et de mise en route, qui permet à l'aide d'un simple bouton de couper tous les moteurs.

Le treuil et la pince sont commandés par la croix directionnelle.

7.1 Contrôle de la préhension

Afin de commander les deux moteurs, nous nous servons d'un hacheur 4 quadrants (L293D) implantés sur la carte PCB. Nous avons choisi ce composant parce qu'il permet de contrôler deux moteurs dans des sens différents, indépendamment. Pour chaque moteur contrôlé, il prend en entrée un signal PWM (entrée « 1,2EN ») modulant la sortie du hacheur et deux signaux booléens (« 1A » et « 2A ») commandant le sens du moteur selon le tableau suivant :

EN	1A	2A	sens de rotation
H	L	H	sens horaire
H	H	L	sens antihoraire

Les deux moteurs étant contrôlés de la même façon, on s'intéresse ici identiquement à l'un ou à l'autre. On génère donc le signal PWM à partir du contrôleur I2C PCA9685 et on génère les signaux booléens commandant le sens de rotation avec les ports GPIO de la Raspberry. Les flèches de la télécommande envoient donc conjointement le signal PWM à l'entrée et le bit correspondant au sens de rotation en entrée 1A ou (exclusif) en entrée 2A du L293D. Le L193D génère donc le signal haché correct.

8 Conclusion

La solution développée permet de répondre à l'ensemble des fonctions attendues tout en respectant la contrainte sur la masse que l'enveloppe peut supporter. L'architecture de propulsion permet de mettre le dirigeable en mouvement selon tous les degrés de liberté de manière indépendants ce qui facilite la commande. Le mécanisme de préhension développé permet de récupérer et de déposer le colis de manière fiable. La carte électronique conçue assure l'alimentation de l'ensemble des composants et permet l'échange d'informations entre les capteurs, les action-

neurs et la Raspberry Pi. Néanmoins, un travail sur ses dimensions peut encore être mené. Il nous reste maintenant à développer un asservissement fiable et à créer la logique permettant la planification des étapes de la mission (prise du colis, déplacement, dépôt du colis, et retour) afin que le dirigeable soit complètement autonome.

9 Annexes

Moteurs du système de préhension :

- <https://www.lextronic.fr/motoreducteur-miniature-1210m4gm06100-76654.html>
- <https://www.gotronic.fr/art-motoreducteur-lp-298-1-5121-39017.htm>

Réalisation de la carte sous Autodesk Eagle :

