



ENGINEERING SCHOOL
Creating the future together

Rapport du projet de semestre

Drone automatique

Promotion : P2025

Classe : AE1

Groupe : GPD1

Membres du groupe : Nicolas ABAT, Alexandre CASTEX, Thomas FORTUNATO,
Joachim KOEHL, Constance NGUYEN, Julien TAKTALIAN

Projet supervisé par : Thierry HALCONRUY

Table des matières

Table des matières	1
Introduction.....	6
Conception fonctionnelle	8
DCU.....	8
DCS	9
Diagramme des transitions	10
Scénarios opérationnels	10
Exigences opérationnelles fonctionnelles.....	12
Logigrammes	16
Conception technique	25
Modèle 3D	25
Hypothèses de modélisation	25
Description du modèle 3D.....	25
Analyse des distances clés.....	27
Détermination des matrices d'inertie	27
Détermination des surfaces projetées	29
Simulations numériques sous le logiciel Simulink.....	30
Structure du modèle.....	30
Détermination des constantes physiques	31
Mise en place du modèle complet	32
Validation du modèle	33
La simulation avec imperfections.....	34
Choix technologique.....	35
Arborescence fonctionnelle	37
Boîte grise.....	39
Exigences techniques	42
Exigence technique liée à l'acquisition.....	42
Exigences technique liée au traitement	42
Exigence technique liée à la conversion électrique/lumineuse	44
Exigence technique liée à la conversion électrique/mécanique.....	44
Exigence technique liée au stockage.....	45
Exigence technique de distribution.....	46

Exigence technique liée à la réception et à l'émission.....	46
Tableau des exigences opérationnelles techniques.....	47
Conception physique.....	50
Exigences physiques.....	50
Composants de la base volante initiale.....	50
Exigences physiques	51
Justification des exigences physiques	51
Test de la chaîne de puissance	53
Exigences et Critères de Réussite.....	53
Composants Testés.....	54
Méthode d'Essai.....	54
Conditions de l'Essai	54
Procédure de test	54
Collecte des données.....	54
Validation des résultats	54
Conclusion	54
Conception du support.....	56
Version 1.....	56
Version 2.....	57
Version finale du support	58
Nouvelle évaluation de l'autonomie et de la manœuvrabilité avec la version complète du système	66
Schémas électriques.....	67
Tests physiques	70
Test de la base volante	70
Objectif d'Essai	70
Exigences et Critères de Réussite	70
Composants Testés.....	70
Méthode d'Essai.....	71
Conditions de l'Essai	71
Procédure de test	71
Collecte des données.....	71
Validation des résultats	71
Conclusion	72
Test d'acquisition des coordonnées dans l'espace	74

Objectif d'Essai	74
Exigences et Critères de Réussite	74
Composants Testés.....	74
Méthode d'Essai.....	74
Conditions de l'Essai	75
Procédure de Test	75
Collecte des Données	75
Validation des Résultats	75
Conclusion	75
Test affichage et acquisition de la distance à parcourir.....	77
Exigences et Critères de Réussite	77
Composants Testés.....	77
Méthode d'Essai	77
Conditions de l'Essai	77
Procédure de test	78
Collecte des données.....	78
Validation des résultats	78
Conclusion	78
Tests fonctionnels	79
Test de commutation et de réception des signaux	79
Objectif d'Essai	79
Exigences et Critères de Réussite	79
Composants Testés.....	79
Méthode d'Essai	80
Conditions de l'Essai	80
Procédure de test	80
Collecte des données.....	81
Validation des résultats	81
Conclusion	81
Test d'autonomie et de vitesse de stabilisation en mode manuel	83
Objectif d'Essai	83
Exigences et Critères de Réussite	83
Composants Testés.....	83
Méthode d'Essai	83
Conditions de l'Essai	84

Procédure de test	84
Collecte des données.....	84
Validation des résultats	84
Conclusion	84
Tests opérationnels	87
Test d'une séquence de vol complète sans les hélices	87
Objectif d'Essai	87
Exigences et Critères de Réussite.....	87
Composants Testés.....	88
Méthode d'Essai	88
Conditions de l'Essai.....	88
Procédure de test	89
Collecte des données.....	89
Validation des résultats	89
Conclusion	89
Test d'une séquence de vol complète avec les hélices semi-automatique	92
Objectif d'Essai	92
Exigences et Critères de Réussite.....	92
Composants Testés.....	93
Méthode d'Essai	93
Conditions de l'Essai.....	93
Procédure de test	94
Collecte des données.....	94
Validation des résultats	94
Conclusion	94
Mode d'emploi	96
Mentions dangereuses	96
Procédure à respecter pour le vol manuel.....	96
Procédure à respecter pour le vol automatique	97
Procédure à respecter en cas de problème	98
Durant le vol automatique	98
Durant le vol manuel	98
Entretien et stockage	99
Assistance et support technique	99
Disposition au sol des balises	99



Conclusion	100
Code ESP32-WROOM-32	101
Code ESP UWB DW1000	119
Listes des figures	123

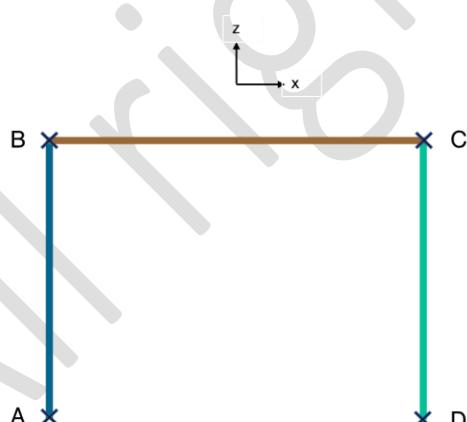
Introduction

Le projet AutoQuad, confié à notre équipe de six ingénieurs par Dragonfly 4U, s'inscrit dans une démarche de développement et de validation d'un système complexe. L'objectif était de concevoir un démonstrateur physique de drone quadricoptère intégrant deux modes de pilotage : manuel et automatique. Ce projet représentait un défi d'ingénierie à haute valeur ajoutée, nécessitant une maîtrise des processus de conception, d'intégration et de validation expérimentale.

Le cahier des charges précisait les exigences suivantes :

1. Modes de fonctionnement :
 - a. En mode manuel, le drone devait être facilement pilotable.
 - b. En mode automatique, il devait exécuter une séquence prédéfinie (décollage, stabilisation, déplacement horizontal, et atterrissage) sans intervention humaine, avec la possibilité de basculer instantanément vers le mode manuel en cas d'urgence.
2. Sécurité et signalisation :
 - a. La commutation entre les modes devait être immédiate.
 - b. Un signal lumineux visible à 20 mètres devait indiquer l'activation du mode automatique.
3. Conception technique :
 - a. Le drone devait s'inscrire dans un gabarit cubique de 50 cm de côté.
 - b. Les composants de base incluaient une plateforme commerciale capable de voler en mode manuel, enrichie de capteurs et calculateurs pour le mode automatique.
 - c. L'armement des moteurs ne devait être possible qu'en mode manuel.
4. Performances attendues :
 - a. Décollage progressif, stabilité au vent, et atterrissage maîtrisé.
 - b. Une évaluation quantitative des capacités de poussée, d'autonomie, et de réponse aux perturbations.

Le mode automatique présent sur le drone quadricoptère doit permettre d'accomplir le parcours illustré sur la figure 1.



Ce parcours sera composé de 4 points A, B, C et D repartis de manière quadratique dans l'espace avec une distance verticale AB et CD fixe de 1,50m +/- 50cm et une distance horizontale BC variable entre 0 et 15m selon l'envie et le besoin de l'opérateur. De plus, un couloir de vol de +/- 30cm à respecter est imposé autour du centre de gravité du sous-système volant.

Le système devra donc être capable de franchir ce parcours dans les conditions explicité précédemment en tenant compte également d'une vitesse maximale autorisée de 0,2m/s et d'une stabilisation aux points B et C de 10s.

Figure 1 : Schéma du parcours devant être réalisé par le drone en mode automatique



Pour atteindre ces objectifs, le développement s'appuie sur des plateformes existantes, sélectionnées comme base pour la conception du démonstrateur. Ces plateformes, bien que limitées par des informations incomplètes sur la structure définitive du produit, permettent de juger de la faisabilité et de la crédibilité du concept vis-à-vis d'une future production en série.

Ce rapport détaille l'ensemble des étapes de développement, depuis l'analyse fonctionnelle initiale jusqu'à la démonstration du démonstrateur assemblé. Il retrace les décisions d'ingénierie critiques, les méthodologies appliquées, ainsi que les résultats des phases de test, dans une optique de traçabilité et d'optimisation des performances.

L'approche adoptée a été structurée autour des principes d'ingénierie système : analyse des besoins, découpage fonctionnel, justification des choix techniques, intégration des sous-systèmes et validation finale par expérimentation. Ce document illustre notre capacité à mener un projet à obligation de résultats, dans le respect des contraintes opérationnelles et des attentes du client.

L'équipe chargée de ce projet est composée de Nicolas ABAT, Alexandre CASTEX, Thomas FORTUNATO, Constance NGUYEN, Julien TAKTALIAN, et Joachim KOEHL en tant que chef de projet, avec le soutien de Marc BEAUCHET et Thomas LISEMBARD en tant qu'autorités techniques.

Conception fonctionnelle

L'analyse opérationnelle vise à répondre à la question fondamentale « Pour quoi ? » c'est-à-dire « Quel est le service rendu ? ». Elle repose sur une série d'éléments détaillant les objectifs, les transitions, les scénarios et les exigences associés à son fonctionnement.

DCU

Le DCU représente les différentes interactions entre le système et son environnement. Il identifie les acteurs internes et externes qui influencent ou dépendent du fonctionnement du système en fonction des différents environnement et/ou phase d'utilisation. Cela permet de visualiser les échanges d'informations et les flux d'énergie, en mettant en évidence les besoins spécifiques de chaque acteur. On identifie pour notre système deux phases avec chacune leurs contraintes spécifiques : la phase d'utilisation et la phase de charge. Deux diagrammes sont donc à représenter illustrés en figures 2 et 3.

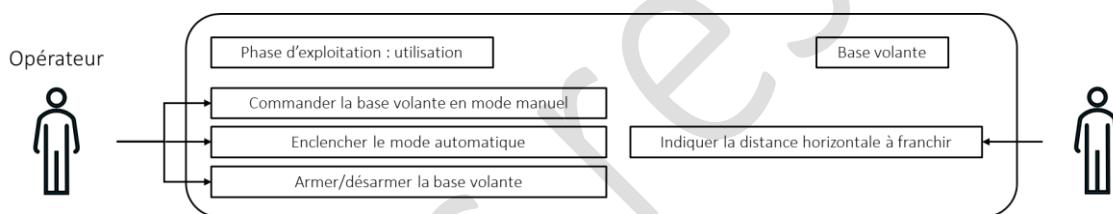


Figure 2 : DCU en phase d'utilisation

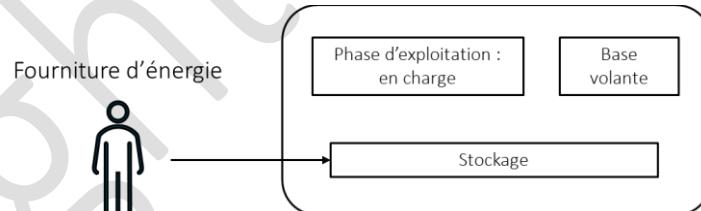


Figure 3 : DCU en phase de charge

DCS

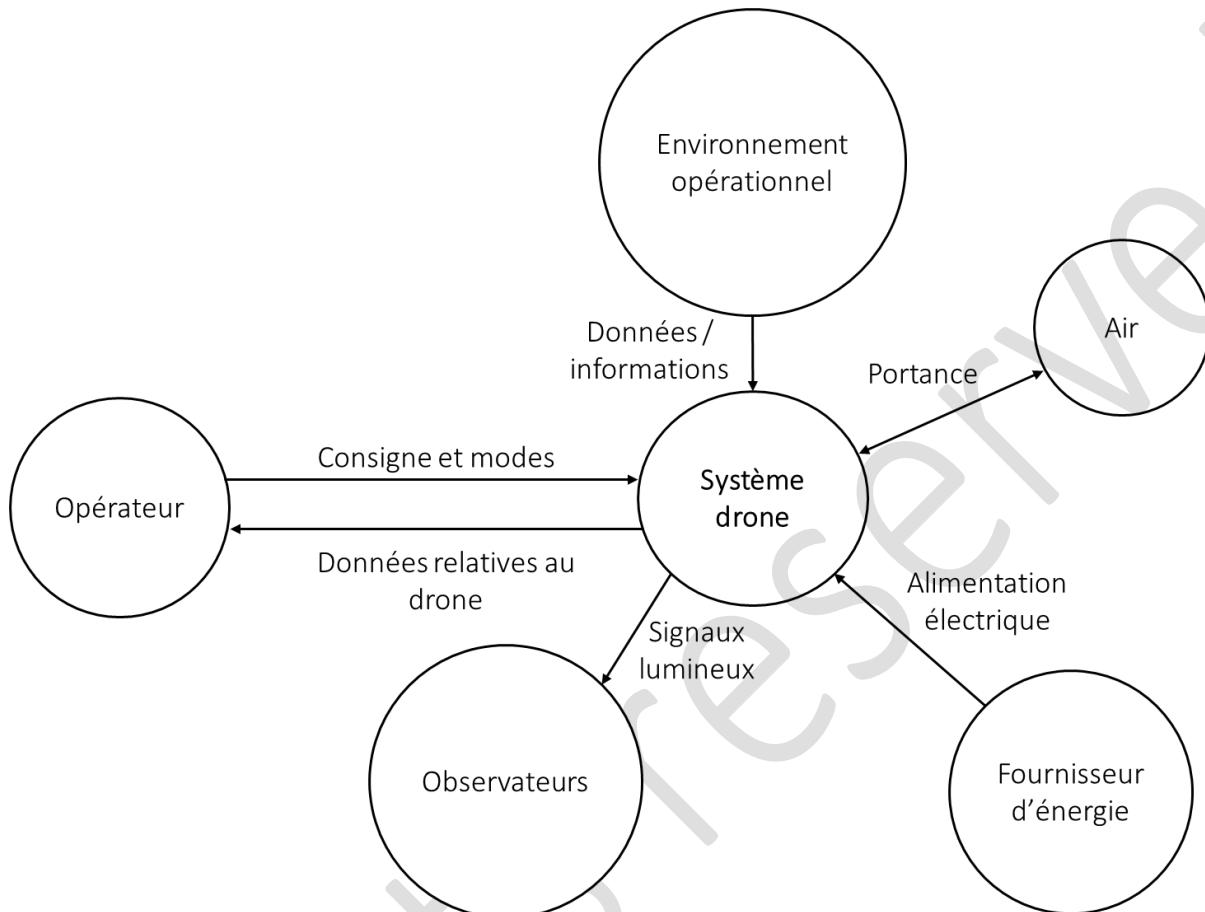


Figure 4 : DCS

Le DCS illustré par la figure 4 représente les différentes interactions du système avec l'environnement extérieur et précise si l'action est subie ou imposé par notre système drone. On peut ainsi définir 5 interactions extérieures à notre système, que sont l'environnement opérationnel, l'air, l'opérateur, le(s) observateur(s) ainsi que le l'énergie électrique d'alimentation.

Diagramme des transitions

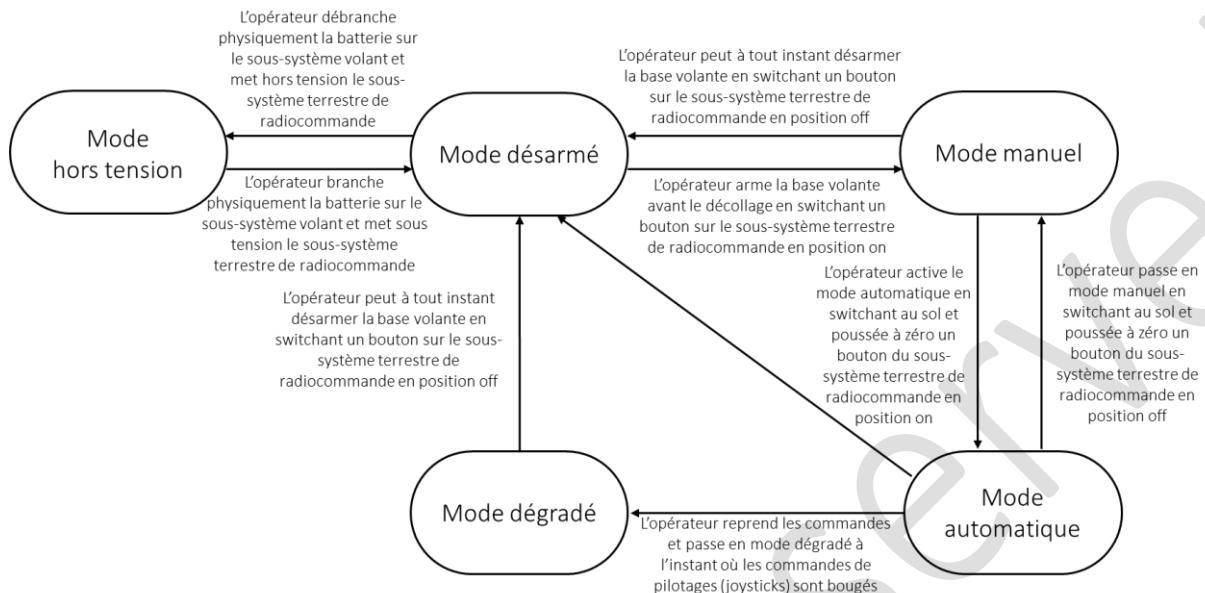


Figure 5 : Diagramme des transitions

Le diagramme des transitions illustré à la figure 5 décrit les différents états possibles du système et les conditions qui entraînent le passage d'un état à un autre. Ces états incluent, par exemple, l'armement, la phase de vol automatique, la stabilisation, ou le désarmement. Chaque transition est associée à un événement déclencheur ou une condition particulière, comme une commande de l'opérateur ou une détection de problème.

Scénarios opérationnels

Les scénarios opérationnels décrivent les séquences d'événements attendus lors de l'utilisation du système dans des conditions réelles. Ils permettent de simuler le comportement du système et de valider sa capacité à répondre aux besoins opérationnels. On peut ainsi construire les scénarios représentés sur les figures 6 et 7 où le premier représente un cycle de déroulement classique sans interruption ni problème rencontré tandis que le deuxième représente un cycle dans lequel le mode dégradé a dû être déclenché par l'opérateur à la suite d'un problème.

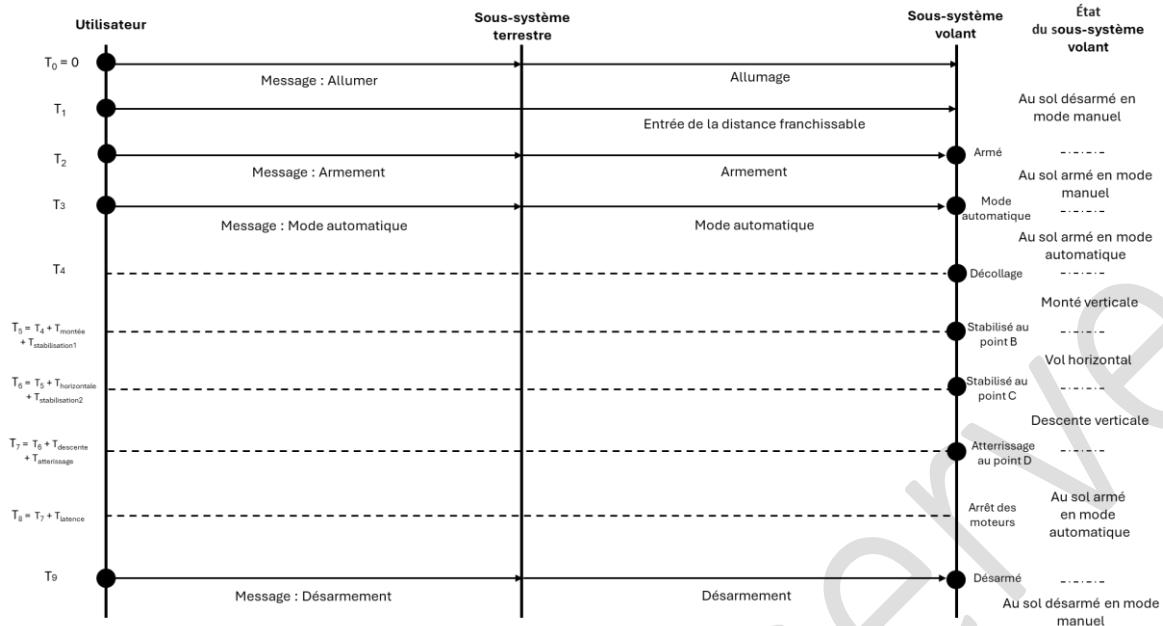


Figure 6 : Scénario opérationnel en situation normal de fonctionnement

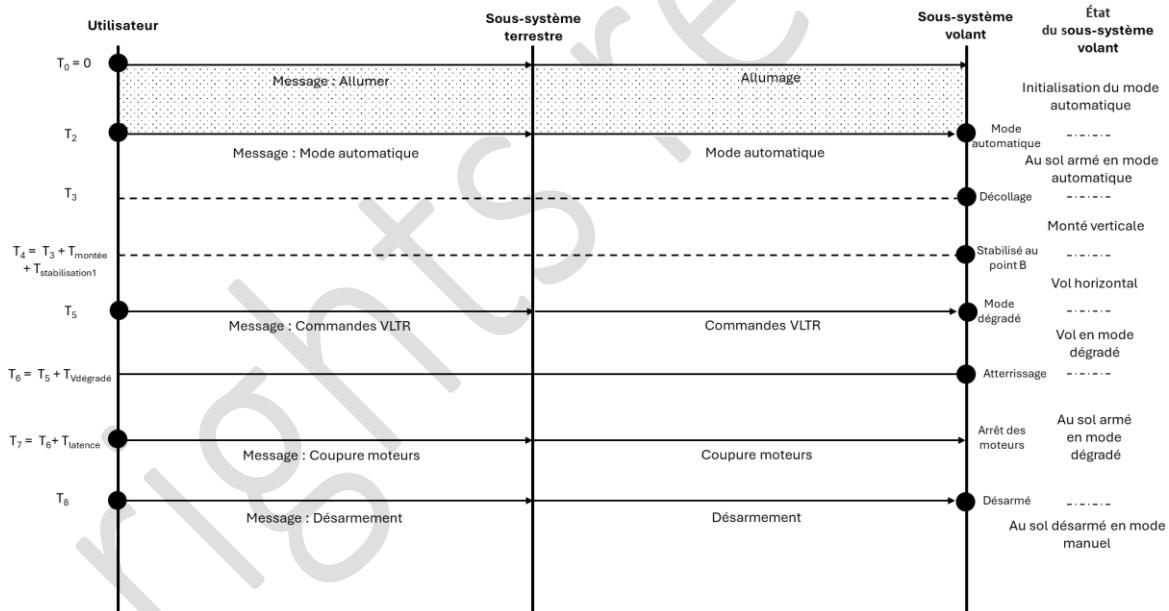


Figure 7 : Scénario opérationnel en cas de problème détecté durant le vol

Exigences opérationnelles fonctionnelles

Le tableau des exigences opérationnelles regroupe de manière synthétique les besoins à satisfaire pour garantir le bon fonctionnement du système à l'échelle macroscopique. Chaque exigence est définie en termes de service attendu, de critère de validation et, dans certains cas, d'indicateurs de performance. Ces exigences sont réparties en deux tableaux distincts. Le tableau des exigences opérationnelles fonctionnelles décrit les services spécifiques que le système doit fournir. Il permet de s'assurer que les fonctions principales répondent aux besoins identifiés. En parallèle, le tableau des exigences opérationnelles non-fonctionnelles spécifie les contraintes globales du système, comme les performances, la robustesse ou les normes de sécurité, garantissant ainsi une conformité avec les standards et les attentes du client.

Tableau 1 : Exigences opérationnelles fonctionnelles

Numéros	Exigences fonctionnelles	Sous-exigences	Critères
EOF1	Pilotage manuel	Le système doit être pilotable par un opérateur débutant	L'opérateur doit pouvoir n'avoir aucune connaissance du système et n'avoir jamais piloté de drone jusqu'à présent
		Le système de pilotage doit être précis	Les commandes doivent offrir une réponse linéaire à l'opérateur
			La résolution des joysticks doit être suffisante pour permettre une finesse de pilotage
			Le système doit rester dans un cylindre de rayon 25cm afin de respecter les exigences de vol
		Le système doit être stable	Le système assure des accélérations et décélérations progressives
			Le système n'effectue aucun mouvement brusque et n'a pas d'instabilité stationnaire
			Le système évolue avec des vitesses permettant de le suivre et de le contrôler
EOF2	Pilotage automatique	Le système doit pouvoir suivre une séquence de vol autonome préprogrammée	Doit pouvoir suivre un trajet type rectangulaire passant par des points de stabilisation précis A (décollage au sol), B (stabilisation en l'air avant avancement), C (stabilisation en l'air après avancement) et D (atterrissement au sol)

			<p>La séquence est définie par une distance franchissable à définir par l'opérateur avant le vol</p>
			<p>Le décollage et l'atterrissement doivent être effectués de manière douce et progressive</p>
		<p>Le système de pilotage doit être précis</p>	<p>Les commandes doivent offrir une réponse linéaire</p>
			<p>Le système doit rester dans un cylindre de rayon 25cm afin de respecter les exigences de vol</p>
		<p>Le système doit être stable</p>	<p>Le système assure des accélérations et décélérations progressives, avec un seuil fixé à $0,5\text{m/s}^2$</p>
			<p>Le système évolue avec des vitesses permettant de le suivre et de le contrôler</p>
			<p>Le système n'effectue aucun mouvement brusque et n'a pas d'instabilité stationnaire</p>
		<p>Le système doit pouvoir interpréter les signaux du mode automatique comme si ceux-ci venaient de la radiocommande</p>	<p>Les signaux transmis par le pilote automatique doivent avoir la même forme que ceux transmis par la radiocommande</p>
			<p>Le signal lumineux doit être clairement identifiable</p>
		<p>Un signal lumineux visible doit indiquer que le mode automatique est actif</p>	<p>Le signal lumineux doit être visible en continu</p>
			<p>Le signal lumineux doit être visible sur une portée au moins égale à 20m</p>
EOF3	Commutation	<p>La commutation du mode automatique vers le mode dégradé doit être possible instantanément</p>	<p>Reprise des commandes par l'opérateur dès que les joysticks de la radiocommande sont bougés</p>

		Le système n'autorise pas de retour en mode automatique une fois le mode dégradé activé	Impossibilité de réactiver le pilote automatique sans désarmement et réarmement au sol
		Le système doit pouvoir passer du mode automatique au mode manuel et inversement	Le système doit pouvoir en permanence et sur commande de l'opérateur pouvoir passer en mode manuel ou en mode automatique au sol avec une poussée nulle
EOF4	Résistance face à un vent relatif	Le système doit pouvoir rester stable	Le système assure des accélérations et décélérations progressives
			Le système évolue avec des vitesses permettant de le suivre et de le contrôler
			Le système n'effectue aucun mouvement brusque, n'a pas d'instabilité stationnaire et maintient une stabilité dynamique
		Le système doit pouvoir rester précis	Doit pouvoir rester dans un cylindre de rayon 25cm
		Le système doit pouvoir rester pilotable	Les commandes doivent rester instinctive, c'est-à-dire linéaire et sans à-coups
EOF5	Alimentation	Le système doit pouvoir se recharger	Doit être compatible avec l'alimentation électrique disponible sur secteur
		Le système doit pouvoir stocker l'énergie nécessaire au vol	Doit contenir une énergie suffisante pour effectuer 6 vols consécutifs d'une longueur maximale de 15m de long
EOF6	Pilotage dégradé	Le système doit être stable	Le système assure des accélérations et décélérations progressives
			Le système n'effectue aucun mouvement brusque et n'a pas d'instabilité stationnaire
			Le système évolue avec des vitesses permettant de le suivre et de le contrôler
		Le système doit être pilotable	Le système doit être en mesure de se reposer en toute sécurité grâce au contrôle de l'opérateur
		Le système doit être incapable de repasser en mode automatique	Le système doit être incapable de revenir en mode automatique tant qu'il n'a pas été totalement désarmé et mis hors tension

Tableau 2 : Exigences opérationnelles non-fonctionnelles

Numéros	Exigences non fonctionnelles	Critères
EONF1	Robustesse	Le système doit pouvoir être résistant face aux divers chocs et atterrissage brutaux
		Le système doit pouvoir gérer les erreurs et les changements de configuration (robustesse logicielle)
EONF2	Maintenance	Les composants internes doivent être facilement accessibles et remplaçables
EONF3	Démonstrateur	Le système doit être un démonstrateur et non un produit commercialisable
EONF4	Fiabilité	Le système doit avoir un fort taux de disponibilité (proche de 100%) tel que MBTF = 200 jours
EONF5	Personnalisation	Le système doit pouvoir être personnaliser selon les goûts esthétiques du client
EONF6	Consommation	Le système doit limiter et optimiser sa consommation énergétique afin de prolonger la durée des vols et la durée de vie du stockage
EONF7	Implémentation	Le système doit être compatible avec la base volante

Logigrammes

La conception fonctionnelle du système vise à répondre à la question « Comment le système fonctionne-t-il ? ».

Cette question peut notamment être répondu grâce à la mise en forme de chaque processus clé sous forme de logigramme. Ceux-ci permettent de visualiser les différentes étapes, les conditions logiques et les actions exécutées pour assurer le fonctionnement fluide et sécurisé de notre système. On peut alors formaliser tout le processus sous un même logigramme général, recensant l'ensemble des actions globales de notre système :

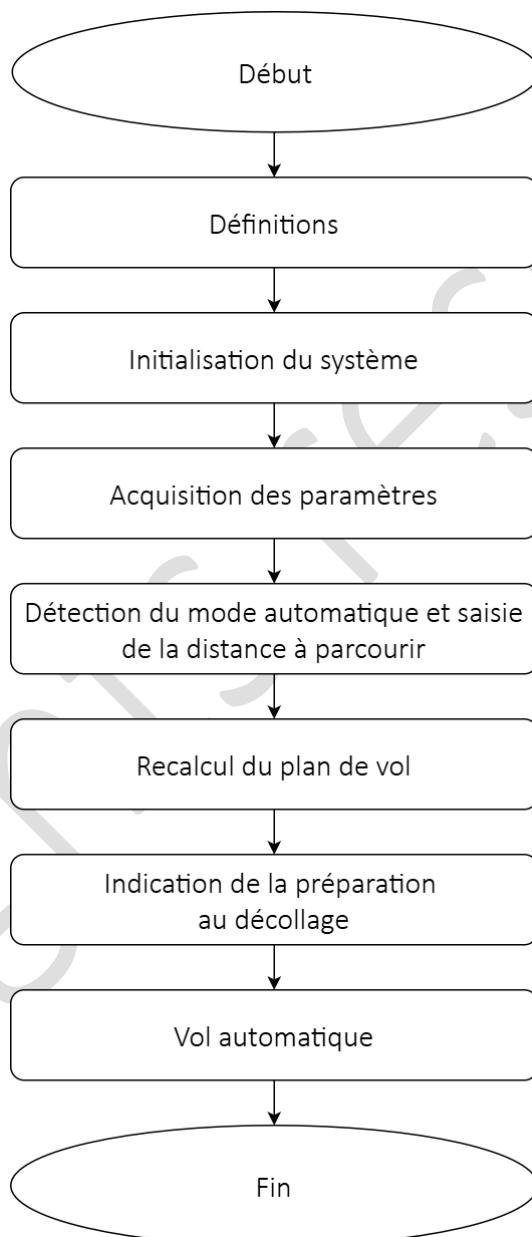


Figure 8 : Logigramme de fonctionnement général

Tout commence par l'initialisation du système, où les paramètres nécessaires sont définis et les différentes variables attribuées. Les données par la suite reçues via les capteurs sont ensuite traitées.

La détection du mode automatique est la composante permettant de basculer vers une phase de saisie interactive où l'utilisateur indique la distance horizontale que le système devra parcourir. Cette saisie est validée en vérifiant les entrées et en s'assurant que les données sont conformes. Ensuite, le système procède au recalcul du plan de vol, une étape cruciale où les coordonnées des points de passage sont déterminées en fonction des consignes. Si les coordonnées sont incohérentes, le processus est ajusté pour éviter toute exécution erronée.

Une fois le plan validé, l'indication de préparation au décollage informe l'utilisateur que le système est prêt à démarrer. Le système entre alors en mode vol automatique, subdivisé en plusieurs, qui suivent un fonctionnement standardisé. Chaque phase commence par le calcul des consignes pour ajuster la position verticale et horizontale, en tenant compte des erreurs actuelles. Ces erreurs sont utilisées pour générer des commandes, envoyées aux moteurs pour ajuster la trajectoire du drone.

Il est à noter qu'entre les phases de déplacement (« PHASE 1 », « PHASE 2 » et « PHASE 3 »), des phases de stabilisation (« STABILIZE AT B » et « STABILIZE AT C ») garantissent que le drone reste stable à des points intermédiaires clés. Ces stabilisations utilisent des algorithmes spécifiques pour corriger les dérives en temps réel, en ajustant les angles de contrôle en tangage et roulis ainsi que la poussée pendant un laps de temps donné.

Enfin, chaque étape inclut une vérification continue des conditions de sécurité, notamment via une fonction d'arrêt d'urgence, permettant de vérifier si le mode manuel où a été activé, entraînant le passage du système en mode dégradé. De plus, le désarmement du système lors de n'importe quelle phase, entraîne de manière irréversible l'arrêt total et immédiat de l'ensemble des systèmes de motorisation présents à bord.

Les processus s'enchaînent jusqu'à ce que la mission soit terminée. Les logigrammes suivants détaillent ces étapes de manière précise depuis les phases d'interaction avec l'utilisateur, comme la saisie des paramètres, jusqu'aux calculs dynamiques et la génération des commandes moteurs afin d'assurer une représentation claire des différents composants du système et de leur coordination pour atteindre les objectifs du vol.

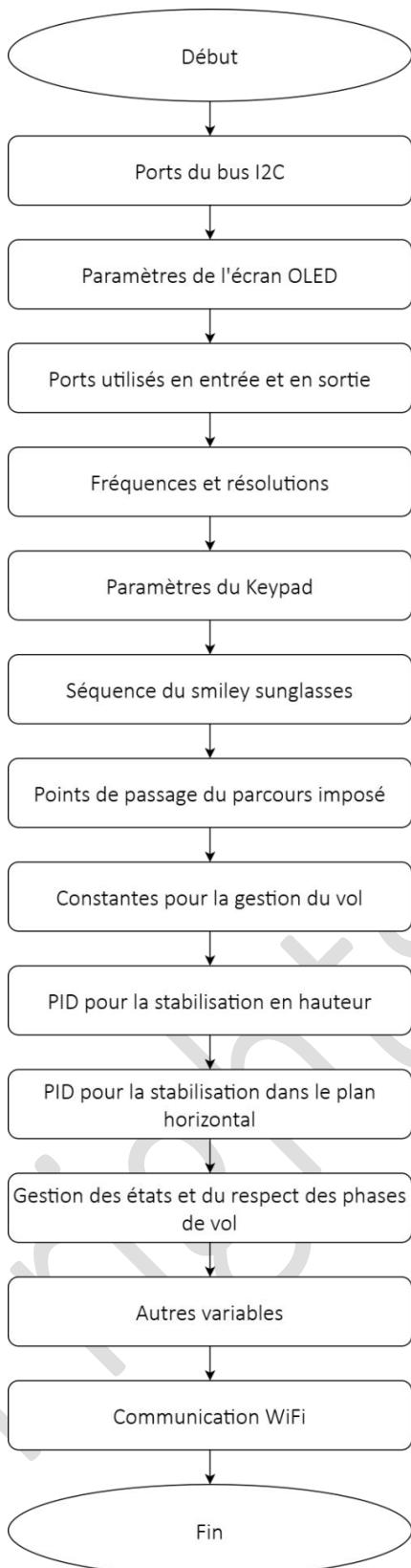


Figure 9 : Logigramme des définitions

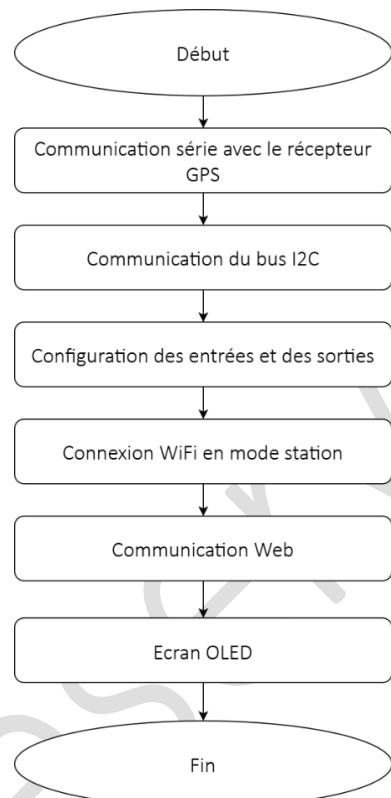


Figure 10 : Logigramme de l'initialisation du système

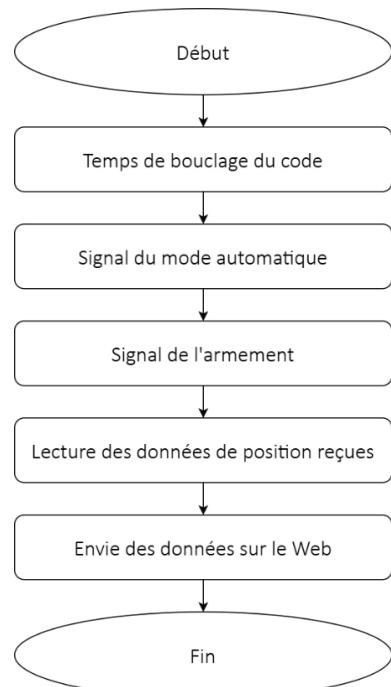


Figure 11 : Logigramme de l'acquisition des paramètres

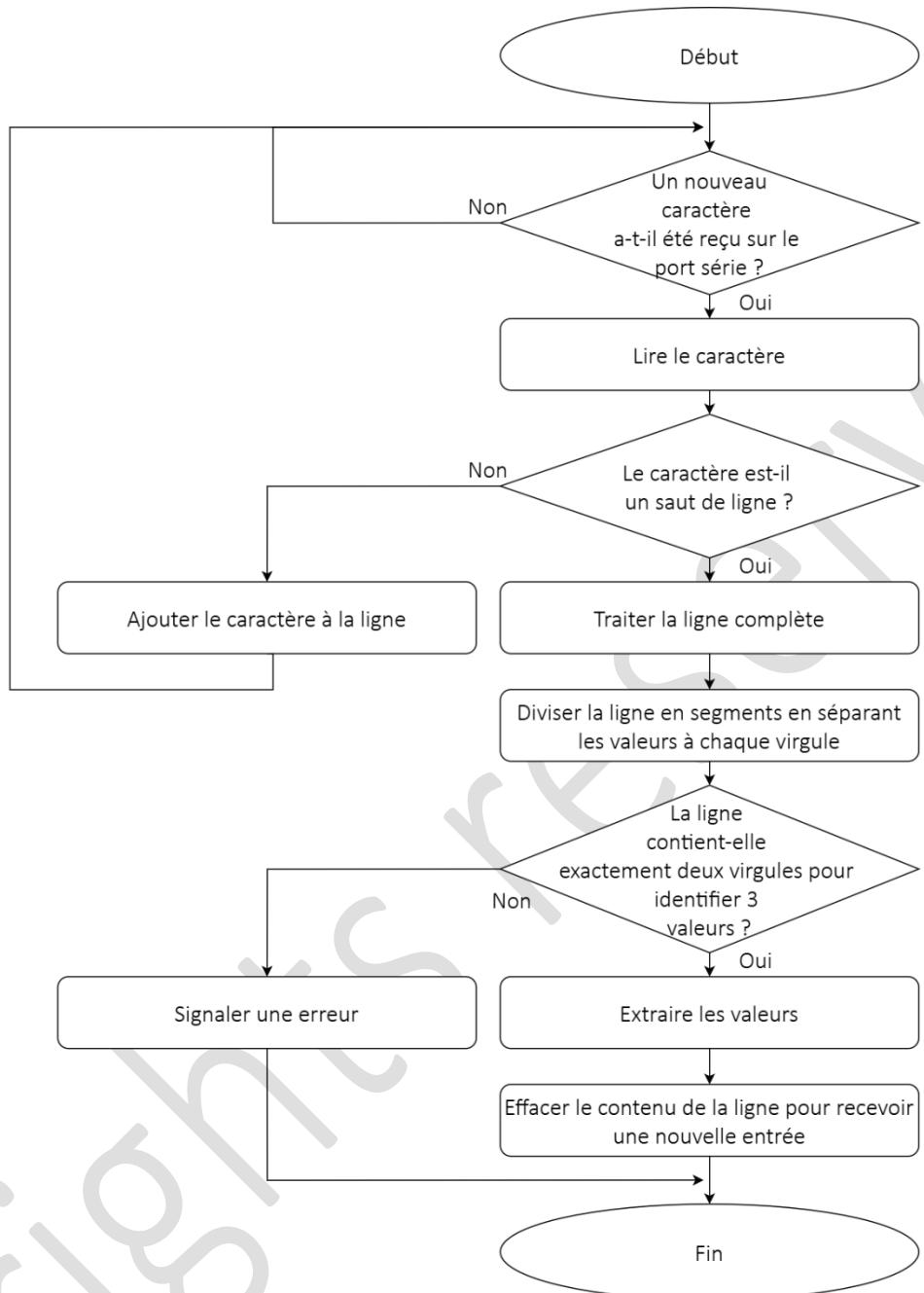


Figure 12 : Logigramme de la lecture des données de position reçues

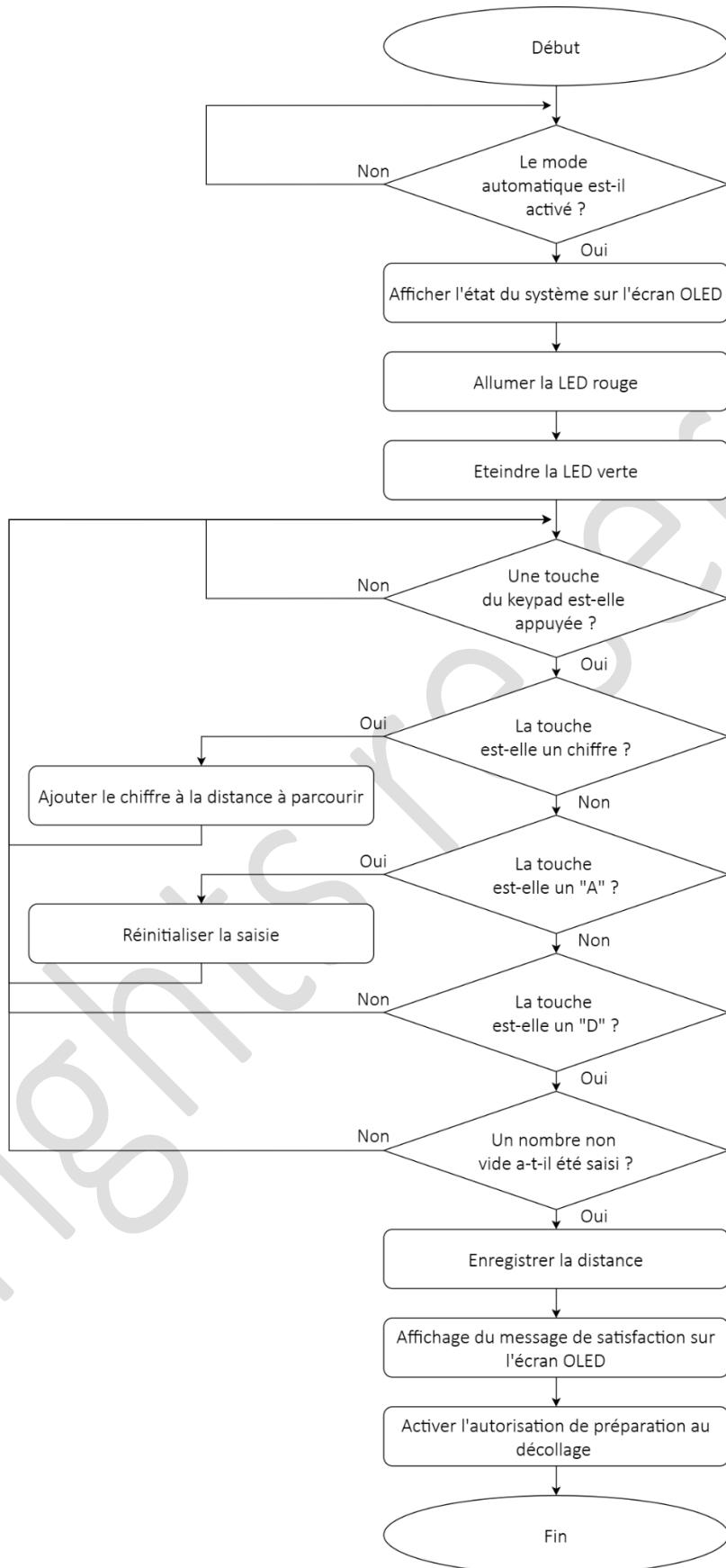


Figure 13 : Logigramme de la détection du mode automatique et de la saisie de la distance à parcourir

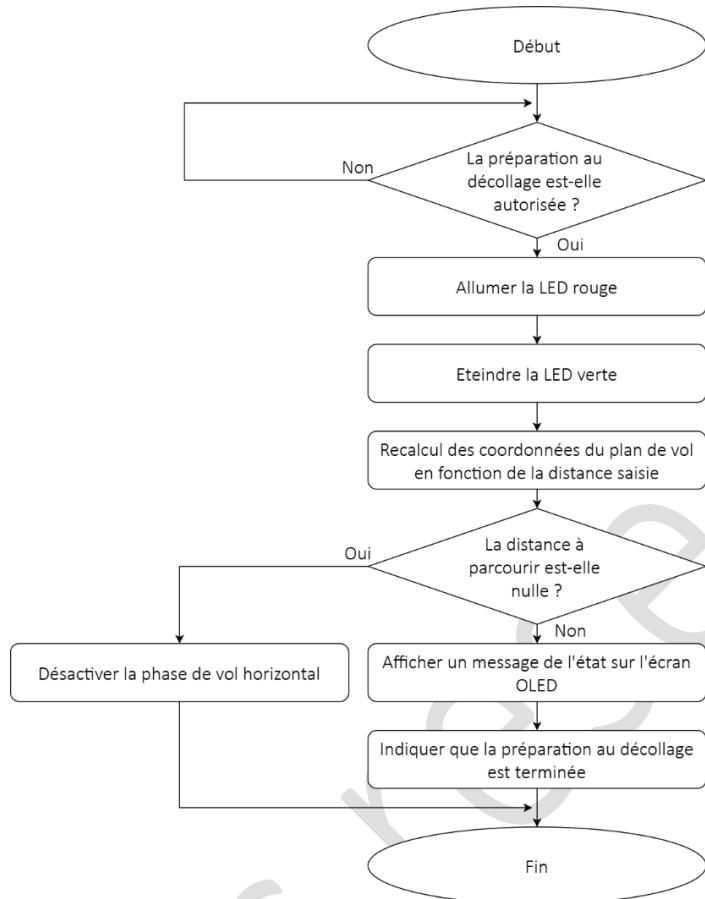


Figure 14 : Logigramme du recalcul du plan de vol

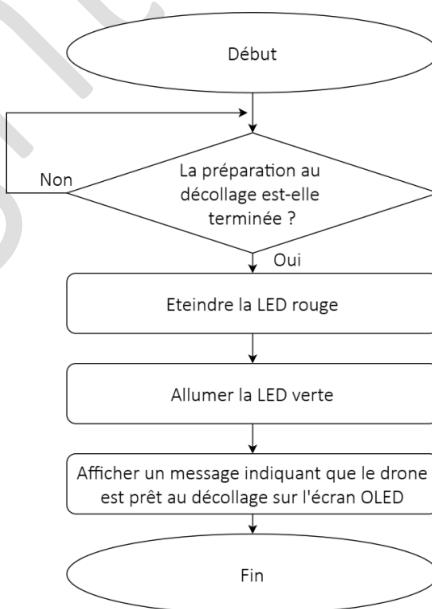


Figure 15 : Logigramme de l'indication de la préparation au décollage

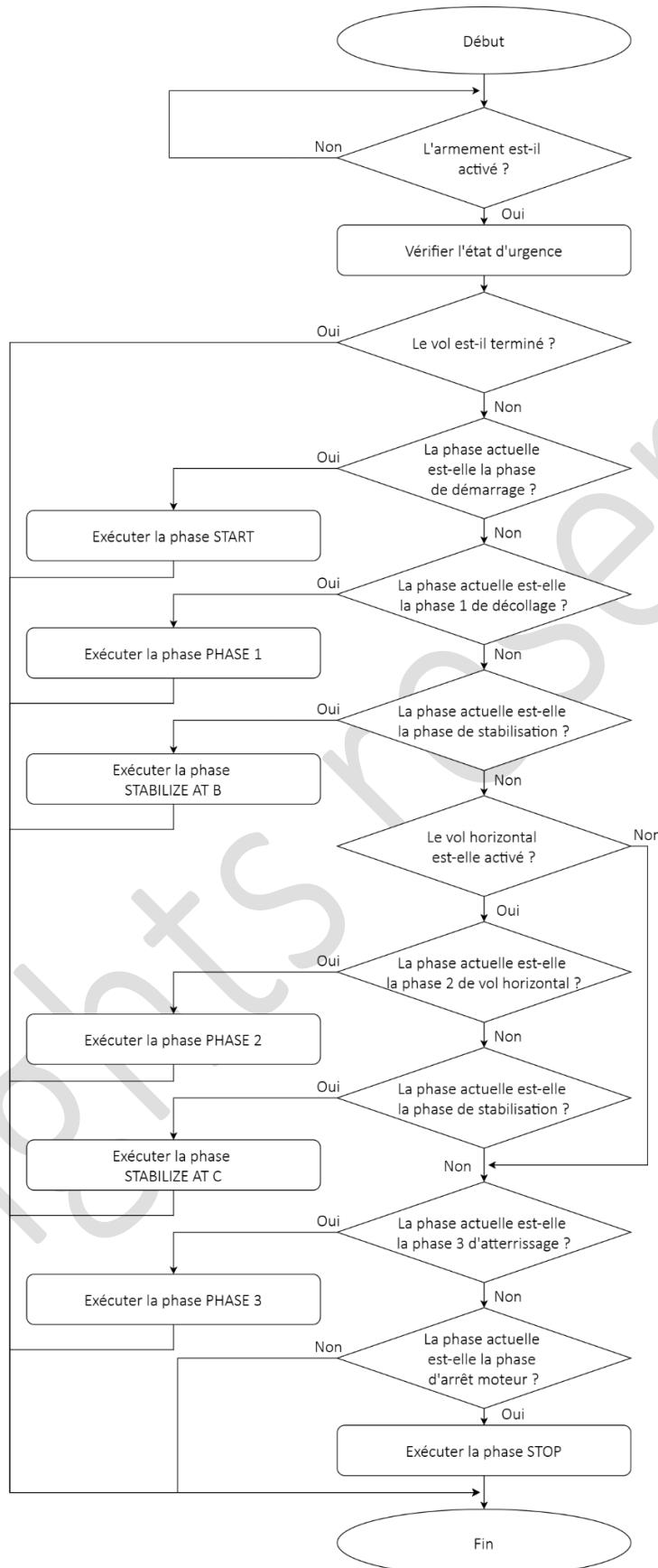


Figure 16 : Logigramme de la phase de vol automatique

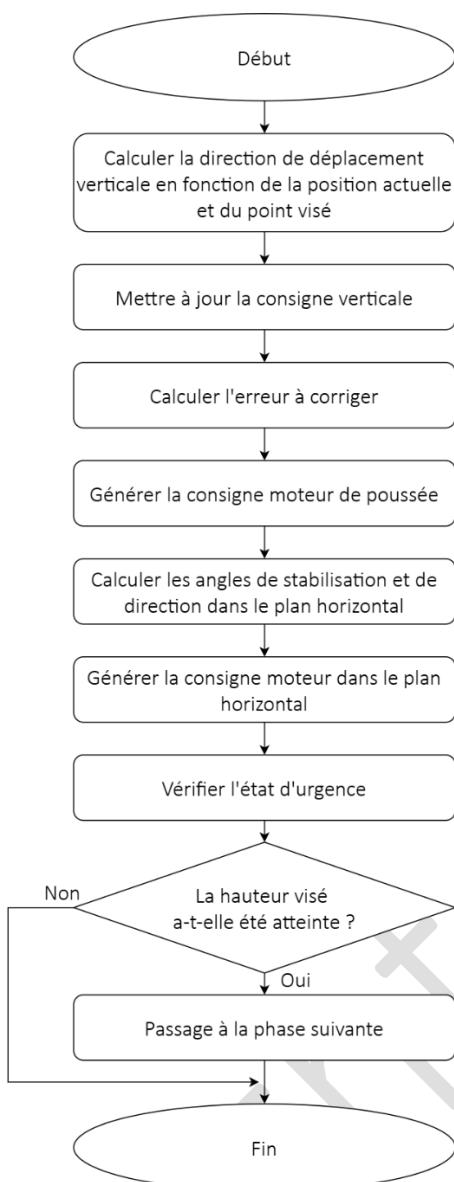


Figure 17 : Logigramme des phases de déplacement en vol

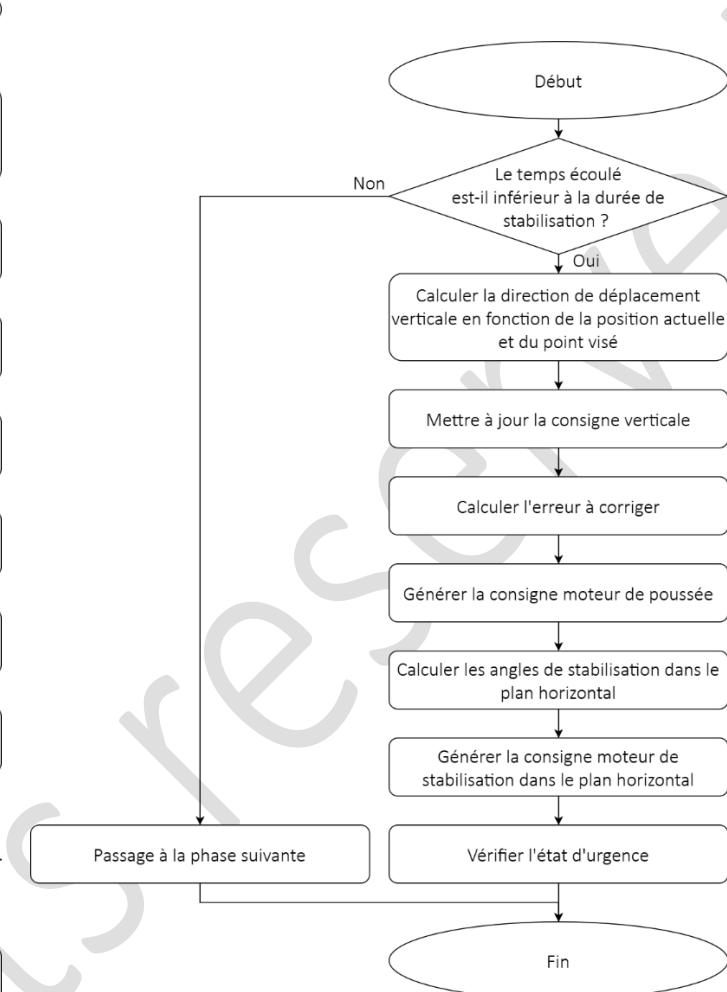


Figure 18 : Logigramme des phases de stabilisation en vol

Les logigrammes précédemment présentés détaillent le fonctionnement du système, en exposant les différentes étapes, conditions, et actions nécessaires à l'atteinte des objectifs fixés. Néanmoins, un point spécifique reste toutefois à préciser : comme indiqué sur la figure 17, le passage d'une phase à une autre intervient lorsque la hauteur cible est atteinte. Cependant, afin de prendre en compte les erreurs d'acquisition, des sphères de rayon 15 cm sont définis autour des points visés de telle sorte à ce que si le système volant se trouve à l'intérieur de ces sphères, la phase en cours est validée et le système passe à l'étape suivante. Ce principe est illustré dans les figures ci-dessous, qui montrent également le couloir de vol à respecter ainsi que la trajectoire prévue à suivre.

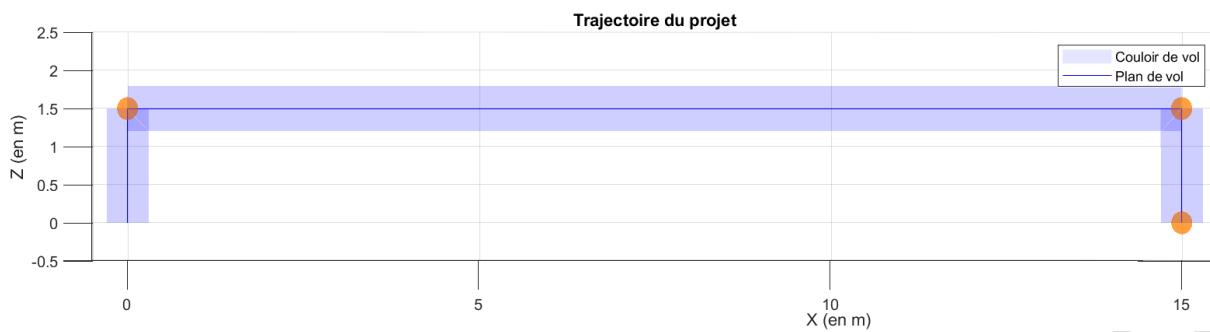


Figure 19 : Vue latérale du plan de vol

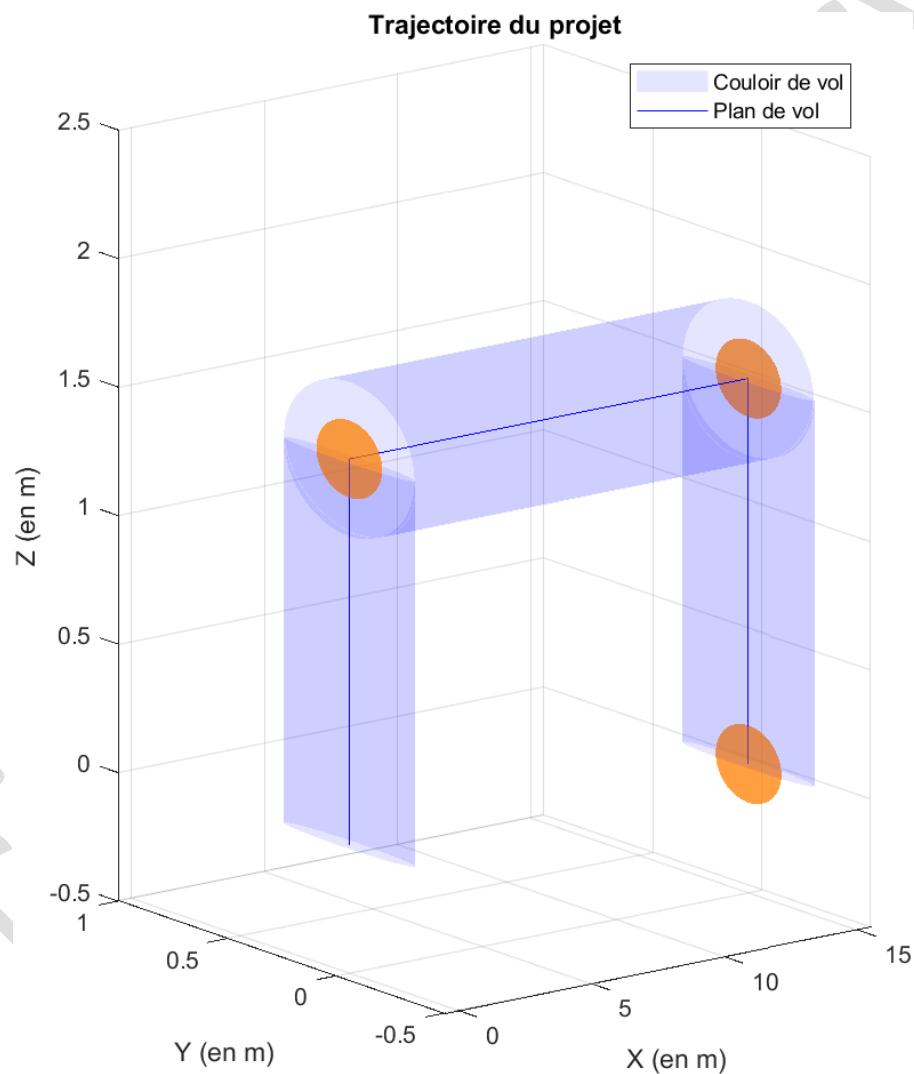


Figure 20 : Vue 3D du plan de vol



Conception technique

Modèle 3D

Pour obtenir des données précises sur les caractéristiques du drone, nous avons procédé à une modélisation complète à l'aide du logiciel CATIA V5. Cette modélisation permet de visualiser et de mesurer les paramètres critiques, tout en simplifiant les calculs par l'adoption d'un ensemble d'hypothèses.

Hypothèses de modélisation

Afin de modéliser notre système, nous mettons en place les hypothèses suivantes :

- La masse du drone, en l'absence de la batterie, est considérée comme parfaitement symétrique
- La batterie, en tant que composant le plus lourd, est le seul élément modifiant significativement l'inertie du drone
- Le centre de gravité est supposé situé au point de symétrie du drone

Description du modèle 3D

La figure 1 illustre le modèle 3D, incluant :

- Le châssis du drone et les hélices
- Deux blocs gris en haut dont le bloc central représente le contrôleur de vol et le bloc sur le côté représente le récepteur radio utilisé
- Un bloc central gris représentant la batterie encastrée à l'intérieur du châssis entre les deux plaques supports noires
- La masse des fils, modélisée sur chaque bras et convergeant vers un point central

A titre de comparaison, la figure 22 montre une image réelle du drone utilisé comme base pour la modélisation.

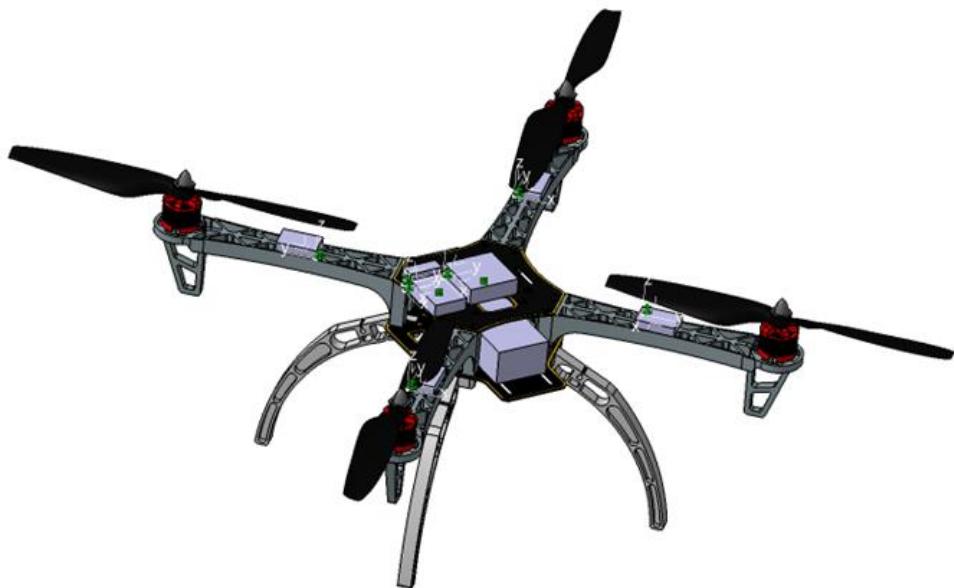


Figure 21 : Modèle 3D initial du système volant



Figure 22 : Modèle réel du système volant initial

Analyse des distances clés

La modélisation 3D permet, entre autres, d'extraire des données essentielles aux calculs théoriques qui seront amenés à être fait par la suite. Ces données sont notamment la distance entre les hélices, soit l'envergure du système volant, et le centre de gravité.

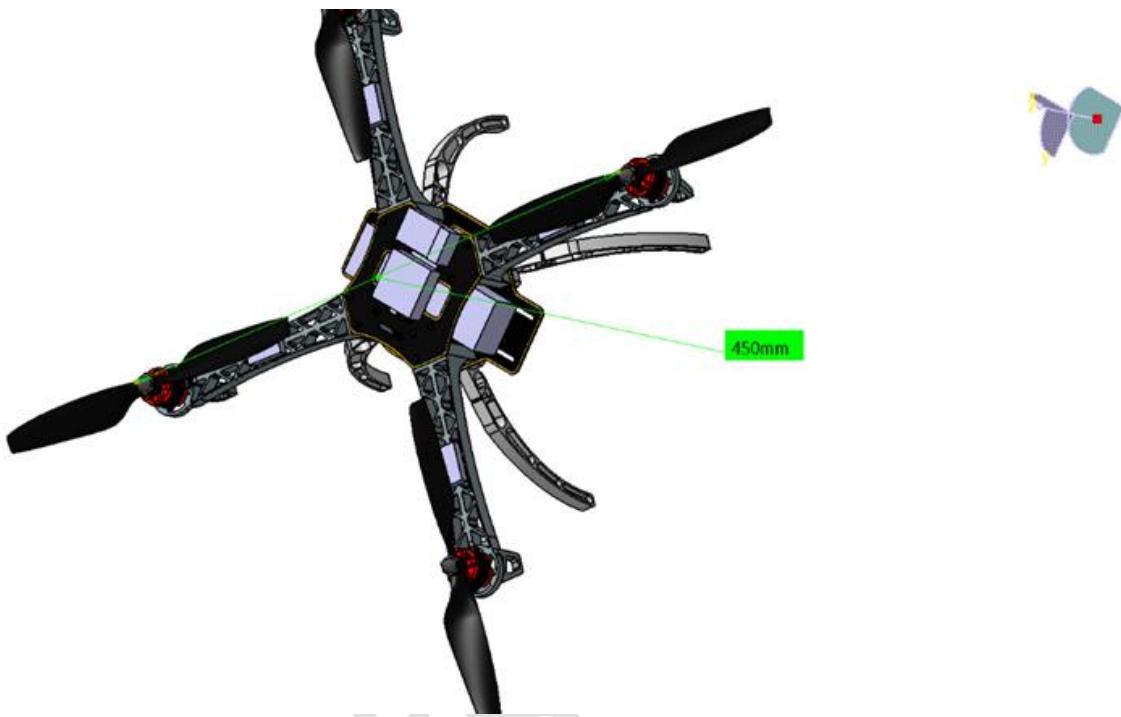


Figure 23 : Envergure du système

On mesure ainsi une distance totale entre deux hélices opposées de 450 mm. Etant donné les hypothèses posées précédemment et notamment par symétrie du modèle, nous pouvons alors en déduire un centre de gravité situé à 22,5 cm en partant du centre d'une hélice et en suivant la droite formée par le centre de celle-ci et le centre de l'hélice opposée.

Détermination des matrices d'inertie

À partir de la masse totale du drone, mesurée expérimentalement à **1,232 kg**, il est alors possible d'en extraire sous le logiciel Catia V5 les matrices d'inertie en tenant compte de la symétrie de la masse.



Figure 24 : Pesée du système volant réel

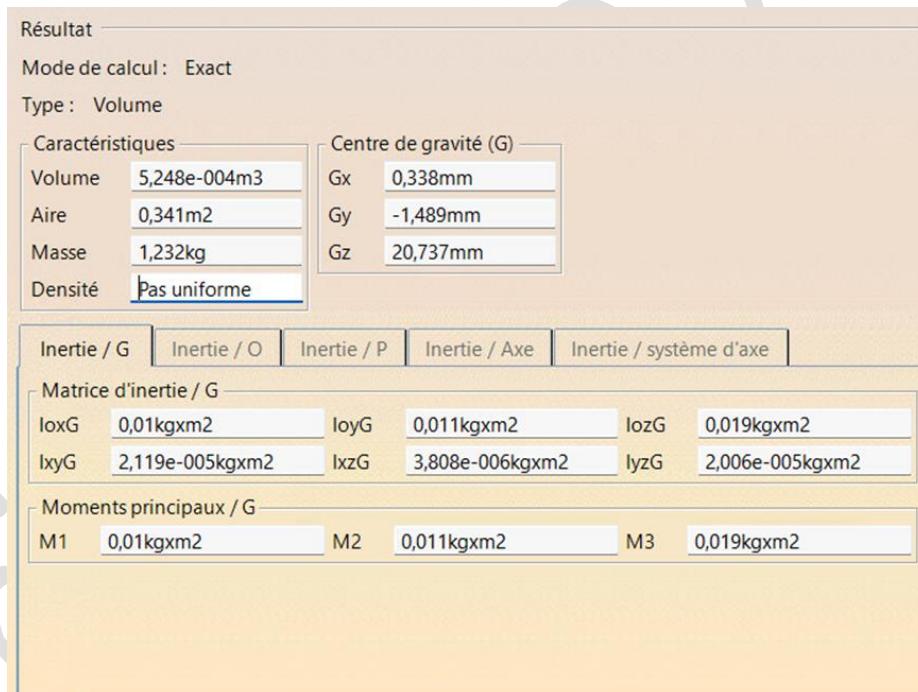


Figure 25 : Matrices d'inertie déterminées numériquement grâce au modèle 3D

On obtient ainsi les valeurs suivantes où toutes les unités sont en kg.m² :

I _{xx}	I _{yy}	I _{zz}	I _{xy}	I _{xz}	I _{yz}
0,01	0,011	0,019	2,119.10 ⁻⁵	3,808.10 ⁻⁶	2,006.10 ⁻⁵

De plus, étant donné la symétrie du système, nous considérerons que I_{xy}=I_{yx}, I_{xz}=I_{zx} et I_{yz}=I_{zy}.

Détermination des surfaces projetées

Les surfaces projetées du système volant ont également été modélisées afin de pouvoir calculer les coefficients de traînée aérodynamique en combinaison avec les caractéristiques des hélices et des moteurs.

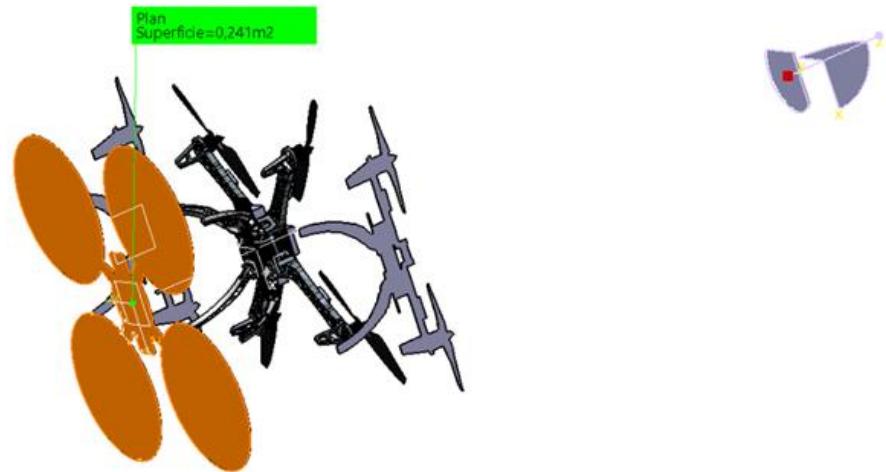


Figure 26 : Surfaces projetées vue du dessous

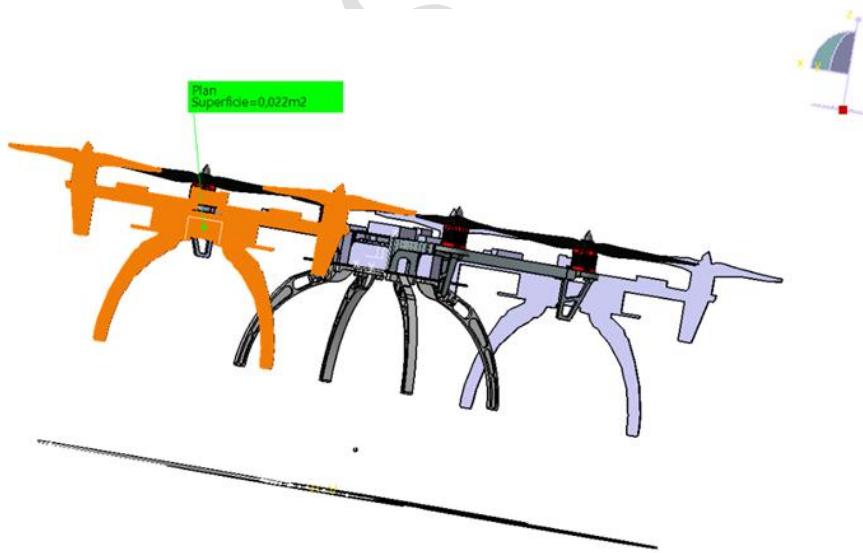


Figure 27 : Valeur des surfaces projetées vue de côté

On est alors en mesure d'obtenir les résultats suivants :

- Sur la figure 26 : La surface inclut les hélices modélisées en rotation (cercles) et correspond à $0,241 \text{ m}^2$
- Sur la figure 27 : La surface correspondante est estimée à $0,022 \text{ m}^2$

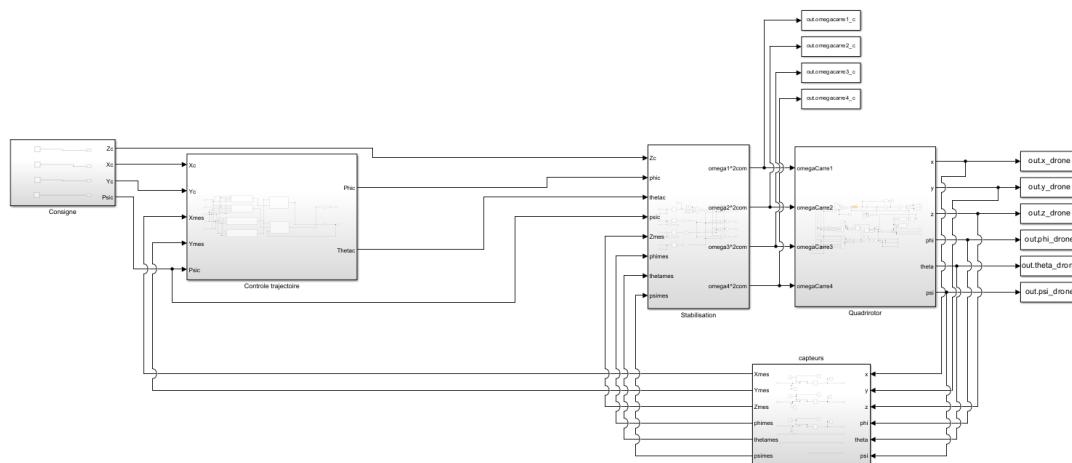
Simulations numériques sous le logiciel Simulink

Dans le cadre de notre projet, nous avons développé un modèle sous Simulink afin de simuler les limites de notre système. L'objectif principal de cette simulation était de nous permettre d'analyser les différentes options technologiques de capteurs, notamment ceux mesurant la position, la vitesse ou l'accélération, et de déterminer lequel serait le plus adapté pour l'asservissement en trajectoire de notre drone quadrirotor. Cette démarche impliquait de modéliser avec précision le comportement du drone, afin que les résultats obtenus soient fiables et exploitables.

Structure du modèle

La première étape pour concevoir notre simulation sous Simulink a consisté à créer un modèle représentatif du drone quadrirotor, que nous avons qualifié de « jumeau numérique ».

Ce modèle devait reproduire de manière réaliste les dynamiques du drone afin d'anticiper les performances du système, même en présence de capteurs imparfaits ou de bruit dans les mesures. Cette modélisation rigoureuse était essentielle pour garantir que les simulations effectuées dans Simulink resteraient pertinentes lors des tests en conditions réelles.



Pour construire ce modèle, nous avons commencé par intégrer les équations décrivant les mouvements dynamiques du drone en partant d'un modèle de drone quadrirotor déjà préconçu. Ces équations traduisent les interactions complexes entre les forces générées par les moteurs, les moments de rotation, et les forces externes comme la gravité ou les frottements. Ces éléments ont été directement tirés des données fournies par notre unité technique, ce qui nous a permis de démarrer sur des bases solides et adaptées au drone que nous utilisons.

Détermination des constantes physiques

Les constantes physiques utilisées dans les calculs ont été déterminées à partir du modèle 3D du drone réalisé sous Catia, ainsi que de mesures directes effectuées sur le drone réel. Le modèle Catia nous a permis d'extraire des informations essentielles, notamment la matrice d'inertie III, exprimée en kg.m², qui caractérise la résistance du drone à la rotation autour de ses axes principaux. La matrice obtenue est donnée par :

$$I = \begin{pmatrix} 0,01 & 2,119 * 10^{-5} & 3,808 * 10^{-6} \\ 2,119 * 10^{-5} & 0,011 & 2,006 * 10^{-5} \\ 3,808 * 10^{-6} & 2,006 * 10^{-5} & 0,019 \end{pmatrix}$$

Cette matrice joue un rôle clé dans la modélisation des mouvements du drone, en particulier lors des rotations autour des axes x, y et z.

La matrice de trainée A, qui décrit les forces et moments aérodynamiques en fonction de la vitesse du drone et été définie arbitrairement de la manière suivante :

$$A = \begin{pmatrix} 0,0001 & 0 & 0 \\ 0 & 0,0001 & 0 \\ 0 & 0 & 0,0001 \end{pmatrix}$$

En parallèle, on définit également la masse totale du drone telle que m=1,232kg ainsi que la distance entre les rotors et le centre de gravité telle que l=0,225mm.

Ensuite, la constante de poussée k a été calculée à partir de la condition de vol stationnaire du drone. En effet, en vol stationnaire, la poussée totale générée par les hélices doit équilibrer le poids du drone. La poussée totale étant répartie de manière égale sur les quatre hélices, la poussée par hélice est donc donnée par :

$$T = \frac{m * g}{4} = 3,02N$$

En considérant la masse totale comment étant de 1,232kg et la constante de pesanteur g telle qu'égale à 9,81m.s⁻². En supposant que la poussée soit proportionnelle au carré de la vitesse de rotation des hélices et en supposant que celle-ci soit de 10464RPM d'après les données constructeur, la constante de poussée k est obtenue par :

$$k = \frac{T}{\omega^2} = 2,51 * 10^{-6} N.s^2$$

La constante de traînée b a été estimée en utilisant la relation suivante :

$$b = \frac{C_{total}}{\omega^2} = \frac{P_{total}}{\omega^3} = 3,55 * 10^{-7} N.s^2/m$$

En considérant ici à nouveau une vitesse de rotation de moteur de 10464RPM, soit 1096rad/s. De plus, P_{total} désigne ici la puissance nécessaire pour maintenir le drone en vol stationnaire, ici de 170,4W.

Le moment d'inertie gyroscopique J_r est calculé à partir de l'inertie du rotor des moteurs et de la vitesse de rotation des hélices. En supposant que le rotor peut être approximé par un cylindre mince de masse 17g et de diamètre 27,7mm, le moment d'inertie du rotor peut être déterminé comme suite :

$$I_M = \frac{1}{2} * m * \left(\frac{d}{2}\right)^2 = 1,63 * 10^{-6} kg.m^2$$

Le moment d'inertie gyroscopique est alors obtenu en multipliant l'inertie du rotor par la vitesse de rotation, soit :

$$J_r = I_{rotor} * \omega = 1,78 * 10^{-3} kg.m^2/s$$

L'ensemble de ces constantes, obtenues à partir du modèle Catia et des mesures expérimentales, a permis de paramétrier précisément le modèle Simulink du drone. Ces valeurs garantissent une représentation fidèle de la dynamique du système, rendant les simulations fiables et exploitables pour le développement et l'optimisation des algorithmes de contrôle et d'asservissement.

Mise en place du modèle complet

Dans un premier temps, les capteurs ont été considérés comme parfaits, c'est-à-dire sans bruit ni biais, afin de valider le comportement fondamental du modèle. Une fois cette étape initiale réalisée, des imperfections réalistes, telles que le bruit et les biais, ont été progressivement intégrées pour évaluer leur impact sur les performances globales du système.

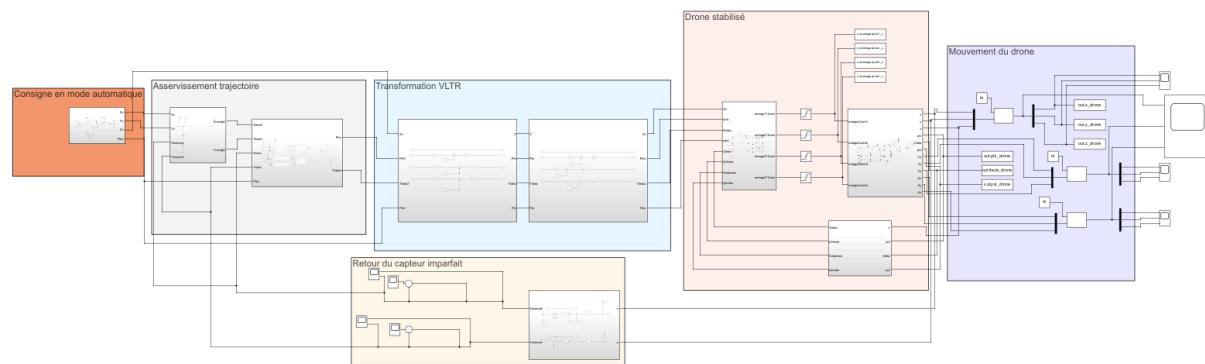


Figure 28 : Modèle Simulink complet

Une fois les équations fondamentales intégrées, nous avons paramétré le modèle en spécifiant les caractéristiques physiques du drone. Cela comprenait notamment la masse, les dimensions, et les propriétés d'inertie, qui influencent la manière dont le drone réagit aux commandes. Par ailleurs, nous avons modélisé les actionneurs, c'est-à-dire les moteurs, avec leurs réponses dynamiques spécifiques, en incluant leurs limitations, comme les temps de montée et les saturations.

Pour rendre notre modèle plus réel, nous avons fait aussi simuler la transformation des signaux envoyées par le sous-système terrestre. En effet, les consignes envoyées par le sous-système terrestre sont transmises sous forme de signaux PPM (Pulse Position Modulation) dans une plage comprise entre 1000 et 2000 µs. Ces signaux doivent être convertis pour correspondre aux variables physiques contrôlant le drone. Chaque commande correspond à une grandeur physique : la vitesse ascensionnelle (V_{VLTR}), la vitesse de rotation autour de l'axe de lacet (ψ_{VLTR}), l'angle de tangage (θ_{VLTR}) et l'angle de roulis (ϕ_{VLTR}). Les équations qui régissent l'ensemble des angles de consigne et la hauteur de consigne sont les suivantes :

$$Z_c = \int \left(\frac{V_{z_{VLTR}}}{\Omega_{maxmoteur} - \Omega_{V_z=0moteur}} \right) * \left(\frac{\Omega_{maxmoteur}}{2000 - 1000} \right) * (V_{z_{VLTR}} - 2000)$$

$$\psi_c = \int \left(\frac{2 * \psi_{VLTR}}{2000 - 1000} \right) * (-\psi_{VLTR} - 1000)$$

$$\theta_c = \left(\frac{2 * \theta_{VLTR}}{2000 - 1000} \right) * (-\theta_{VLTR} - 1000)$$

$$\phi_c = \left(\frac{2 * \phi_{VLTR}}{2000 - 1000} \right) * (-\phi_{VLTR} - 1000)$$

La conversion des signaux repose sur des équations spécifiques, comme illustré ci-dessus. Ces équations adaptent les signaux bruts reçus en unités physiques utilisables par le drone. Par exemple, la vitesse ascensionnelle (Z_c) est calculée en tenant compte des caractéristiques dynamiques du moteur ($\Omega_{maxmoteur}$) et $\Omega_{(Vz = 0moteur)}$), ce qui permet de traduire la commande en une consigne adaptée. De manière similaire, les commandes pour la rotation de lacet (ψ_c), le tangage (θ_c) et le roulis (ϕ_c) normalisent les signaux reçus dans la plage [1000 ; 2000] µs et les mettent à l'échelle pour produire des consignes proportionnelles à leur grandeur physique respective.

Ce processus de conversion est essentiel pour assurer la correspondance entre les commandes générées par le sous-système terrestre et les dynamiques physiques du drone. Il garantit ainsi une interprétation correcte des consignes par le système embarqué, permettant un contrôle précis et fiable du drone.

Nous allons voir si notre modèle réagit bien pour des consignes données.

Validation du modèle

Nous avions comme consigne entrée un couloir de vol à respecter et voici les résultats :

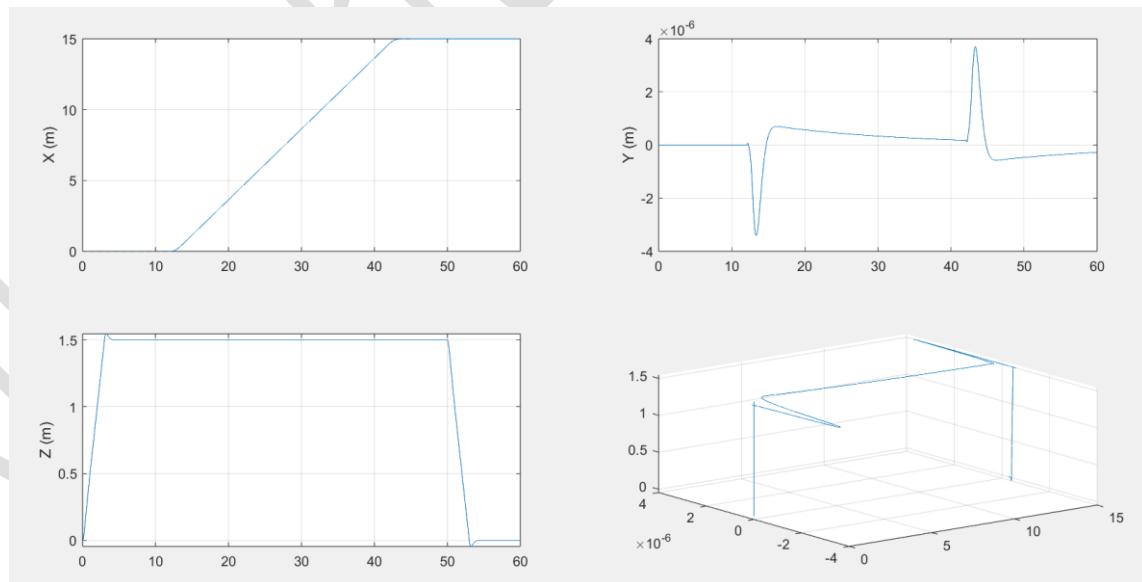


Figure 29 : Résultats du respect du couloir de vol

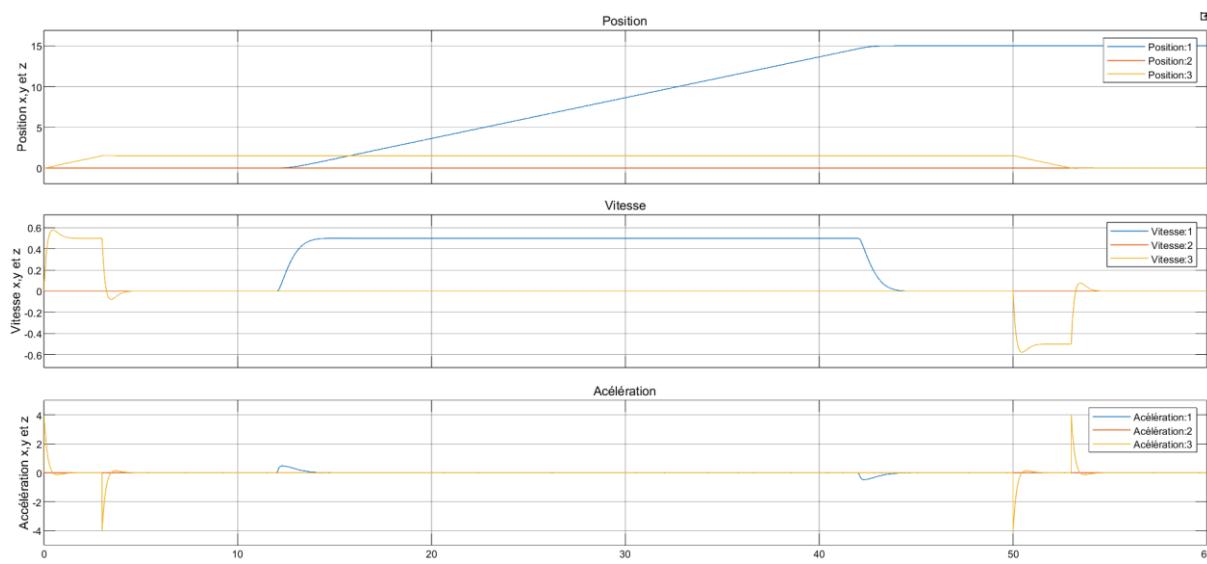


Figure 30 : Résultats télémétriques du respect du couloir de vol

L'évaluation de la pertinence du modèle repose sur une comparaison entre les résultats de la simulation et les comportements attendus du drone en conditions réelles. Les simulations ont montré que le modèle est capable de reproduire fidèlement les dynamiques du drone, y compris les interactions complexes entre les forces et les moments. Par ailleurs, les variations de consigne lors de la simulation et durant toute les phases correspondent aux performances attendues.

Ces figures nous montrent que notre modèle est en cohérence avec le drone réel. Nous pouvons donc l'utiliser pour simuler la boucle de retour pour l'asservissement en trajectoire et dégager une des technologies possibles à adopter.

La simulation avec imperfections

Une fois le modèle du drone finalisé, nous avons utilisé Simulink pour simuler le comportement de l'asservissement en trajectoire avec différents types de capteurs. L'idée était d'observer comment chaque technologie (position, vitesse ou accélération) influençait la capacité du système à suivre une trajectoire donnée.

Nous avons simulé différents types de capteurs (mesure de position, vitesse et accélération). Ces capteurs ont été intégrés au modèle avec des imperfections réalistes, telles que du bruit ou des biais, afin d'évaluer leurs performances et leurs impacts sur l'asservissement du drone. Cette étape était cruciale pour tester des scénarios proches des conditions opérationnelles.

Au cours de ces simulations, nous avons injecté des consignes de trajectoire que le drone devait suivre et avons analysé la manière dont le système réagissait. En utilisant un retour basé sur la mesure de position, nous avons pu observer une bonne précision globale, mais avec une réponse légèrement plus lente et sensible au bruit de mesure. Lorsque nous avons utilisé des capteurs de vitesse, le système a présenté des temps de réponse plus rapides, bien que la précision puisse être compromise en cas de dérive ou de bruit non compensé. Enfin, en basant l'asservissement sur la mesure d'accélération, nous avons constaté une grande réactivité du système, mais au prix d'une instabilité accrue due à l'amplification des incertitudes de mesure.

Choix technologique

Ces simulations ont permis de dégager plusieurs enseignements. D'une part, elles ont mis en évidence les avantages et les inconvénients de chaque type de capteur en fonction des critères de performance, comme le temps de réponse, la stabilité et la précision. D'autre part, elles ont montré l'importance de considérer les limitations technologiques des capteurs, notamment lorsqu'ils sont exposés à des environnements bruyants ou perturbés. Grâce à ces résultats, nous avons pu établir des critères clairs pour orienter notre choix technologique en fonction des besoins spécifiques de l'asservissement en trajectoire.

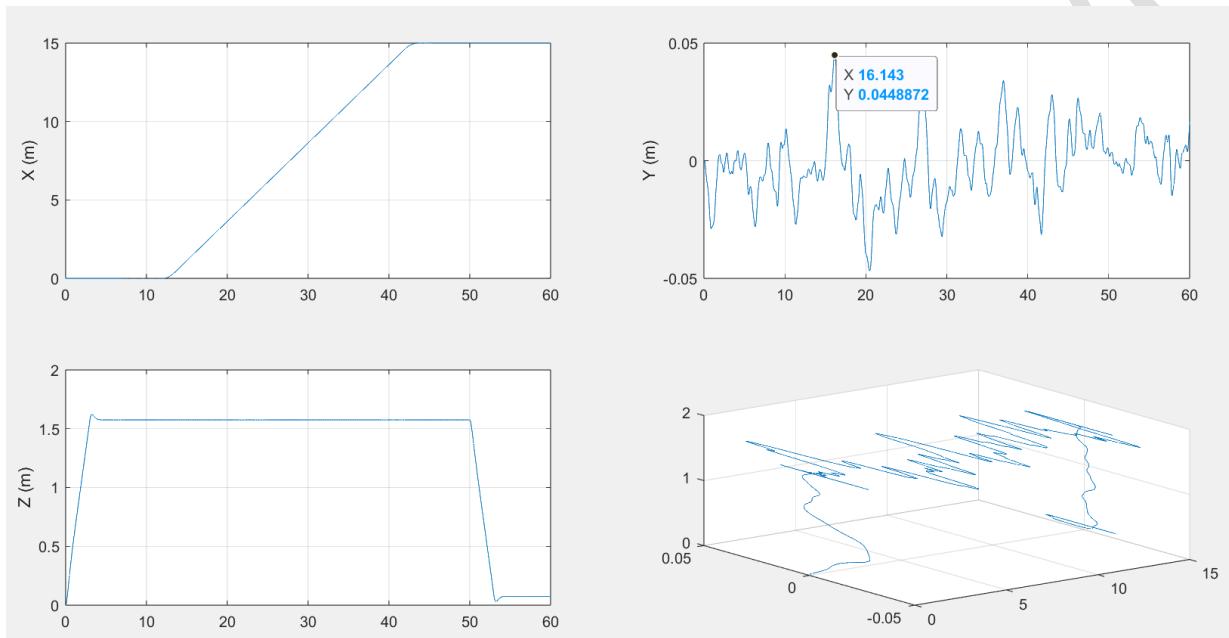


Figure 31 : Résultats du respect du couloir de vol avec bruit

Après avoir analysé les résultats des simulations, nous avons choisi d'opter pour la technologie basée sur la mesure de position comme le montre la figure au-dessus.

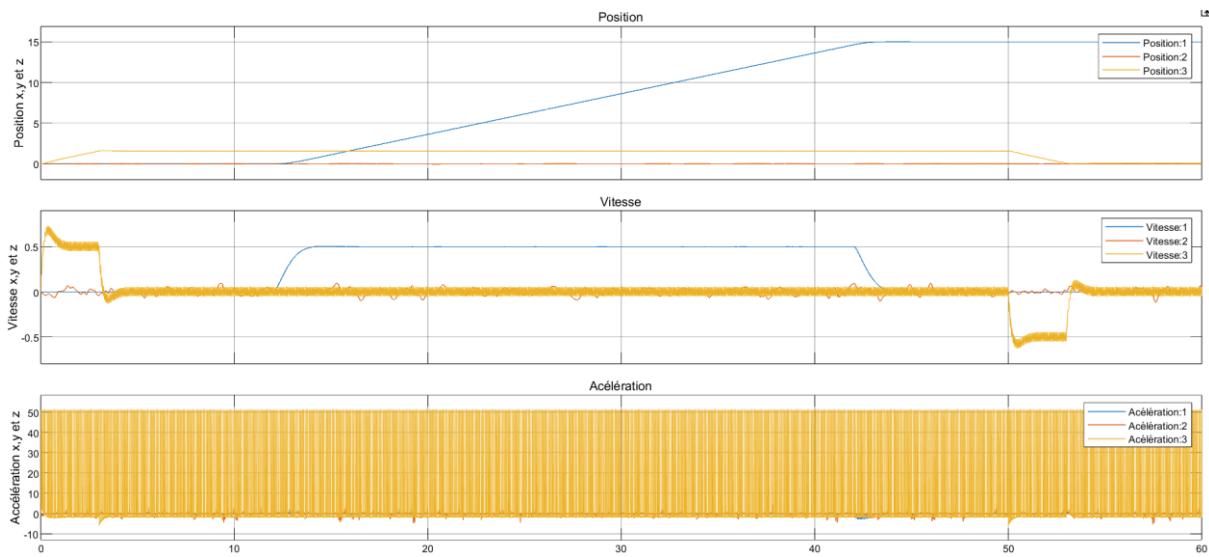
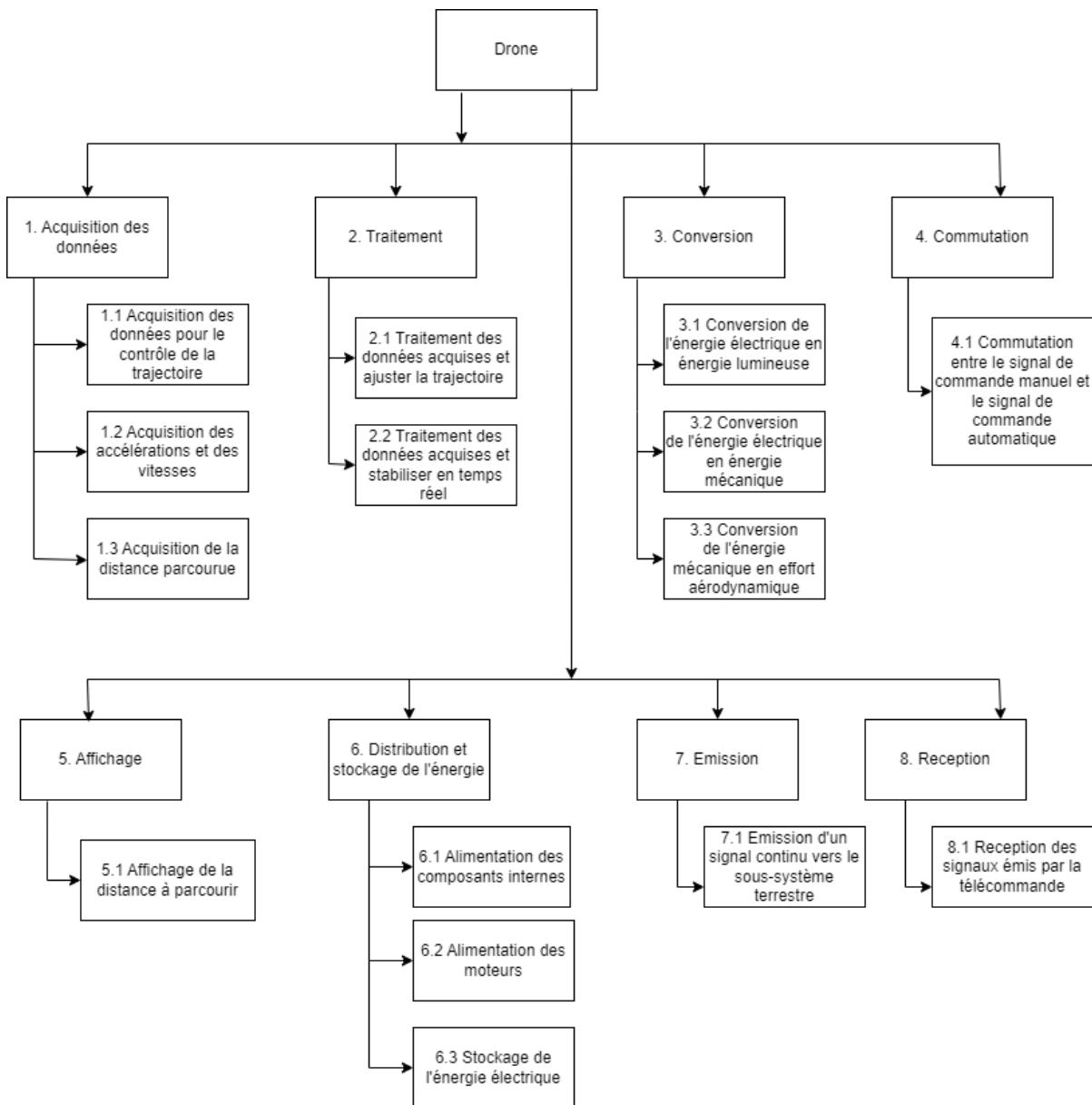


Figure 32 : Résultats télémétriques du respect du couloir de vol avec bruit

Ce choix s'explique par plusieurs avantages significatifs. Tout d'abord, les capteurs de position ont montré une meilleure tolérance au bruit, ce qui est essentiel pour maintenir la stabilité et la précision du système dans des environnements perturbés. De plus, ces capteurs ne présentent pas de dérive, contrairement aux capteurs de vitesse et d'accélération, qui peuvent être affectés par des biais cumulés au fil du temps. Cette caractéristique garantit une précision constante sur le long terme.

Grâce à cette analyse, Simulink nous a permis de valider non seulement la faisabilité de cette technologie, mais également son adéquation avec les exigences de l'asservissement en trajectoire. Ce travail nous offre ainsi une base solide pour le développement futur du système, en assurant des performances optimales et une robustesse accrue face aux contraintes opérationnelles.

Arborescence fonctionnelle



L'arborescence fonctionnelle du drone est organisée autour de huit grandes fonctions interdépendantes, subdivisées en sous-fonctions, afin de garantir un fonctionnement global efficace en mode manuel comme en mode automatique. La première fonction, l'acquisition des données (1), concerne la collecte des informations essentielles au pilotage et à la navigation du drone. Cela inclut l'acquisition des données de trajectoire transmises par la radiocommande et les capteurs de position (1.1), l'enregistrement des accélérations et des vitesses via l'unité de mesure inertie (IMU) pour assurer un contrôle précis (1.2) ainsi que la mesure de la distance parcourue pour évaluer l'autonomie restante (1.3). Une fois ces données collectées, elles sont exploitées en temps réel dans le cadre de la fonction de traitement des données (2), qui englobe l'ajustement continu de la trajectoire (2.1) et la stabilisation du drone face aux perturbations telles que le vent ou les variations brusques de direction (2.2). La conversion d'énergie (3) constitue une fonction clé, car elle permet de transformer l'énergie



électrique en énergie lumineuse pour les dispositifs de signalisation (3.1), en énergie mécanique pour alimenter les moteurs (3.2) et enfin en effort aérodynamique pour produire la portance et la propulsion nécessaires au vol (3.3). La commutation (4) gère le basculement entre les modes manuel et automatique selon les besoins opérationnels, permettant ainsi au drone de s'adapter rapidement aux exigences de l'utilisateur (4.1). L'affichage (5) offre une interface en temps réel à l'opérateur en fournissant des informations cruciales, telles que la distance restante à parcourir jusqu'à l'objectif (5.1), ce qui lui permet de surveiller et de contrôler efficacement le vol. La gestion de l'énergie (6) inclut à la fois l'alimentation des composants internes comme les capteurs et les systèmes électroniques (6.1), l'alimentation des moteurs pour maintenir la propulsion (6.2) et le stockage de l'énergie pour garantir la continuité des opérations, même en cas de forte sollicitation (6.3). Enfin, la communication (7 et 8) assure l'interaction entre le drone et son opérateur : elle repose sur l'émission d'un signal continu contenant les données du drone vers le sous-système terrestre (7.1) ainsi que sur la réception des commandes transmises par la télécommande (8.1), permettant une prise de contrôle efficace et une exécution rapide des instructions.

Boîte grise

La boîte grise constitue une représentation détaillée du système global, illustrant les interactions entre les différents sous-systèmes. Compte tenu de la complexité et de la taille du système, celle-ci est subdivisée en trois parties complémentaires : un schéma général pour une vue d'ensemble, un schéma des sous-systèmes terrestres (balise et radiocommande) décrivant les opérations au sol, et un sous-système volant.

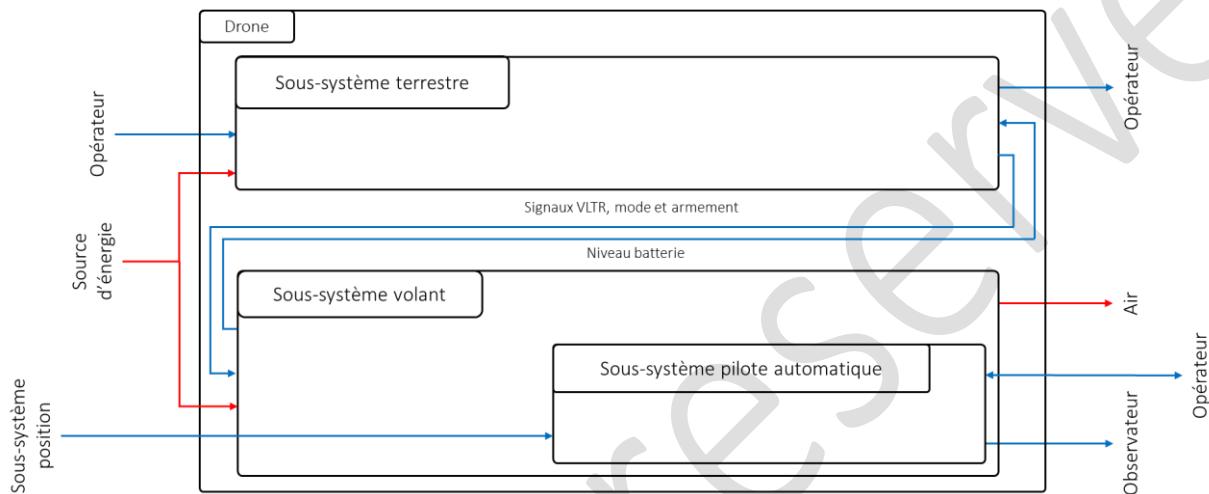


Figure 33 : Boîte grise générale

La figure 33 illustre de manière macroscopique les interactions principales entre les sous-systèmes terrestres et le sous-système volant. Ces deux composantes échangent des informations cruciales pour le fonctionnement du système. Du sous-système terrestre vers le sous-système volant, sont transmis les signaux de commande VLTR, le mode de pilotage (manuel ou automatique) ainsi que le signal d'armement. En retour, le sous-système volant informe le sous-système terrestre sur le paramètre de niveau de batterie.

L'opérateur joue un rôle central en définissant les consignes à suivre via les interfaces du sous-système terrestre. En réponse, ce dernier transmet des données à l'opérateur pour fournir des indications sur l'état du sous-système volant, assurant ainsi une supervision continue. Parallèlement, le sous-système de pilotage automatique, intégré au sous-système volant, collecte en temps réel les données issues du sous-système de positionnement/balise. Ce pilote automatique transmet à la fois des informations visuelles sur le fonctionnement à destination des observateurs à proximité afin d'avertir de son fonctionnement, et des directives précises sur les procédures à suivre à l'intention de l'opérateur. Ce dernier interagit directement avec le sous-système de pilotage automatique, notamment durant les phases critiques de mise en route du système complet.

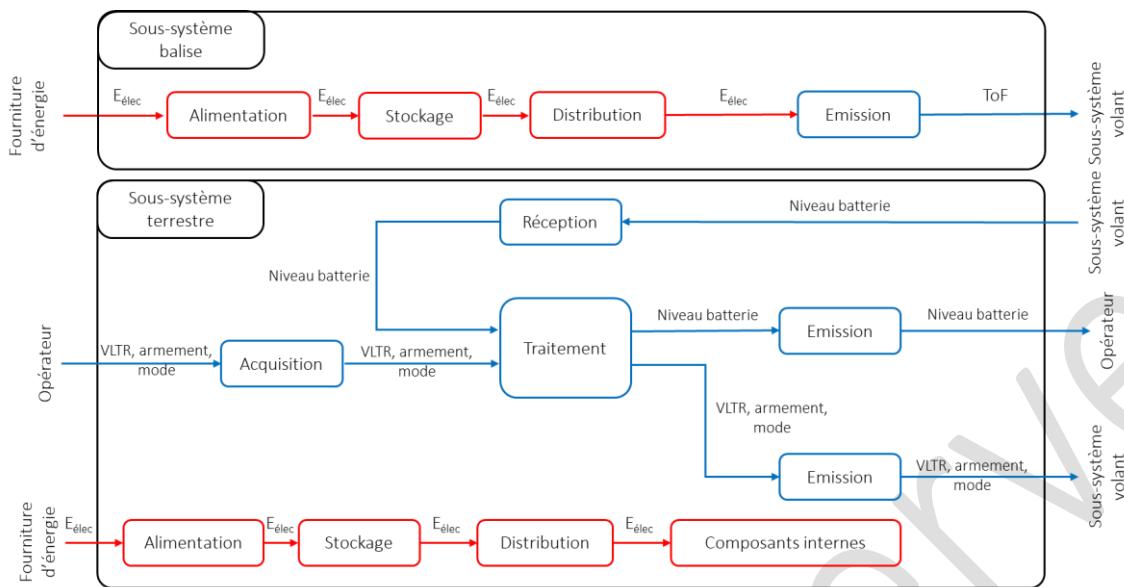


Figure 34 : Boîte grise des sous-systèmes terrestre et balise

Le sous-système terrestre, représenté sur la figure 34, joue un rôle central dans la gestion et le contrôle des interactions entre l'opérateur et le sous-système volant. Il permet la transmission des consignes, telles que le signal d'armement ou le mode de pilotage (manuel ou automatique), vers le sous-système volant, où elles sont interprétées et exécutées. En retour, il reçoit l'information du niveau de batterie, permettant une supervision en temps réel par l'opérateur. Par le biais d'une interface utilisateur intuitive, l'opérateur peut interagir efficacement avec le système, tandis que le sous-système terrestre assure le traitement, la validation et la centralisation des flux d'informations, garantissant ainsi un fonctionnement fluide et coordonné entre les différentes composantes du système.

D'un autre côté, le sous-système position/balise émet en continu un signal capté et traité par le sous-système volant, permettant de calculer le temps de vol de l'onde (Time of Flight, ToF). Ce principe, combiné à l'utilisation de plusieurs sous-systèmes balises, permet, grâce à une méthode de trilateration, de déterminer une position spatiale précise, indispensable au bon fonctionnement de notre système.

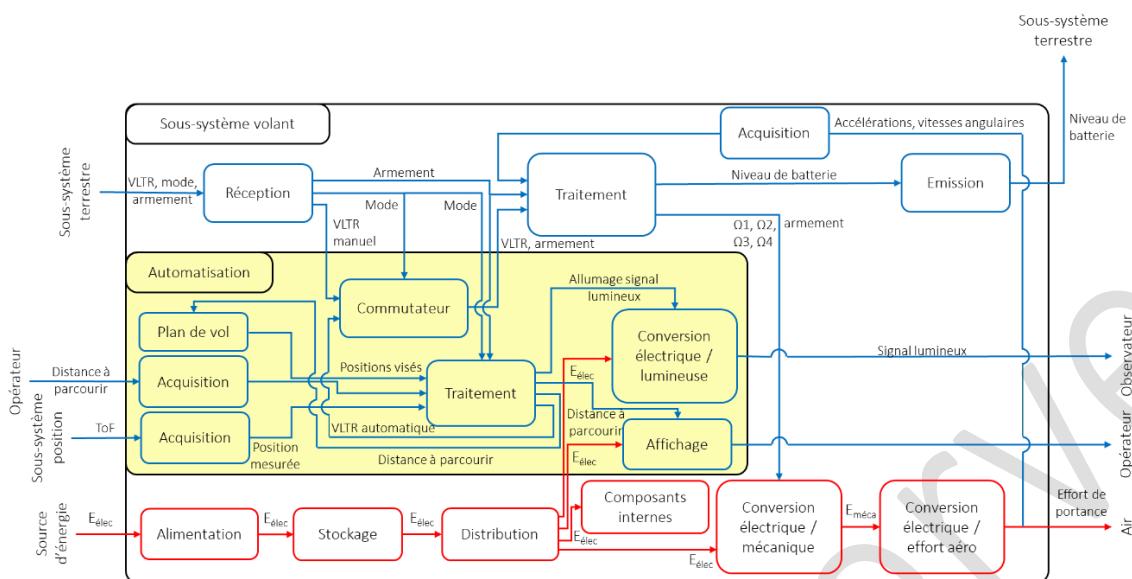


Figure 35 : Boîte grise du sous-système volant

Le sous-système volant, représenté sur la figure 35, constitue le cœur opérationnel du système global. Sa mission principale est d'exécuter les consignes transmises par le sous-système terrestre tout en assurant une navigation autonome et précise. Il traite en continu les données issues des balises de position pour calculer la position spatiale grâce au temps de vol de l'onde (ToF) et à la méthode de trilateration. Ces informations permettent d'ajuster en temps réel les trajectoires, la stabilité en vol, et les transitions entre phases. Le pilote automatique, intégré au sous-système volant, est chargé de ces tâches et gère également les moteurs, les angles de contrôle, et la poussée, garantissant une exécution fluide et précise du plan de vol lorsque le mode automatique est activé.

Le sous-système volant assure également des fonctions essentielles telles que la conversion d'énergie, l'armement du drone et les commandes du système en mode manuel. C'est à travers ce sous-système que le drone peut être armé ou désarmé, permettant ainsi le démarrage ou l'arrêt des moteurs. L'armement est activé via une commande spécifique détectée par le pilote automatique, déclenchant la séquence de vol. En cas de problème visible par l'opérateur, celui-ci peut alors basculer en mode dégradé durant le vol en basculant en mode manuel. Ce passage en mode dégradé désactive alors le pilote automatique jusqu'au prochain redémarrage, évitant ainsi tout conflit d'information entre les deux modes. En outre, le système permet à tout moment un désarmement immédiat et indépendant du pilote automatique, garantissant la sécurité de l'opérateur et des observateurs présents autour.

Ce sous-système, conçu pour fonctionner de manière autonome, repose sur des mécanismes robustes de contrôle en boucle fermée. Cela lui permet d'assurer la fiabilité et la sécurité des opérations en vol, tout en s'adaptant aux éventuelles contraintes imprévues.

Exigences techniques

A présent, afin de construire le tableau des exigences techniques recensant l'ensemble des caractéristiques que devront prévoir les différents composants, il est nécessaire de définir les différentes exigences en fonction de simulations numériques, de données constructrices et d'analyses expérimentales. Les caractéristiques seront par la suite testées afin de vérifier leur validité.

Exigence technique liée à l'acquisition

Les critères techniques liés à l'acquisition des données ont été définis à partir des simulations effectuées précédemment sous Simulink. Ces simulations, qui ont pour but de pousser le système dans ces retranchements, ont permis d'évaluer les tolérances acceptables pour le bruit, le biais, et le retard maximum que le système peut supporter.

Exigences technique liée au traitement

Les exigences relatives au traitement des données ont été établies sur la base des nombreux tests de pilotage réalisés en mode manuel. Ces tests, effectués avec le logiciel Mission Planner, ont permis de calibrer les paramètres du contrôleur de vol.

Des mesures expérimentales ont été prises en filmant les trajectoires du drone, puis en les analysant avec le logiciel Tracker. Ces données ont permis de déterminer, à l'aide également du ressenti de différents pilotes, les critères déterminants tels que la vitesse, l'accélération maximale, et le temps de réponse/stabilisation. Les résultats obtenus, illustrés à la figure 36, montrent par les résultats de la quantité de mouvement, que le drone présente une grande agilité sans oscillations indésirables. A noter que la courbe de mixage poussée/commande a été définie de manière linéaire afin de faciliter l'implémentation et réglage des PID sans avoir recours à des PID adaptatifs. De plus, ceci possède l'avantage pratique d'être intuitif pour n'importe quel opérateur débutant.

Quant aux angles, ceux-ci ont été réglés aux valeurs minimales acceptables définies par le contrôleur de vol dans Mission Planner afin de permettre au système d'être suivable de manière visuelle par l'opérateur et que celui-ci soit en mesure d'anticiper les mouvements du drone.

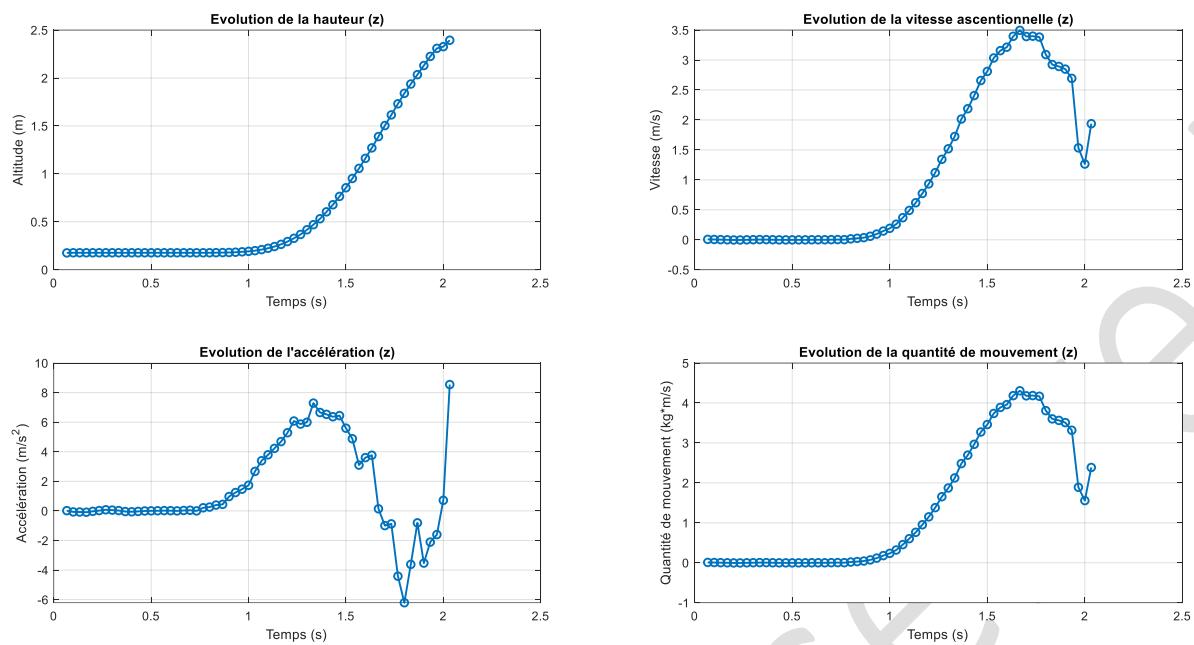


Figure 36 : Courbes des critères de pilotage en phase de montée

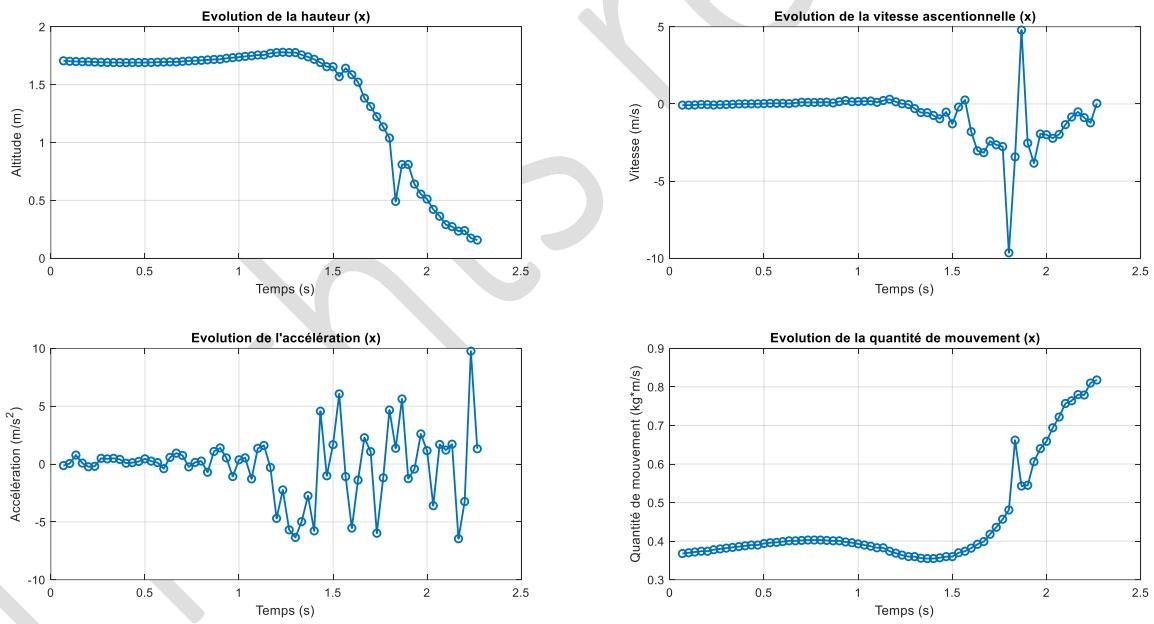


Figure 37 : Courbes des critères de pilotage en phase de descente

Exigence technique liée à la conversion électrique/lumineuse

Pour garantir la visibilité de la LED à une distance de 20 m dans une pièce éclairée, nous avons retenu une intensité lumineuse de l'ordre de 4000 mcd (0,01 lux), basée sur des calculs théoriques. En effet, en considérant l'environnement de travail de notre système comme étant une pièce fermée éclairé avec une lumière ambiante qui permet de distinguer l'ensemble de l'espace sans produire de fortes ombres ou d'éblouissements, on peut estimer que notre LED devrait avoir un éclairement d'environ $E = 0,01$ lux afin d'être visible. De ce fait, on peut alors en estimer l'intensité lumineuse nécessaire telle que :

$$I = E \cdot d^2 = 4000 \text{ mcd}$$

Où d est la distance de 20m au bout de laquelle la lumière doit être visible.

De plus, la lumière verte, la plus sensible à l'œil humain, a été privilégiée pour assurer une meilleure perception. Ce dispositif permettra d'informer l'opérateur et les observateurs de l'activation du système automatique.

Exigence technique liée à la conversion électrique/mécanique

L'exigence techniques de la conversion électrique/mécanique s'appuiera sur les moteurs AIR2216 KV880 fournis avec la base volante et illustré sur la figure 38.

D'après les données constructrices renseignés en figure 39, ceux-ci peuvent atteindre une vitesse maximale de 10 464 RPM pour une puissance consommée maximale de 259,2 W.



Figure 38 : Moteur AIR2216 KV880

Item No.	Propeller	Throttle	Voltage (V)	Torque (N*m)	Thrust (g)	Current (A)	RPM	Input power (W)	Efficiency (g/W)	Operating Temperature
AIR2216 KV880	T-motor T1045	50%	16	0.07	435	3.5	6015	56	7.77	53.5°C
		55%	16	0.08	527	4.6	6620	73.6	7.16	
		60%	16	0.09	608	5.6	7113	89.6	6.79	
		65%	16	0.11	702	6.8	7563	108.8	6.45	
		75%	16	0.13	888	9.5	8545	152	5.84	
		85%	16	0.15	1076	12.3	9442	196.8	5.47	
		100%	16	0.18	1293	16.2	10464	259.2	4.99	

Notes: Motor temperature is motor surface temperature @100% throttle running 10 mins.
 (Data above based on benchtest of 2018 are for reference only. Comparison with that of other motor types is not recommended.)

Figure 39 : Données constructeur des moteurs AIR2216 KV880

Concernant les hélices, nous utiliserons celles indiquées par la base volante, qui sont des T-motor T1045. Celles-ci sont représentés sur la figure 40.

De plus, d'après les données constructrices, il est indiqué que ces hélices sont en capacité de fournir une poussée maximale de 1,2kg.



Figure 40 : Hélices T-motor T1045

Connaissant le poids de notre système actuel, de 1,232kg, il est possible d'en déduire un rapport poids/poussée tel que :

$$r = (1,2 * 4)/1,232 = 3,89$$

Or, en sachant qu'il faut théoriquement avoir une valeur au moins égale à 2 pour permettre au système drone d'être maniable, cela permet d'offrir une marge confortable pour l'ajout de composants supplémentaires nécessaire au système de pilotage automatique. Pour plus d'information, veuillez consulter l'adresse suivante : [T-motor](#)

Exigence technique liée au stockage

En se basant sur la batterie LiPo 3700mAh, 16,8V fournie illustré sur la figure 41 également fournit avec la base volante, l'autonomie théorique a pu être calculée à l'aide du logiciel eCalc, largement utilisé par des entreprises comme Airbus, Boeing, Embraer et bien d'autres. De fait de l'utilisation de la version gratuite, certains composants n'étaient pas disponibles. Les calculs ont donc été faits en prenant en compte des composants et paramétrages proches de notre cas. Les résultats obtenus sur la figure 42 restent toutefois représentatifs des performances réelles.



Figure 41 : Batterie LiPo 3700mAh, 16,8V

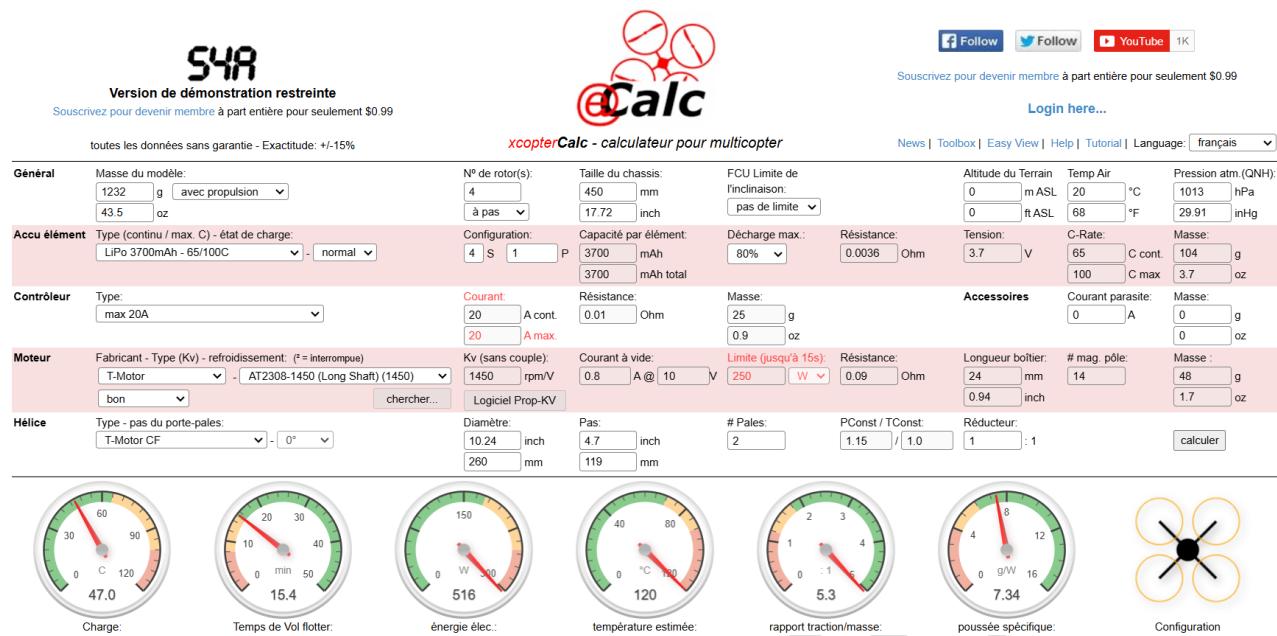


Figure 42 : Simulation de l'autonomie de la batterie dans la configuration initiale

Exigence technique de distribution

Cette exigence technique est déjà remplie par le module Holybro PM06 V2, qui est également un des composants fournit en même temps que la base volante. Ces caractéristiques du constructeur fournissent les spécifications techniques nécessaires pour la distribution d'énergie, garantissant un fonctionnement stable. Pour plus d'information, veuillez consulter l'adresse suivante : [Holybro PM06 V2](#)

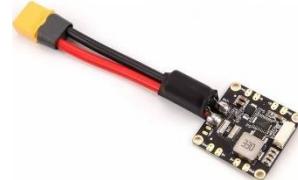


Figure 43 : Distributeur Holybro PM06 V2

Exigence technique liée à la réception et à l'émission

Les spécifications des composants de communication avec le récepteur FS-iA6B et la radiocommande FlySky FS-i6 définissent les paramètres d'émission et de réception. Les données constructrices pour chacun de ces composants permettent de définir les différents critères à suivre. Cependant, la fréquence de rafraîchissement a, quant à elle, dû être mesurée expérimentalement, comme illustré dans la figure 44. Pour plus d'information, veuillez consulter l'adresse suivante : [Flysky](#)



Figure 44 : FlySky FS-i6 et FS-iA6B

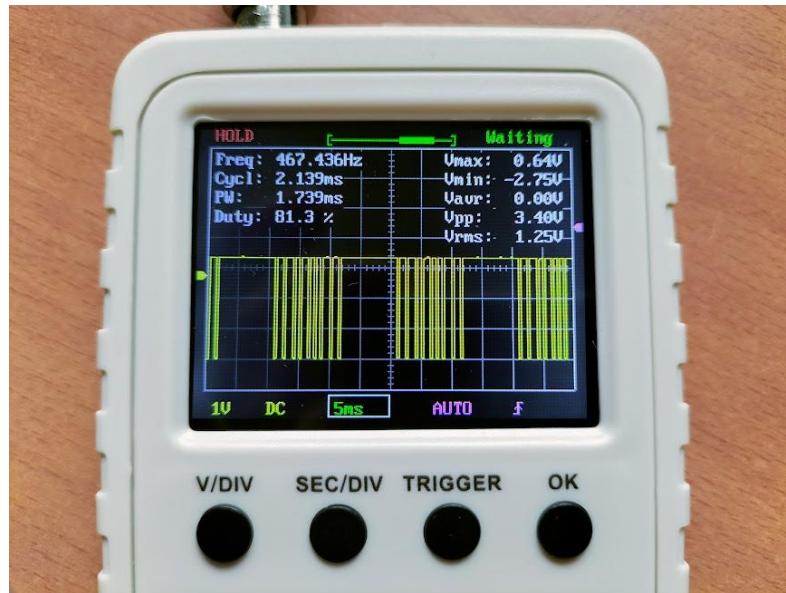


Figure 45 : Mesure de la fréquence de rafraîchissement de la communication radio

Tableau des exigences opérationnelles techniques

Ainsi, à l'aide de l'ensemble des informations et données énoncées jusqu'à présent, nous sommes maintenant en mesure de construire le tableau des exigences opérationnelles techniques suivant :

Tableau 3 : Tableau des exigences opérationnelles techniques

Numéros	Exigences techniques	Descriptions	Critères
EOT1	Acquisition	Le système acquiert les données nécessaires afin d'assurer le contrôle de la trajectoire en mode automatique	Bruit maximal acceptable : 0,352 m Biais maximal acceptable : 0,024 m Retard maximal acceptable : 0,12 s
		Le système acquiert la donnée relative à la distance à parcourir afin d'effectuer le vol en mode automatique	Distance comprise entre 0 et 15m
EOT2	Traitement	Le système traite les données acquises et ajuste la trajectoire en temps réel	
		Accélération : < 2 m/s ² Vitesse : < 0,5 m/s Erreur admise pour le système lors de ces mouvements par rapport à la position idéal : +/- 25cm	
			Courbes de mixage linéaires

		Le système traite les données acquises et se stabilise en temps réel et en permanence	Lacet : [-40 ; 40] °/s Tangage : [-10 ; 10] ° Roulis : [-10 ; 10] ° Temps de stabilisation : 0,1s Temps de réponse : 0,2s
EOT3	Conversion électrique/lumineuse	Le système convertit l'énergie électrique en énergie lumineuse afin de rendre visible le signal lumineux nécessaire au vol automatique	Intensité lumineuse nécessaire pour être vu à au moins 20m : 4000 mcd Longueur d'onde la plus sensible pour l'œil humain : 540nm
EOT4	Commutation	Le système commute entre le signal de commande manuel et le signal de commande automatique afin de correspondre au mode de pilotage souhaité par l'opérateur	Retour visuel de la commutation par un signal lumineux pour l'utilisateur
EOT5	Affichage	Le système affiche la valeur de la distance à parcourir souhaitée par l'opérateur	Retour visuel de la distance à parcourir
EOT6	Conversion électrique/ mécanique	Le système convertit l'énergie électrique en énergie mécanique afin de produire le mouvement nécessaire au vol	Régime moteur : <=10464RPM Puissance consommée : <259,2W
EOT7	Conversion mécanique/ Effort aérodynamique	Le système convertit l'énergie mécanique en effort aéro afin de produire la force nécessaire au vol du système	Rapport poussée/poids au moins égal à 2 Nombre d'éléments de conversion d'énergie : 4 Poussée : < 1,2kg
EOT8	Alimentation	Le système est alimenté en énergie électrique afin de pouvoir fonctionner	Branchemet via prise sur secteur 230V CC
EOT9	Stockage	Le système stocke l'énergie électrique afin de pouvoir réaliser des vols sans fils et en autonomie énergétique	Capacité : 3700mAh 14,8V CC Environ 15min d'autonomie
EOT10	Distribution	Le système distribue la juste énergie nécessaire au bon fonctionnement de chaque organe interne du drone	Courant : <3A Tension : <5V DC

		hors convertisseur électromécanique	
		Le système distribue la juste énergie nécessaire au bon fonctionnement des convertisseur électromécanique	Courant : >70A Tension : >14,8V DC
EOT11	Emission	Le système émet un signal continu vers le sous-système terrestre afin de notifier de l'état du stockage d'énergie interne	Fréquence d'émission : 2,4GHz Fréquence de rafraîchissement : 467Hz Type de branchement : PPM (Pulse Position Modulation)
EOT12	Réception	Le système réceptionne les signaux émis par la télécommande afin de dicter la conduite à suivre	Channels : 6 Fréquence d'émission : 2,4GHz Plage de valeurs des channels : 1000 - 2000µs Fréquence de rafraîchissement : 467Hz Résolution : 1049

Conception physique

Exigences physiques

La conception physique du drone consiste à répondre à la question : « Avec quels composants ? », en s'appuyant sur les exigences techniques établies dans le tableau des spécifications. Nous avons identifié deux catégories de composants : ceux fournis avec la base volante initiale et ceux ajoutés pour implémenter le mode de vol automatique.

Composants de la base volante initiale

Les éléments constituant la base volante initiale sont les suivants :

Tableau 4 : Eléments physiques de la base volante initiale

Elément	Quantité	Description
Moteur	2	CW AIR226II 920kv T-Motor
Moteur	2	CCW AIR226II 920kv T-Motor
ESC	4	AIR 20A T-Motor
Hélice	2	CW Holybro 1045 S500 v2
Hélice	2	CCW Holybro 1045 S500 v2
Contrôleur de vol	1	Radiolink Crossflight V1.0
Châssis	1	F450
Batterie	1	Tatu 4S 3700mA.h 45C
Radio commande	1	FLY Sky FSI6
Emetteur	1	FS-iA6B
Module de distribution d'alimentation	1	Holybro PM06 V2

Ces composants de base sont adaptés pour un vol manuel. Cependant, leur compatibilité avec les nouvelles exigences du mode de vol automatique a été confirmée grâce aux caractéristiques techniques. Nous allons donc les conserver et y adjoindre les composants manquants permettant de répondre aux exigences attendues.



Exigences physiques

Au regard des exigences techniques du drone avec le mode de vol automatique implémenté, les composants listés dans le tableau des exigences physiques suivant ont été sélectionnés afin de satisfaire les différents critères.

Tableau 5 : Tableau des exigences physiques

Numéros	Exigences techniques	Exigences physiques associées
EOP1	Acquisition	ESP32 UWB DW3000
		Ecran OLED 0.96
		AZDelivery keypad 4x4
EOP2	Traitement	ESP32-WROOM-32
		Radiolink Crossflight V1.0
		Encodeur 8CH PWM PPM SBUS
EOP3	Conversion électrique/lumineuse	LEDs verte/rouge 5mm
EOP4	Commutation	Protronik 4ch trainer
EOP5	Conversion électrique/mécanique	T-motor T1045
xEOP6	Conversion mécanique/effort aéronautique	AIR2216
EOP7	Alimentation	Chargeur type M7 ToolkitRC
EOP8	Stockage	Tattu lipo 4S 3700mAh 45C
EOP9	Distribution	Holybro PM06 V2 Power Module
EOP10	Emission	FS-iA6B
EOP11	Réception	FlySky FS-i6

Justification des exigences physiques

La sélection des composants a été effectuée de manière rigoureuse afin de garantir une parfaite compatibilité avec les exigences techniques du système. En intégrant les éléments de la base volante initiale avec des composants supplémentaires choisis, le drone est capable d'effectuer des vols automatiques tout en maintenant une excellente maniabilité en mode manuel.

Pour l'acquisition des données, les balises ESP32 UWB DW1000 ont été choisies en raison de leur capacité à répondre aux critères stricts de bruit, biais, et retard (EOT1), essentiels pour le contrôle de trajectoire en mode automatique. La technologie UWB (Ultra-Wideband) se distingue des autres solutions, telles que le Wi-Fi, le Bluetooth ou la radio, par sa précision exceptionnelle en localisation et par sa faible sensibilité aux interférences environnementales. Contrairement au Bluetooth ou au Wi-Fi, qui ont une précision limitée (généralement de l'ordre du mètre dans des environnements intérieurs), l'UWB peut atteindre une précision de localisation de quelques centimètres, ce qui est crucial pour notre système. De plus, l'UWB consomme moins d'énergie que le Wi-Fi et reste plus robuste que la radio traditionnelle face aux perturbations. Pour plus d'information, veuillez consulter les adresses suivantes : [DW1000 datasheet](#) et [ESP32 UWB DW 1000](#)



Figure 46 : ESP32 UWB DW1000



Figure 48 : Keypad 4x4

Le clavier AZDelivery Keypad 4x4 et l'écran OLED 0,96 pouces ont été sélectionnés pour leur simplicité d'intégration et leur capacité à offrir un retour visuel immédiat à l'opérateur. Ces composants permettent de saisir directement les distances à parcourir (EOT5), avec un affichage clair et lisible, ce qui simplifie l'interaction avec le système.



Figure 47 : Ecran OLED 0,96 pouces

Le traitement des données est confié à la carte ESP32-WROOM-32, choisie pour sa puissance de calcul et sa polyvalence. Elle permet d'exécuter les algorithmes nécessaires à la gestion en temps réel des trajectoires, en respectant les exigences de stabilisation et de précision (EOT2). En complément, un encodeur 8ch PWM vers PPM

assure une communication fluide et fiable entre la carte ESP32-WROOM-32 et le contrôleur de vol, qui ne peut être commandé qu'avec des signaux PPM, avec suffisamment de broches pour faire passer tous les signaux nécessaires aux commandes du drone. Ce choix garantit une transmission correcte des commandes et une compatibilité optimale avec les composants existants.

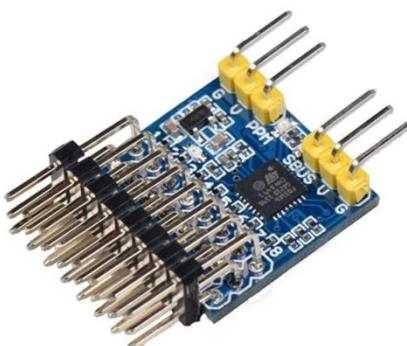


Figure 50 : Encodeur 8ch PWM vers PPM

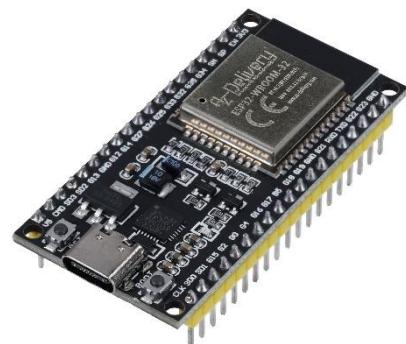


Figure 49 : ESP32-WROOM-32



Pour la signalisation lumineuse, les LEDs rouge et verte de haute intensité ont été retenues pour assurer une bonne visibilité, même en environnement éclairé. La couleur verte a été privilégiée, car elle correspond à la longueur d'onde la plus sensible pour l'œil humain, tandis que la LED rouge complète le dispositif en offrant une signalisation claire des modes manuel et automatique (EOT3).

Figure 51 : LED verte



Enfin, la commutation entre les modes manuel et automatique est réalisée à l'aide d'un commutateur Protronik 4ch trainer, sélectionné pour sa commutation en PWM, sa simplicité d'utilisation et sa fiabilité. Ce composant permet à l'opérateur de basculer rapidement entre les deux modes, tout en recevant un retour visuel immédiat pour confirmer la sélection (EOT4).

Figure 52 : Commutateur 4ch trainer

Quant aux autres composants, les moteurs AIR226II 920kv et les hélices T1045, en accord avec les exigences de puissance et de poussée (EOT6, EOT7), ainsi que la batterie Tattu 4S 3700mAh 45C, offrant une autonomie de 15 minutes (EOT9), forment le cœur de la propulsion et de l'énergie du drone. Enfin, le module d'alimentation Holybro PM06 V2 distribue l'énergie en respectant les contraintes de courant et de tension (EOT10), tandis que le récepteur FS-iA6B et la radiocommande FlySky FSI6 assurent une communication fiable et rapide entre le drone et l'opérateur (EOT11, EOT12).

Test de la chaîne de puissance

Afin de valider la choix de la motorisation et d'être certain que celle-ci satisfera le besoin, on réalise un test de la chaîne de puissance dont l'objectif est de quantifier la poussée produite par le moteur en fonction des vitesses de rotations sur un banc d'essai moteur.

Les hypothèses considérées sont :

- Le test n'est pas réalisé sur le drone
- L'essai est réalisé avec la batterie que celle utilisée sur le système volant à pleine charge

Exigences et Critères de Réussite

Le système doit vérifier l'ensemble des exigences techniques suivantes :

- Le système convertit l'énergie électrique en énergie mécanique afin de produire le mouvement nécessaire au vol (EOT6)
- Le système convertit l'énergie mécanique en effort aéro afin de produire la force nécessaire au vol du système (EOT7)

La cohérence des données reçues est primordiale. En effet, la poussée produite doit coïncider avec la vitesse de rotation de moteur.



Composants Testés

Les composants testés sont simplement le moteur en lui-même ainsi que l'hélice fixé sur celui-ci.

Méthode d'Essai

La méthodologie adoptée pour cet essai repose un banc d'essai moteur qui fournit la poussée en faisant varier les vitesses de rotation du moteur à partir de consigne envoyée depuis l'ordinateur lié au banc d'essai.

Conditions de l'Essai

L'environnement de test doit répondre à des conditions spécifiques. L'essai sera réalisé dans une salle fermée et au repos, à une température ambiante de 20°C et avec une humidité relative de 50 %. Aucune interférence radio ou perturbation ne devra être présente. Le test se fait sous cage.

Procédure de test

La procédure de test se déroule en plusieurs phases :

1. Initialisation
 - Configurer le banc d'essai
 - Mise sous tension du moteur
2. Acquisition

Collecte des données

La collecte des données s'effectue par étapes successives. Pour différentes vitesses de rotation des hélices sur le moteur, une mesure est effectuée en actionnant un bouton.

Ces données sont directement enregistrées dans un fichier Excel. Les acquisitions sont réalisées à l'aide de jauge spécifiques, permettant de mesurer diverses grandeurs physiques, notamment les forces et le couple.

Validation des résultats

La validation des résultats a été réalisée sur le drone complet et confirmée a posteriori. Ces résultats ont permis d'effectuer des tests sans les hélices, qui ont ensuite été approuvés par les différentes autorités compétentes.

Conclusion

Ce test a été essentiel, car il nous a permis de réaliser l'ensemble de nos autres tests et de garantir que notre drone soit certifié capable de remplir les objectifs définis, sans prendre de risques en le testant dans des simulations réelles. Il constituait donc une étape indispensable dans notre stratégie de validation.



Figure 53 : Vue de face du banc d'essai

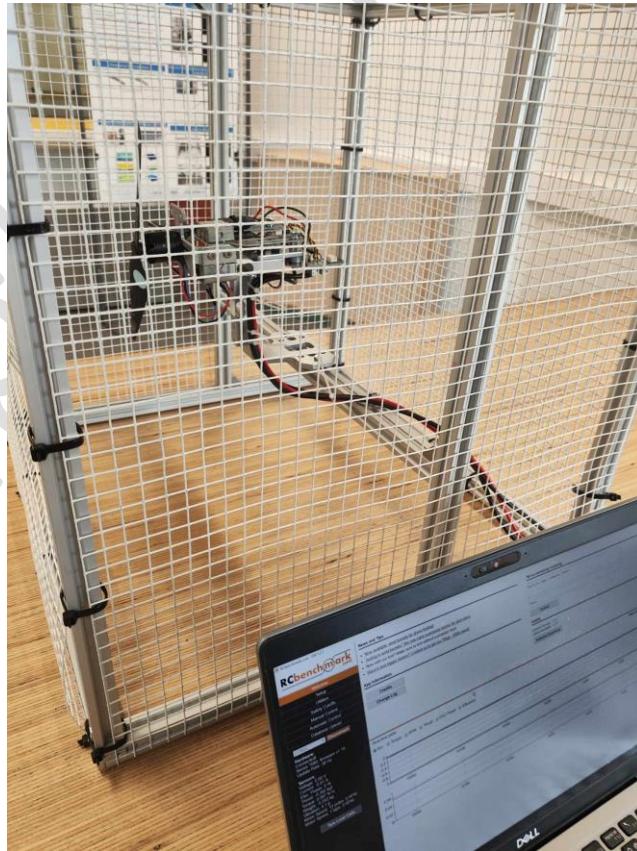


Figure 54 : Vue de côté du banc d'essai

Conception du support

Lors des tests de stabilité du drone, nous avons constaté que les capteurs, placés sur le châssis pour assurer la stabilité et la connexion avec l'utilisateur, avaient tendance à se déplacer significativement pendant le vol. Pour résoudre ce problème, un support dédié a été modélisé et conçu pour maintenir les capteurs en position fixe tout au long des opérations. Cette conception devait répondre aux contraintes suivantes :

- Limiter la surface projetée : pour minimiser l'augmentation de la traînée aérodynamique.
- Masse réduite : pour éviter de déstabiliser le drone.
- Maintenir le centre de gravité proche de sa position d'origine pour préserver les performances en vol.
- Facilité d'accès : les capteurs doivent rester accessibles pour leur entretien ou leur modification.

Le choix de placer les composants électroniques au sommet du drone, plutôt qu'en dessous, repose sur plusieurs considérations pratiques et techniques. En premier lieu, cela permet de protéger les composants en cas d'atterrissement brusque ou de collision. Les composants situés au sommet sont mieux protégés qu'en dessous, où ils seraient directement exposés aux impacts. De plus, cette configuration permet d'avoir facilement accès aux composants pour les réglages ou la maintenance. Enfin, cela permet de grandement minimiser les interférences. En effet, placer les capteurs au sommet réduit les risques de perturbations causées par les moteurs, les hélices ou autres composants électriques situés en bas.

Version 1

Une première version de la base du support a été modélisée pour tester la compatibilité des emplacements avec les capteurs.

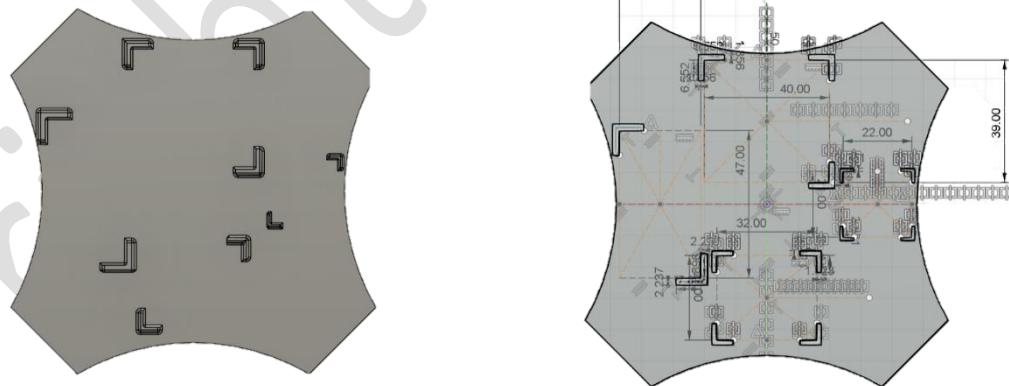


Figure 55 : Modèle 3D de la première version du support sous Fusion 360

Une impression test a révélé que seuls deux capteurs sur quatre étaient correctement positionnés. De plus, certains capteurs, en raison de leur proximité, interféraient avec le passage des câbles, entraînant un risque de déconnexion ou de déplacement pendant le vol.

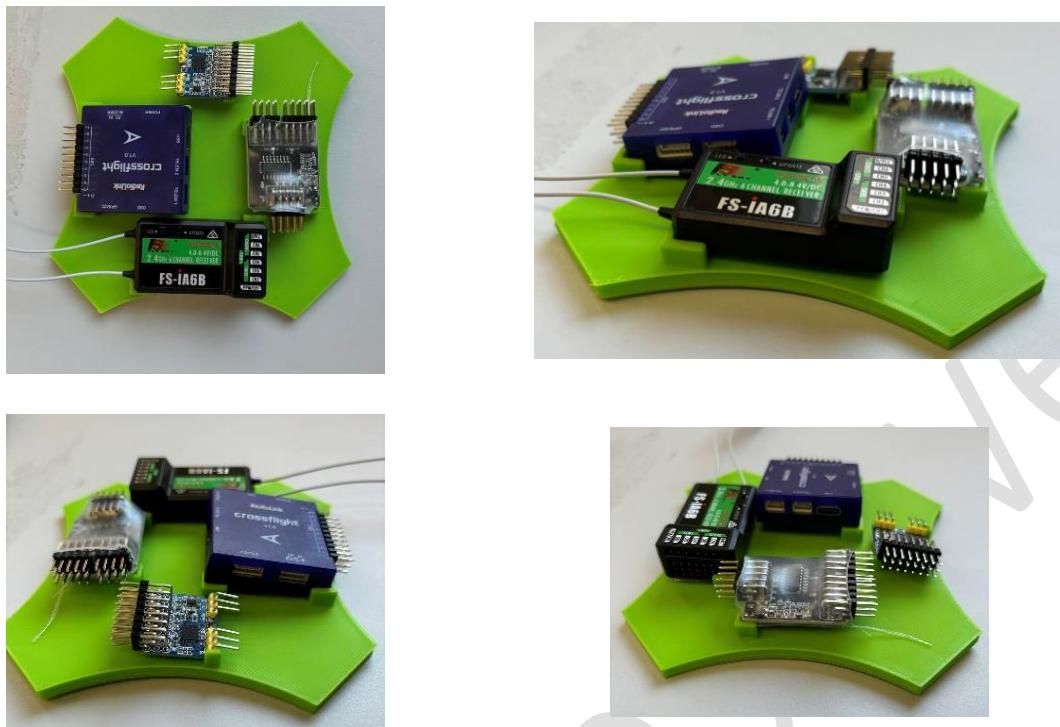


Figure 56 : Impression de la première version du support

Version 2

Pour résoudre les problèmes identifiés, plusieurs ajustements ont été apportés :

- Espacements augmentés : Les socles des capteurs ont été écartés de 0,5 mm pour éviter les interférences
- Translation des emplacements : Les socles ont été repositionnés pour prendre en compte les contraintes liées aux câbles
- Poids optimisé : Des "poches" ont été intégrées dans la structure pour augmenter la rigidité tout en réduisant la masse

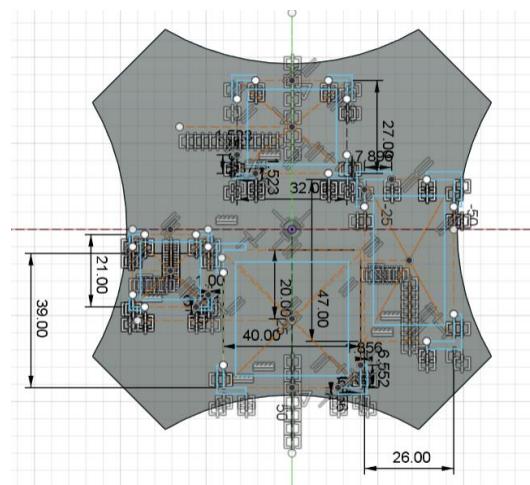
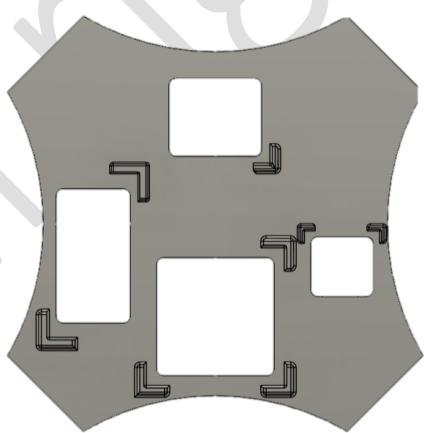


Figure 57 : Modèle 3D de la deuxième version du support sous Fusion 360

Après une nouvelle impression et un test d'intégration, cette version a été validée. Tous les capteurs étaient correctement fixés, et les câbles disposaient de l'espace nécessaire pour éviter les tensions ou les chevauchements.



Figure 58 : Impression de la deuxième version du support

Version finale du support

L'étape précédente était importante car celle-ci a permis de valider les espacements entre les capteurs, confirmant ainsi la bonne implémentation de l'ensemble. Ainsi, cette dernière version comporte également des piliers en arc, qui ont été ajoutés pour supporter l'étage supérieur, réservé à l'emplacement des cartes ESP32. Cette configuration a été choisie du fait que les arcs répartissent les contraintes de manière optimale, réduisant les risques de déformation ou de rupture en cas de chute ou de vibrations. De plus, suivant le même raisonnement que la version précédente, des "poches" supplémentaires ont été ajoutées pour alléger davantage la structure. A noter que le matériau utilisé pour l'impression 3D est le PLA, choisi pour sa rigidité, sa facilité d'impression et son accessibilité.



Figure 59 : Modélisation 3D de la version finale du support sous Fusion 360

Finalement, le résultat après impression est le suivant :

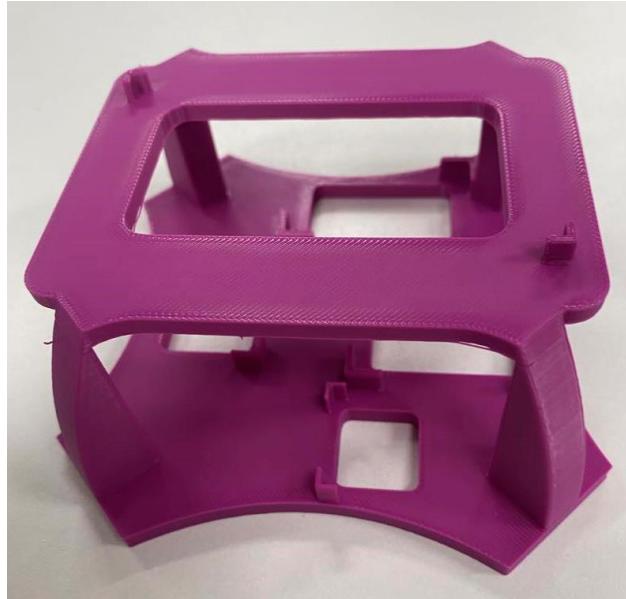


Figure 60 : Impression de la version finale du support

On peut également en extraire le plan industriel, en vue d'une fabrication commerciale :

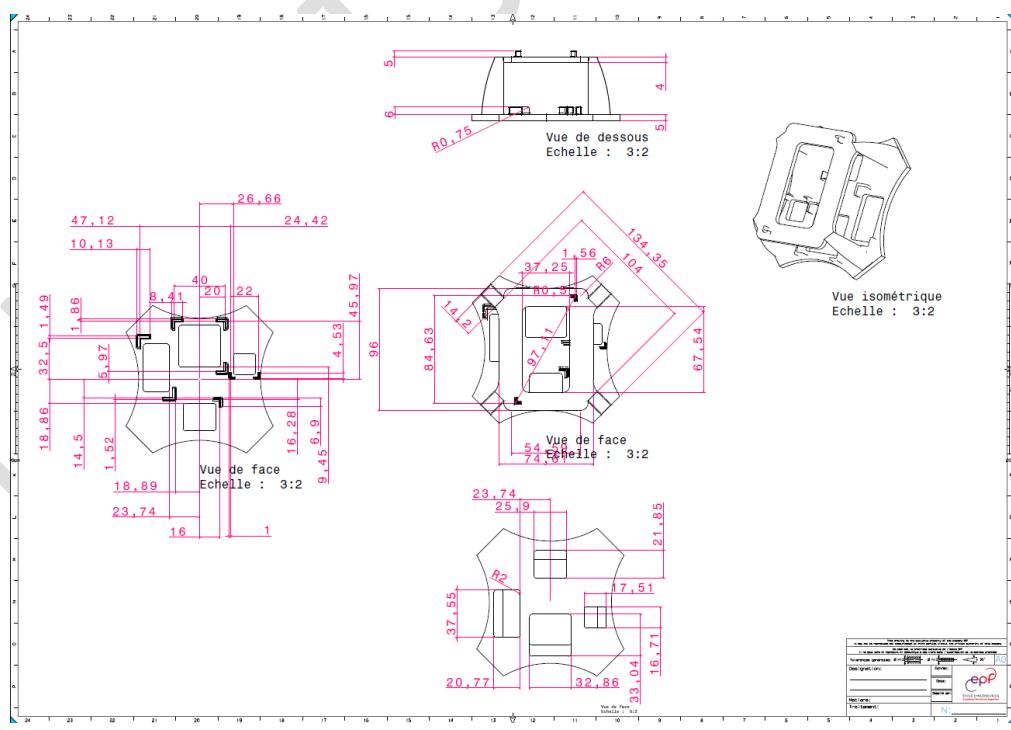


Figure 61 : Plan industriel du support

Modélisation numérique complète

Finalement, après assemblage complet de l'ensemble des composants sur le support et du support sur la base volante, le résultat ressemble à ceci :

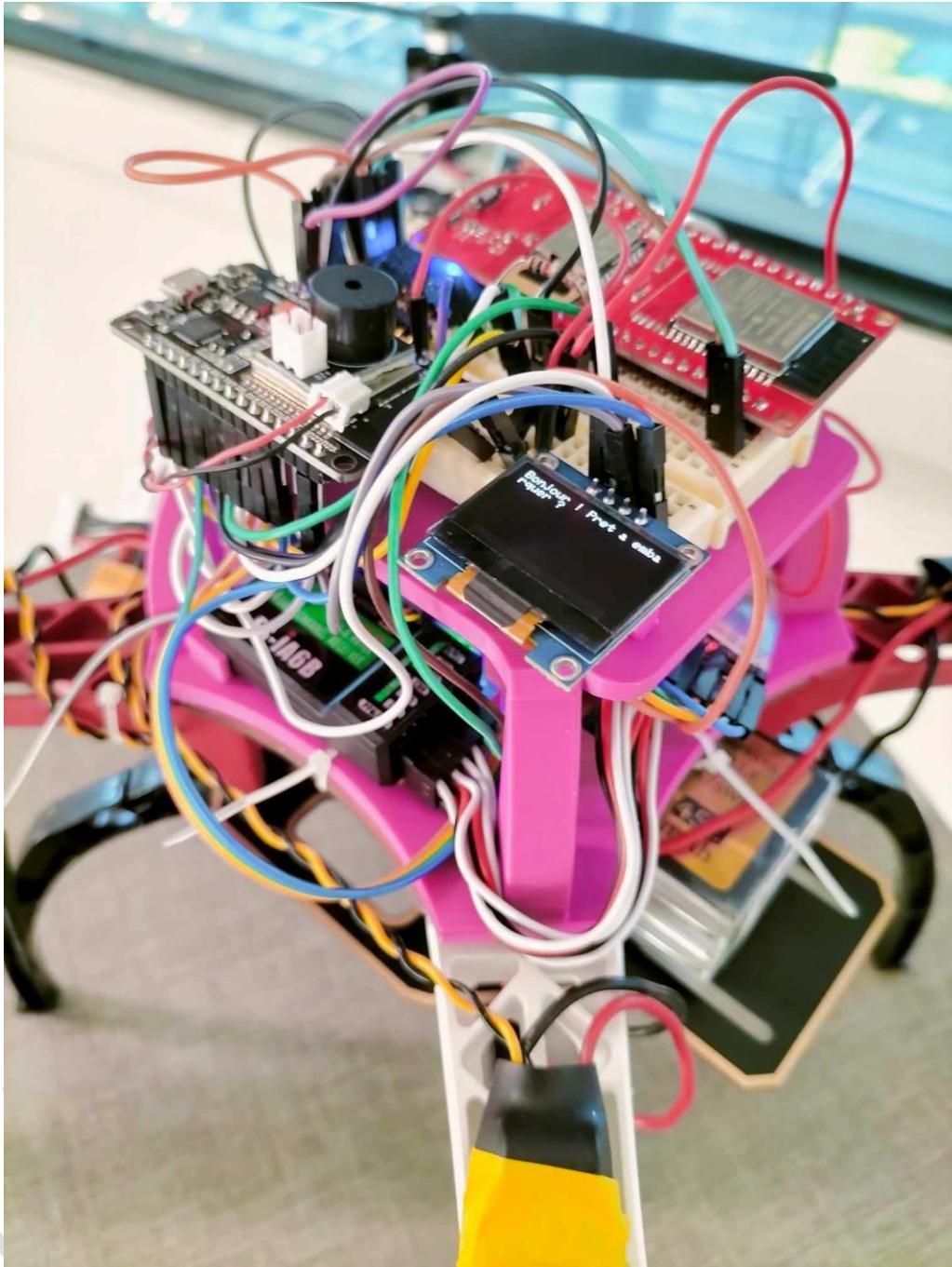


Figure 62 : Vue du support avec l'assemblage complet sur la base volante

Le support final, conçu pour maintenir l'ensemble des composants électroniques en position fixe, a été modélisé en 3D sous Catia V5.

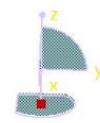


Figure 63 : Modélisation complète du système volant sous Catia V5

Cette modélisation permet d'évaluer l'impact du support sur le centre de gravité et la matrice d'inertie du drone.

Résultat

Mode de calcul : Exact

Type : Volume

Caractéristiques		Centre de gravité (G)	
Volume	6,249e-004m ³	Gx	1,057mm
Aire	0,403m ²	Gy	-1,094mm
Masse	1,32kg	Gz	24,648mm
Densité	Pas uniforme		

Inertie / G Inertie / O Inertie / P Inertie / Axe Inertie / système d'axe

Matrice d'inertie / G			
IoxG	0,011kgxm ²	loyG	0,012kgxm ²
IxyG	2,923e-005kgxm ²	IxzG	-2,129e-005kgxm ²
IyzG	3,839e-005kgxm ²		

Moments principaux / G			
M1	0,011kgxm ²	M2	0,012kgxm ²
M3	0,02kgxm ²		

Axes principaux			
A1x	0,998584	A2x	-0,053135
A1y	-0,053123	A2y	-0,998577
A1z	0,00269	A2z	0,004587
A3x	0,002442	A3y	-0,004724
A3z	-0,999986		

Garder la mesure corps principaux uniquement Créer la géométrie Export Personnaliser...

Mesurer uniquement les éléments visibles

Figure 64 : Résultats numériques du centre de gravité et des matrices d'inerties avec l'assemblage complet

Il est à noter que la masse du drone dans la modélisation correspond bien à celle du drone assemblé, de 1320g, mesuré expérimentalement, garantissant que les données extraites de la simulation reflètent bien le comportement réel du drone.



Figure 65 : Mesure expérimentale du poids du système volant

La matrice d'inertie joue un rôle fondamental dans la dynamique du drone. Une augmentation des moments d'inertie ralentit les réponses du drone aux commandes de rotation, ce qui peut réduire sa maniabilité. À l'inverse, une diminution de ces moments rend le drone plus réactif, mais au détriment de la stabilité. Il est donc essentiel d'analyser les différences entre les matrices d'inertie initiale et assemblée pour évaluer leur impact potentiel sur la stabilité et la pilotabilité.

Les matrices d'inertie exprimées en kg/m^2 sont les suivantes :

	I_{xx}	I_{yy}	I_{zz}	I_{xy}	I_{xz}	I_{yz}
Drone initial	0,01	0,011	0,019	$2,119 \cdot 10^{-5}$	$3,808 \cdot 10^{-6}$	$2,006 \cdot 10^{-5}$
Drone assemblé	0,011	0,012	0,02	$2,923 \cdot 10^{-5}$	$-2,129 \cdot 10^{-5}$	$3,839 \cdot 10^{-5}$
Ecart relatif	9%	8%	5%	28%	118%	48%

Les écarts observés sur les éléments de la diagonale de la matrice d'inertie atteignent jusqu'à 9 %, tandis que ceux des éléments hors diagonale atteignent 118 %. Ces derniers, bien que relativement importants, ont un impact moindre sur la stabilité globale du drone en raison de leurs valeurs absolues plus faibles. En effet, les éléments principaux étant bien plus élevés que les éléments secondaires, leur influence sur la dynamique du drone est prépondérante. Ainsi, un écart maximal de 9 % reste tolérable

et ne devrait pas compromettre la maniabilité, compte tenu de la robustesse de la loi de commande utilisée dans le contrôleur de vol.

Par ailleurs, le déplacement du centre de gravité peut également affecter la stabilité en modifiant les moments de rotation autour des axes principaux. Un décalage du centre de gravité par rapport au centre de poussée peut entraîner une inclinaison spontanée du drone, nécessitant une compensation par une répartition asymétrique de la puissance des moteurs. Si ce déplacement dépasse une certaine limite, il pourrait dépasser les capacités de stabilisation du contrôleur de vol, risquant ainsi de compromettre le maintien d'un vol stable.

Les coordonnées du centre de gravité, exprimées en millimètres, sont récapitulées dans le tableau ci-dessous :

	G _x	G _y	G _z
Drone initial	0,0338	-1,489	20,737
Drone assemblé	1,075	-1,094	24,648
Ecart relatif	97%	36%	16%

Bien que les écarts relatifs puissent sembler significatifs, ils ne dépassent pas en réalité les quelques millimètres avec un écart maximal observé de 4 mm selon l'axe z. Comparés aux dimensions globales du drone, qui mesurent plusieurs dizaines de centimètres, ces déplacements restent négligeables. Par conséquent, ces variations n'impactent pas suffisamment la stabilité pour compromettre la pilotabilité du drone.

Enfin, pour plus de précision, on peut extraire un plan industriel détaillé du châssis de la base volante, et de ses pieds sous leurs différentes vues dans la perspective d'une production commerciale du système, en plus du plan industriel du support déjà présenté précédemment :

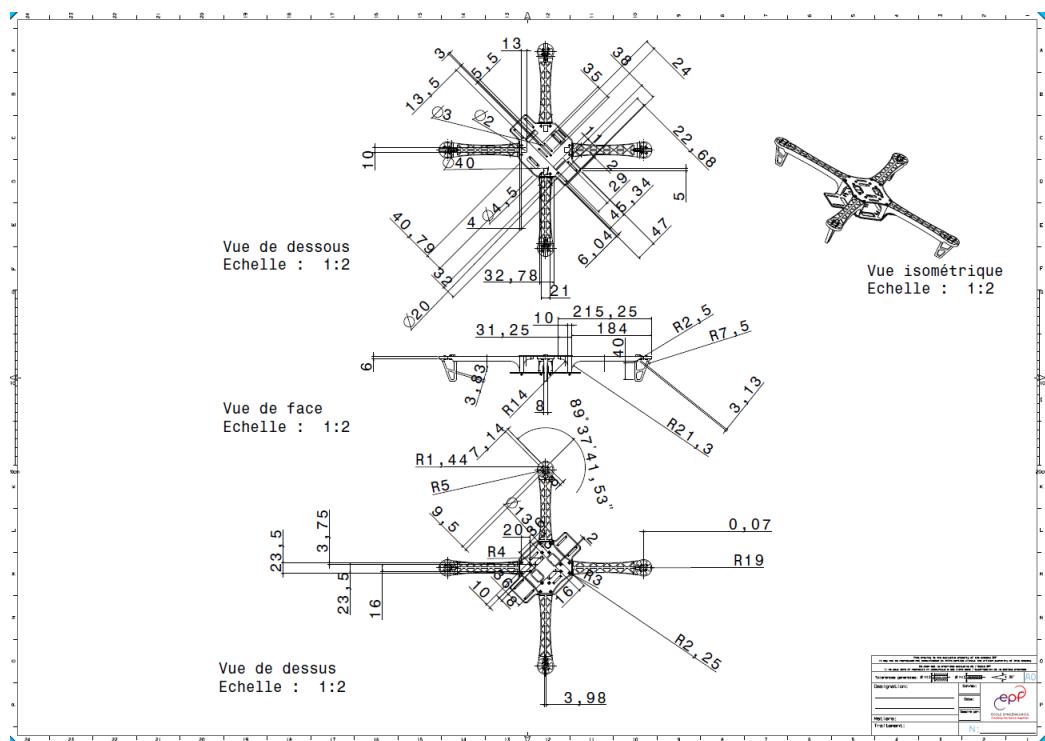


Figure 66 : Plan industriel du châssis de la base volante

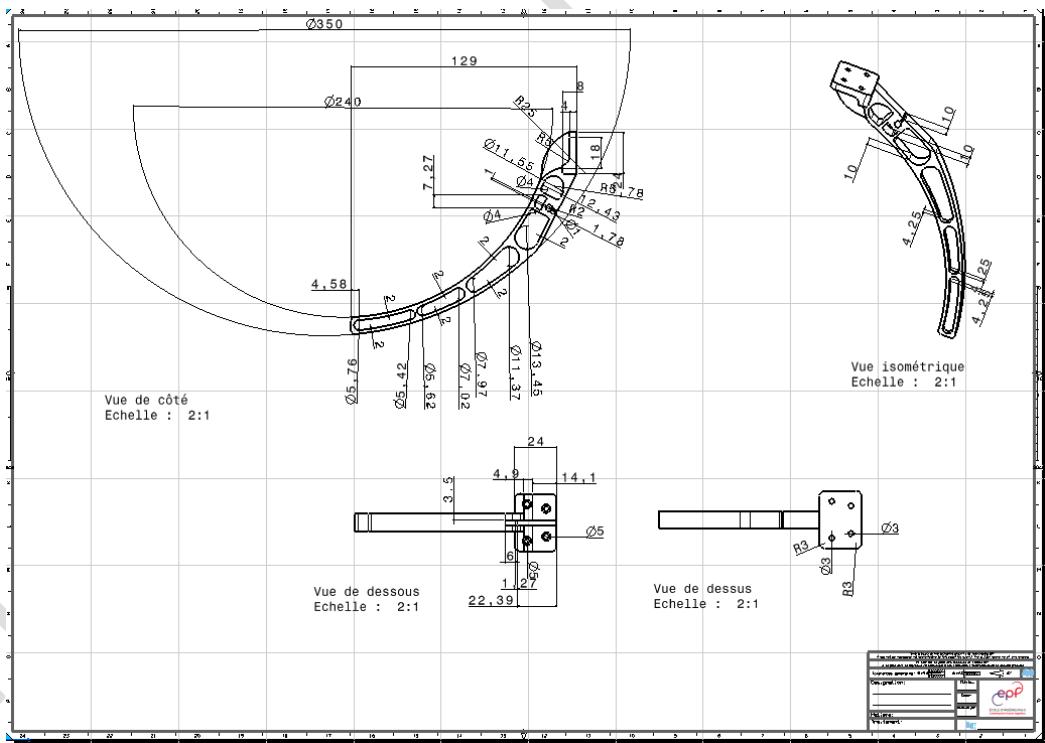


Figure 67 : Premier plan industriel des pieds de la base volante

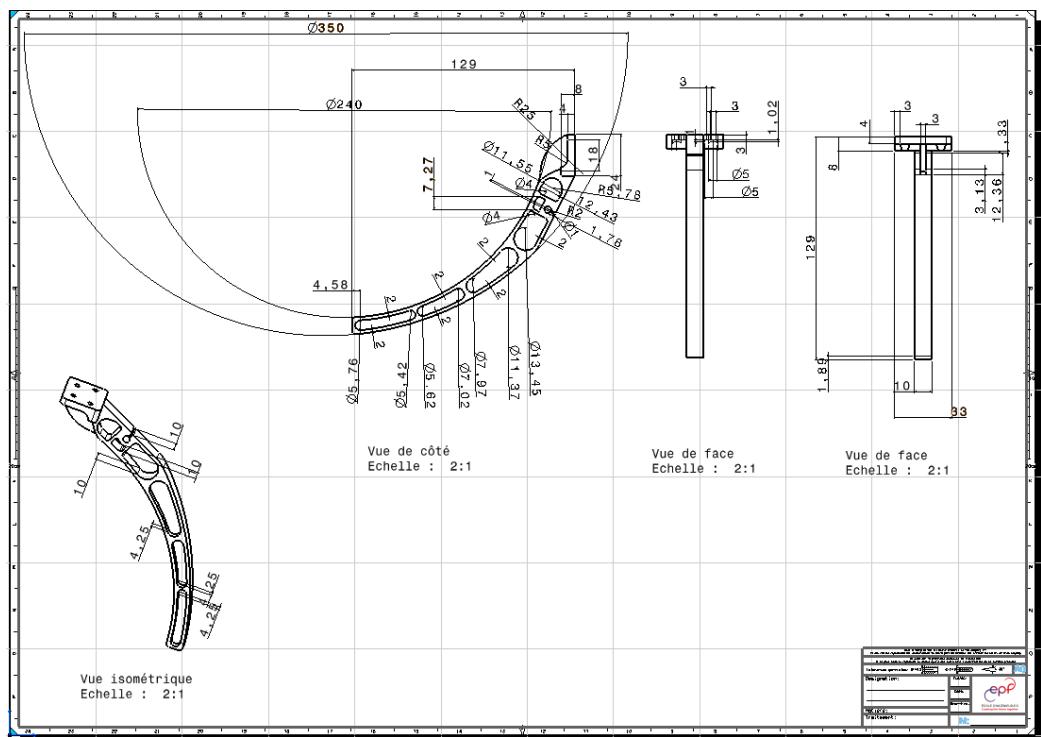


Figure 68 : Deuxième plan industriel des pieds de la base volante

Nouvelle évaluation de l'autonomie et de la manœuvrabilité avec la version complète du système

D'après les nouveaux paramètres de poids obtenus avec le nouveau système, il serait intéressant de simuler une nouvelle le système d'alimentation afin d'en connaître une nouvelle estimation de l'autonomie de la batterie. Pour se faire, et comme précédemment, on utilise le logiciel eCalc en y référençant le nouveau poids de 1,320kg. On obtient ainsi les résultats suivants :



Figure 69 : Simulation de l'autonomie de la batterie dans la configuration finale

Comme attendu, les valeurs obtenues sont en baisses par rapport aux résultats précédents. Cependant, en considérant d'après nos paramétrages et notre conception des cas d'utilisation de 6 vols consécutifs d'une durée d'environ 1min chacun dont 3 vols automatiques comme donnés dans le cahier des charges, l'autonomie de notre système sera suffisante et offrira une marge non négligeable qui permettra même d'étendre le domaine de vol jusqu'ici très limité.

Afin de se rassurer, on peut également quantifier à nouveau la manœuvrabilité de notre système grâce au rapport poussée/poids actuel, définit exactement comme précédemment à la différence de poids près, soit :

$$r = (1,2 * 4)/1,320 = 3,64$$

On peut ainsi en conclure, comme attendu que la valeur ne diffère que très peu de celle déjà calculé précédemment et que celle-ci reste toujours au moins égale à 2, garantissant ainsi une bonne manœuvrabilité de notre système.



Schémas électriques

Dans le cadre de notre projet, une piste d'amélioration envisagée serait de concevoir notre propre circuit imprimé intégrant tous les composants nécessaires au mode automatique ainsi que les branchements requis pour que le drone fonctionne en mode manuel. Cette solution permettrait d'optimiser l'assemblage et de faciliter une éventuelle production à grande échelle. Pour réaliser les schémas électriques, nous avons choisi d'utiliser KiCad 7.0, un logiciel open source offrant une grande flexibilité et une meilleure gestion des futures mises à jour. Ainsi, d'après les branchements entre les différents composants se feraient suivant le câblage actuel de notre système :

ESP32	ESP32 UWB DW1000	Commutateur	Encodeur	FS- IA6B	Contrôleur de vol	LED verte	LED rouge	Ecran OLED	Keypad	Onduleur	Batterie
21								SDA			
22								SCL			
18		AIL élève									
19		ELE élève									
2		THR élève									
23		RUD élève									
15		SW1		CH5							
35			6	CH6							
5							VCC				
4						VCC					
16	17										
17	16										
VIN	5V0									5V	Borne +
3V3		VCC	VCC	VCC	VCC			VCC			
GND	GND	GND	GND	GND	GND	GND	GND	GND		GND	Borne -
13									1		
12									2		
14									3		
27									4		
26									5		
25									6		
33									7		
32		AIL sortie	1						8		

		ELE sortie	2							
		THR sortie	3							
		RUD sortie	4							
		PPM		PPM						
		AIL maître		CH1						
		ELE maître		CH2						
		THR maître		CH3						
		RUD maître		CH4						

Sur cette base, nous pouvons construire le schéma électrique complet du système suivant :

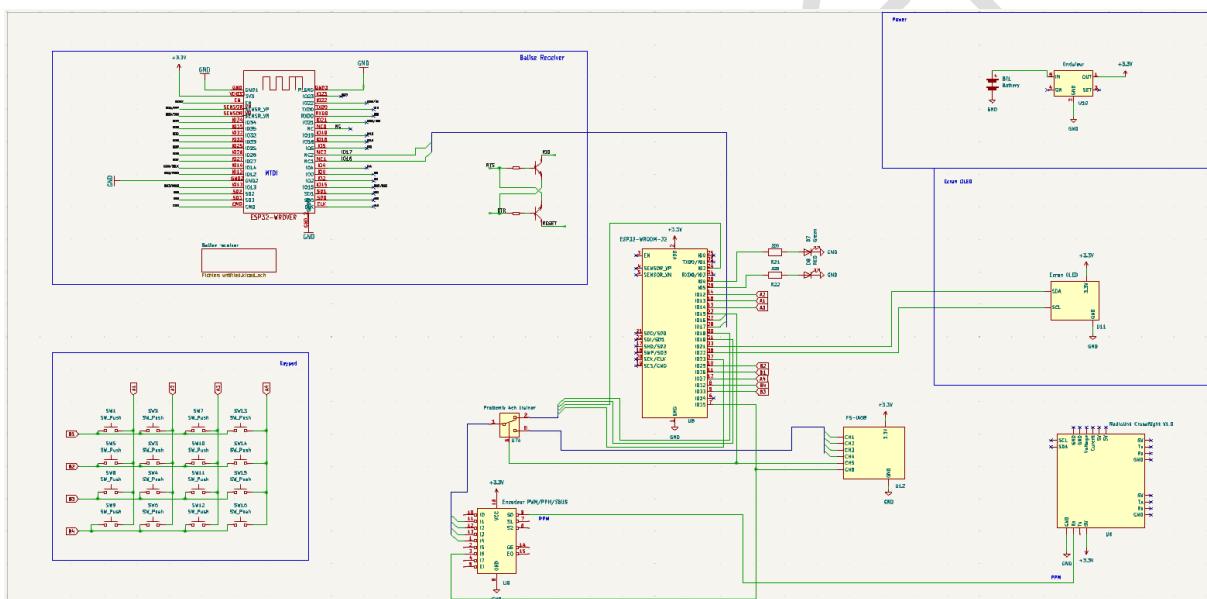


Figure 70 : Schéma électrique autour de l'ESP32-WROOM-32

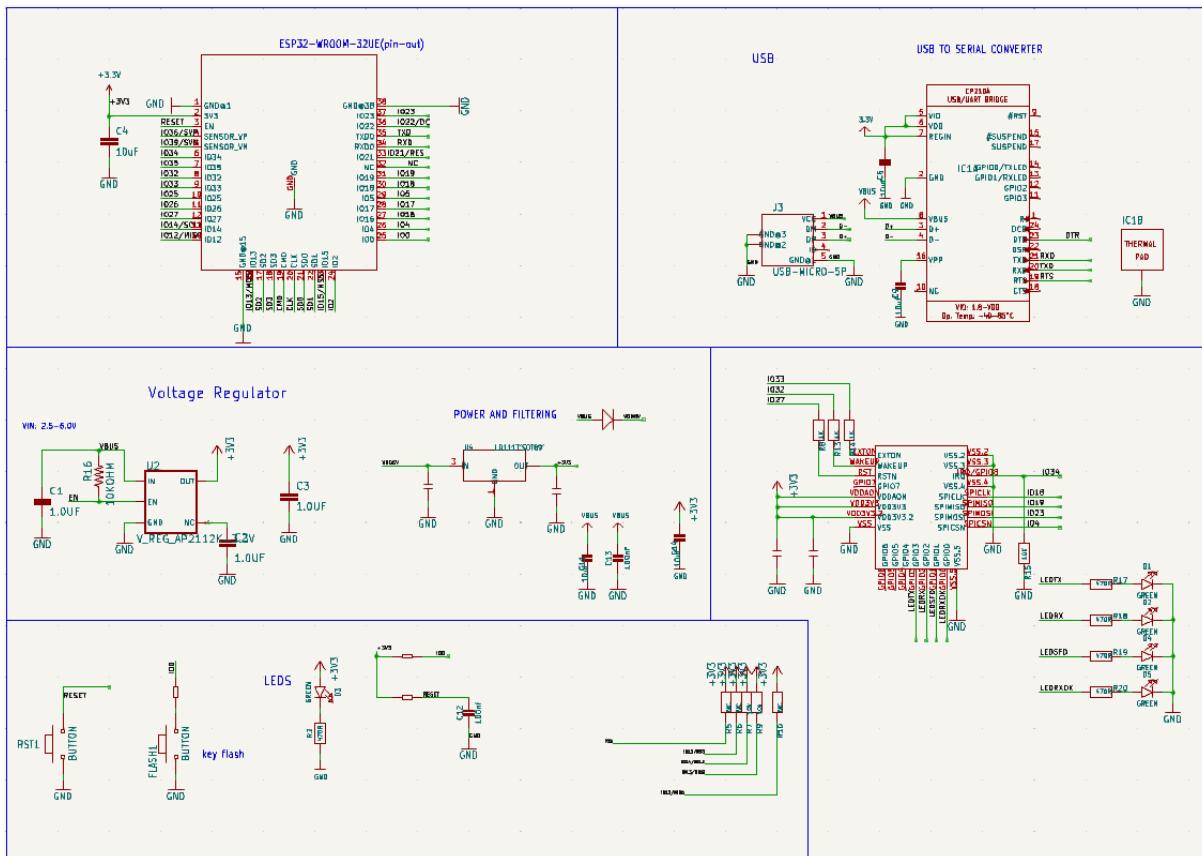


Figure 71 : Schéma électrique du récepteur de position ESP32 UWB DW1000



Tests physiques

Test de la base volante

Objectif d'Essai

L'objectif de cet essai est de vérifier le bon fonctionnement de la base volante dans sa intégralité afin de s'assurer d'avoir une base de travail fiable et fonctionnel pouvant acquérir le pilote automatique.

Les validations attendues sont :

- Réception et transmission des signaux entre la radiocommande et le contrôleur de vol avec 6 channels en entrée
- Stabilisation et pilotage du drone possible sans grande difficulté
- Retour sonore lors de l'armement et le désarmement

Les hypothèses considérées sont :

- La totalité du système de pilotage automatique n'est pas installé sur le système volant
- L'environnement n'est pas altéré par des interférences

Exigences et Critères de Réussite

Les exigences techniques que doit respecter le système sont les suivantes :

- Le système doit traiter l'information pour générer des consignes de vitesses de rotation des moteurs cohérentes (EOT2)
- La communication entre la radiocommande et le système volant est fiable, permanente, cohérente et les informations envoyées par la radiocommande sont correctement interprétées (EOT11 & EOT12)

Les critères de performance permettent d'évaluer la qualité de la mise en œuvre. La cohérence des commandes est primordiale et celles-ci doivent être correctement interprétées par le contrôleur de vol. De plus, l'ensemble du système doit être relativement facile de prise en main en minimisant au maximum les dérives.

Composants Testés

Différents composants essentiels au bon déroulement de cet essai seront testés :

- Le contrôleur de vol, qui transmet les consignes aux moteurs
- Les moteurs, qui permettent de faire tourner les hélices produisant la poussée
- Le récepteur radio, qui reçoit les consignes envoyées par la radiocommande
- La radiocommande, qui permet à l'opérateur de transmettre ses consignes
- Le haut-parleur intégré, qui avertit l'opérateur lors de l'activation ou de la désactivation de l'armement

Méthode d'Essai

La méthodologie adoptée pour cet essai repose sur une méthode assez simple. Le système volant sera d'abord alimenté de manière autonome, sans connexion au sol. Ensuite, l'opérateurs enverra différentes consignes et commande au système volant via la radiocommande sans plan de vol précis. L'objectif ici étant d'observer les réactions du système et de pousser celui-ci dans ces limites électroniques. L'ensemble du test sera observé et analysé de manière qualitative.

Cette méthode est justifiée par le fait de ne pas avoir de plan de vol précis. Il s'agit ici simplement de vérifier le bon fonctionnement de notre système de base et de tester ces limites dans une certaine mesure.

Conditions de l'Essai

L'environnement de test doit répondre à des conditions spécifiques. L'essai sera réalisé dans une salle fermée et au repos, à une température ambiante de 20°C et avec une humidité relative de 50 %. Aucune interférence radio ou perturbation ne devra être présente.

Les paramètres variables au cours de l'essai sont les consignes de vol liées au pilotage ainsi que la consigne d'armement.

Procédure de test

La procédure de test se déroule en plusieurs phases :

1. Initialisation
 - Connecter tous les composants du système volant entre eux
 - Mettre la radiocommande et la base volante sous tension
2. Armement du système
3. Séquence de vol
4. Désarmement du système

Collecte des données

Les données recueillies au cours de cet essai seront uniquement visuelles afin de s'assurer du bon fonctionnement du système, afin de le prendre en main et de sentir de manière qualitative où les limites électroniques se trouvent.

Validation des résultats

Les résultats de cet essai seront considérés comme validés si plusieurs critères sont remplis :

- Signal sonore synchronisé avec l'armement
- Réponse correcte des moteurs en fonction des actions désirées
- L'ensemble des commandes de la radiocommande sont validés

Conclusion

Après avoir effectué les différents essais, notre interprétation des résultats montre que tous les critères de performance ont été respectés. Le drone répond bien aux commandes voulues tandis que l'armement et le désarmement se déroulent sans accro.



Figure 72 : Test de la base volante sous la supervision de nos autorités techniques



Figure 73 : Assemblage complet de la base volante



Test d'acquisition des coordonnées dans l'espace

Objectif d'Essai

L'objectif de cet essai est de valider le bon fonctionnement des balises de positionnement dans le cadre de notre projet de localisation. Ces tests visent à s'assurer que le système permet une acquisition précise des positions et un traitement efficace des données pour le pilotage automatique du drone.

Les validations attendues sont :

- Acquisition des données de position des balises au sol et de la balise principale sur le drone
- Traitement des données acquises pour ajuster la trajectoire du drone en temps réel

Les hypothèses considérées sont :

- Les balises sont positionnées à des distances suffisantes pour éviter toute interférence mutuelle
- L'essai est réalisé dans un environnement contrôlé, sans perturbations radiofréquences extérieures

Exigences et Critères de Réussite

Les exigences techniques que doit respecter le système sont les suivantes :

- Acquisition des données de position des balises (EOT1)
- Traitement des données acquises pour ajuster la trajectoire (EOT2)
- Angle de correction en roulis et tangage afin de corriger la position

Les critères de performance permettant d'évaluer la qualité de la mise en œuvre sont :

- La précision des coordonnées acquises par le système
- La cohérence et la fiabilité des ajustements de trajectoire effectués par le drone

Composants Testés

Les différents composants essentiels qui seront testé lors de cet essai seront :

- Les trois ESP32 UWB DW1000 en mode balise au sol
- L'ESP32 UWB DW1000 en mode récepteur sur le drone

Méthode d'Essai

La méthodologie adoptée repose sur la création d'un repère à l'aide des balises de positionnement.

Voici les étapes du test :

1. Disposer les trois balises fixes au sol de manière à former un triangle à l'aide de scotch qui sert de repère visuel
2. Configurer une balise comme centre du repère, une autre comme extrémité de l'axe x et la troisième comme extrémité de l'axe y
3. Positionner la carte de réception sur le drone
4. Alimenter les balises et visualiser les coordonnées en temps réel sur le moniteur série de l'ordinateur branché à la carte réceptrice
5. Faire varier la position du drone et observer les variations des coordonnées acquises

Cette méthode permet de valider les exigences en s'assurant que le système acquiert correctement les coordonnées et les traite pour ajuster la trajectoire en temps réel.

Conditions de l'Essai

L'essai est réalisé dans une salle fermée et au repos, à une température ambiante de 20°C et avec une humidité relative de 50 %. Aucune interférence radio ou perturbation ne doit être présente.

Les paramètres variables au cours de l'essai sont :

- La position des balises
- Les consignes de vol et de déplacement du drone

Procédure de Test

La procédure de test se déroule en plusieurs phases :

1. Initialisation
 - Installer et connecter les balises au sol
 - Alimenter les balises et lancer le logiciel Arduino pour visualiser les données
2. Positionnement des balises
3. Acquisition des données en déplaçant le drone au sol
4. Traitement des données

Collecte des Données

Les données recueillies lors de cet essai sont simplement les coordonnées différentes coordonnées acquises par la carte réceptrice sur le drone d'après la position des balises au sol.

Validation des Résultats

Les résultats de cet essai seront considérés comme validés si plusieurs critères sont remplis :

- Le système acquiert correctement les coordonnées des balises
- La cohérence et la fiabilité des coordonnées acquises sont satisfaisantes

Conclusion

Après avoir effectué les différents essais, notre interprétation des résultats montre que tous les critères de performance ont été respectés. Le système se positionne correctement dans le repère définit en transmettant des coordonnées cohérentes et se corrige de manière cohérente dans le plan.

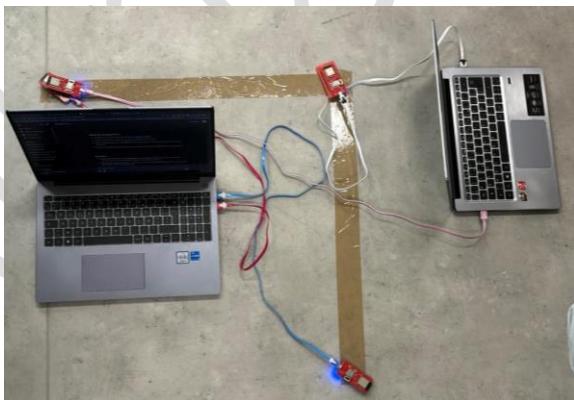


Figure 74 : Essai de repérage au sol rapproché

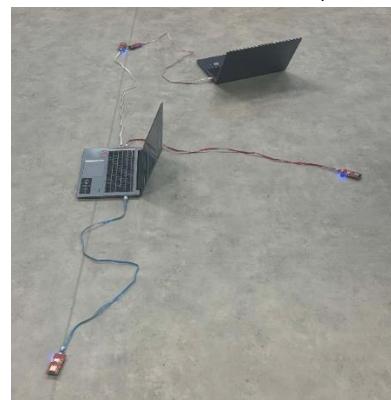


Figure 75 : Essai de repérage au sol éloigné

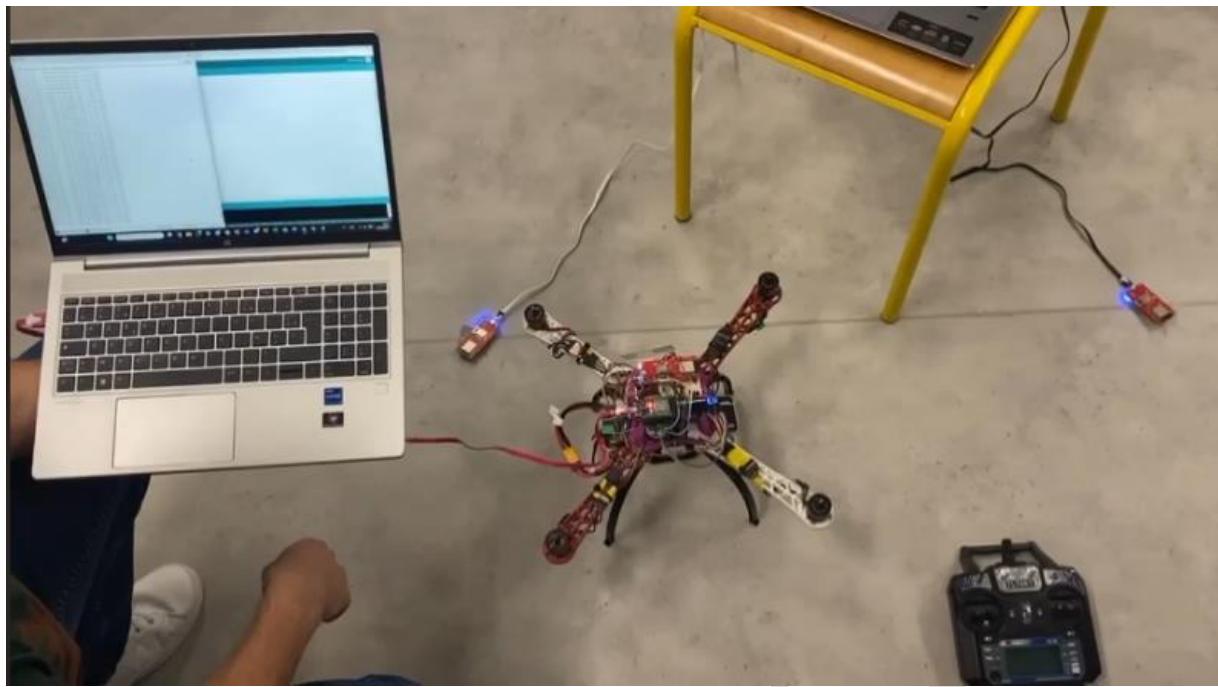


Figure 76 : Essai de repérage dans l'espace avec l'ensemble du système volant

```
    23     {0x84, 0.0, 0.0},  
    24     {0x85, 2.0, 0.0},  
    25     {0x86, 1.0, 1.13}  
};  
27  
28 // Variables globales DW1000  
29 float distances[3] = {0.0, 0.0, 0.0};  
30 float xmes = 0.0, ymes = 0.0;  
31  
32 // PID Gains  
33 const float Kvp = 1.0, Kvi = 4.0, Kvd = 0.001;  
34 const float Kup = 1.0, Kui = 4.0, Kud = 0.001;  
35 const float Kvm = -0.1, Kvi = -0.1, Kvd = -0.001;  
  
Sortie Moniteur série X  
Message (Enter to send message to 'ESP32 Dev Module' on 'COM9')  
Angles calculés : x = 0.020, y = 0.000, z = 0.000  
Angles -> Theta: 0.092, Phi: -0.425  
Position calculée : x = 0.020, y = 0.195  
Position actuelle : (0.020, 0.195, 0.000)  
Angles -> Theta: 0.092, phi: -0.760  
Position actuelle : (0.020, 0.195, 0.000)  
Angles -> Theta: 0.092, Phi: -0.733
```

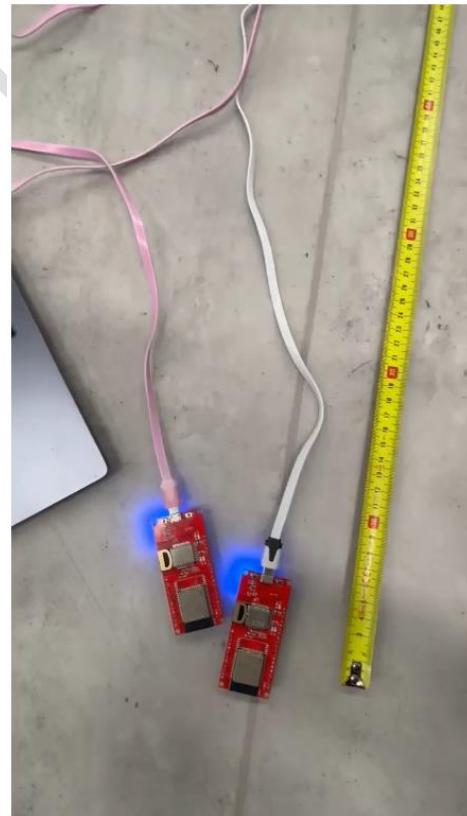


Figure 77 : Acquisition du positionnement et traitement de l'incidence de correction pour une position en (0, 0, 0)

Figure 78 : Essai de repérage dans l'espace avec incidence de correction pour une position en $(0, 0, 0)$



Test affichage et acquisition de la distance à parcourir

L'objectif de cet essai est de vérifier le bon fonctionnement de la réception de la distance à parcourir via le clavier intégrer et de vérifier le bon affichage sur l'écran OLED utilisé.

Exigences et Critères de Réussite

Les exigences techniques que doit respecter le système sont les suivantes :

- Le clavier doit permettre d'acquérir la distance de manière correcte et en fonctionnant correctement (EOT1)
- L'affichage doit se dérouler correctement en fonction des différentes touches pressées sur le clavier (EOT5)

Les critères de performance permettent d'évaluer la qualité de la mise en œuvre. La cohérence des données reçues est primordiale. Les touches du clavier ne doivent pas être mélangés et les données rentrées doivent être correctement interprétées et affichées sur l'écran dédié.

Composants Testés

Différents composants essentiels au bon déroulement de cet essai seront testés :

- L'AZDelivery Keypad 4x4, qui permet de rentrer les données de la distance souhaitée
- L'écran OLED 0,96 pouces, qui permet d'afficher les informations saisies

Méthode d'Essai

La méthodologie adoptée pour cet essai est très simple et consiste simplement à appuyer sur différents chiffres et de constater leur affichage sur l'écran. Ensuite, on testera également les boutons de validation de la sélection afin d'enregistrer la valeur saisie et d'effacement de la saisie, représentés respectivement par les boutons « D » et « A » sur le Keypad.

Cette méthode est justifiée par la raison suivante. Le test de l'acquisition et de la bonne interprétation de la distance est indispensable afin de s'assurer que le système complet et assemblé n'exécute pas l'ordre souhaité.

Conditions de l'Essai

L'environnement de test doit répondre à des conditions spécifiques. L'essai sera réalisé dans une salle fermée et au repos, à une température ambiante de 20°C et avec une humidité relative de 50 %. Aucune interférence radio ou perturbation ne devra être présente.

Les paramètres variables au cours de l'essai sont simplement les différentes touches du Keypad qui peuvent être saisies.

Procédure de test

La procédure de test se déroule en plusieurs phases :

1. Initialisation et mise sous tension
2. Test d'acquisition
 - Saisie de différents chiffres
 - Saisie de la touche « A » afin d'effacer la sélection
 - Saisie d'aucun chiffre
 - Saisie de la touche « D » afin de vérifier que rien ne se passe
 - Saisie d'autres chiffres
 - Saisie de la touche « D » afin de vérifier le bon enregistrement des données. Un engrenage devrait alors apparaître

Collecte des données

Les données recueillies au cours de cet essai seront uniquement visuelles, puisqu'il ne s'agit là que d'une simple interface homme-machine

Validation des résultats

Les résultats de cet essai seront considérés comme validés si les données sont correctement interprétées en fonction des différents touches pressées.

Conclusion

Après avoir effectué les différents essais, notre interprétation des résultats montre que tous les critères de performance ont été respectés. En effet, les touches sont correctement interprétées et lorsque les conditions sont correctement réunies, la saisie est bien enregistrée.

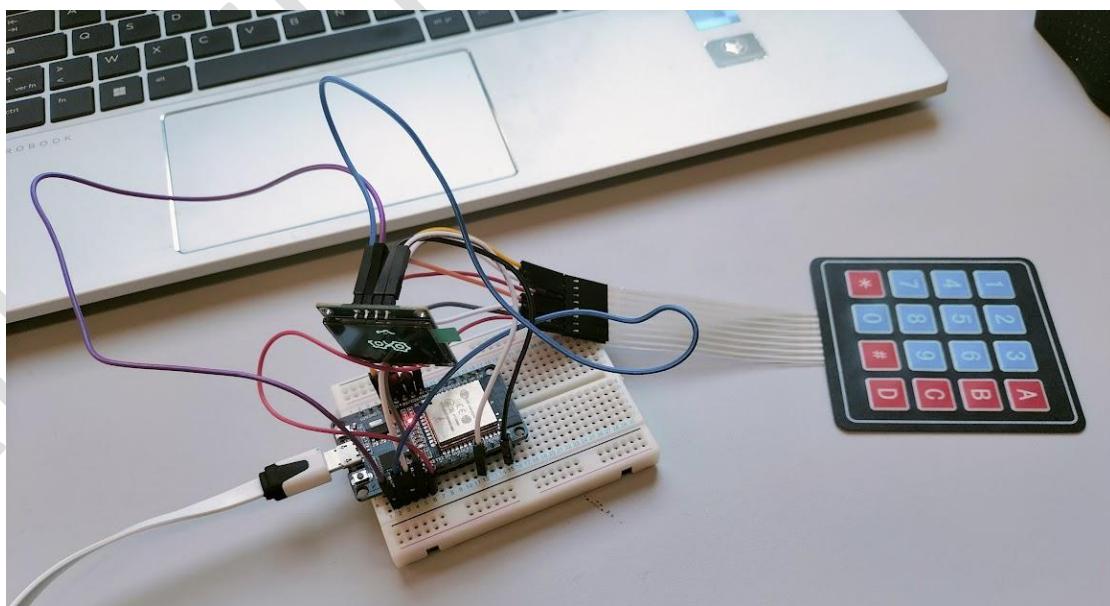


Figure 79 : Montage du test avec l'apparition des engrenages pour montrer l'enregistrement de la distance saisie



Tests fonctionnels

Test de commutation et de réception des signaux

Objectif d'Essai

L'objectif de cet essai est de vérifier le bon fonctionnement de système de commutation et de la bonne réception et émission des signaux. Ceux-ci permettront de valider si le drone répond bien aux bonnes commandes aux bons moments.

Les validations attendues sont :

- Réception et transmission des signaux entre la radiocommande et le contrôleur de vol avec 6 channels en entrée
- Commutation entre le mode manuel et le mode automatique avec un retour visuel pour l'opérateur
- Armement et désarmement du drone avec un retour sonore et une réponse des moteurs cohérente

Les hypothèses considérées sont :

- La carte permettant le pilotage automatique n'est pas installée sur le système volant
- L'environnement n'est pas altéré par des interférences

Exigences et Critères de Réussite

Les exigences techniques que doit respecter le système sont les suivantes :

- Le système doit commuter correctement entre le signal de commande manuel et le signal de commande automatique selon le mode de pilotage choisi par l'opérateur (EOT4)
- Il doit être capable de réceptionner et d'interpréter avec précision les signaux émis par la radiocommande pour exécuter les actions correspondantes (EOT12)

Les critères de performance permettent d'évaluer la qualité de la mise en œuvre. La cohérence des commandes est primordiale et seules les commandes associées au mode actif doivent être acceptées. Le retour utilisateur doit être immédiat, qu'il soit visuel ou sonore. La robustesse du système est également un critère essentiel. En effet, la commutation entre les modes doit pouvoir être réalisée plusieurs fois sans défaillance. Enfin, la sécurité exige que la commutation vers le mode manuel reste possible à tout moment, et la fiabilité implique que les commandes soient correctement attribuées aux actions attendues.

Composants Testés

Différents composants essentiels au bon déroulement de cet essai seront testés :

- Le contrôleur de vol, qui transmet les consignes aux moteurs
- Le commutateur, qui permet de basculer entre le mode manuel et le mode automatique
- L'encodeur, qui a pour rôle de convertir les signaux de la radiocommande sous forme PWM en un signal PPM compréhensible par le contrôleur de vol
- Le récepteur radio, qui reçoit les consignes envoyées par la radiocommande



- La radiocommande, qui permet à l'opérateur de transmettre ses consignes
- Le haut-parleur intégré, qui avertit l'opérateur lors de l'activation ou de la désactivation de l'armement

Méthode d'Essai

La méthodologie adoptée pour cet essai repose sur plusieurs étapes. Le système volant sera d'abord alimenté de manière autonome, sans connexion au sol. Ensuite, les hélices seront retirées afin de garantir la sécurité des opérateurs et du matériel. Des ordres de mouvement seront ensuite envoyés via la radiocommande pour valider la réception des consignes et la réponse des moteurs. Enfin, la synchronisation entre les signaux lumineux et sonores et les actions correspondantes sera observée.

Cette méthode est justifiée par plusieurs raisons. En effet, tester le système sans hélices permet de se concentrer sur la fiabilité de la réception et du traitement des signaux sans risque de dommage. De plus, simuler manuellement les scénarios permet de vérifier la cohérence entre les signaux reçus et les actions réalisées.

Conditions de l'Essai

L'environnement de test doit répondre à des conditions spécifiques. L'essai sera réalisé dans une salle fermée et au repos, à une température ambiante de 20°C et avec une humidité relative de 50 %. Aucune interférence radio ou perturbation ne devra être présente.

Les paramètres variables au cours de l'essai sont les suivants : le mode de vol, qui pourra être manuel ou automatique, le mode engagé, qui pourra être armé ou désarmé, et les différentes actions ordonnées, correspondant aux mouvements du drone.

Procédure de test

La procédure de test se déroule en plusieurs phases :

1. Initialisation
 - Connecter tous les composants du système volant entre eux
 - Mettre la radiocommande et la base volante sous tension
2. Test de commutation manuel/automatique
 - Basculer en mode automatique via la radiocommande, observer le changement d'état de la LED sur le commutateur et vérifier que celle-ci soit bleue
 - Basculer à nouveau en mode manuel et vérifier que la LED allumée soit rouge
 - Répéter ce processus à plusieurs reprises afin de valider sa fiabilité
3. Test d'Armement/Désarmement
 - Activer l'armement via la radiocommande, vérifier que le signal sonore se fasse bien entendre et tester que les moteurs répondent bien aux consignes envoyées par l'opérateur
 - Vérifier que les moteurs ne s'activent pas lorsque le drone est désarmé
 - Répéter ce processus à plusieurs reprises afin de valider sa fiabilité



Collecte des données

Les données recueillies au cours de cet essai seront de deux types. D'une part, les observations qualitatives des signaux lumineux et sonores permettront de vérifier leur synchronisation avec les actions réalisées. D'autre part, les observations qualitatives de la réponse des moteurs permettront de comparer les réponses aux consignes envoyées via la radiocommande. Enfin, une vérification des six channels de la radiocommande sera effectuée pour s'assurer qu'ils sont correctement reçus et interprétés par le contrôleur de vol.

Validation des résultats

Les résultats de cet essai seront considérés comme validés si plusieurs critères sont remplis :

- LED du commutateur synchronisée avec le mode désiré
- Signal sonore synchronisé avec l'armement
- Réponse correcte des moteurs en fonction des actions désirées
- L'ensemble des commandes de la radiocommande sont validés

Conclusion

Après avoir effectué les différents essais, notre interprétation des résultats montre que tous les critères de performance ont été respectés. Les transitions entre les modes manuel et automatique se sont déroulées de manière rapide et fiable. Les indicateurs visuels et sonores ont été cohérents avec les actions réalisées, et la communication avec la radiocommande a été stable, sans latence notable.

Après avoir effectué les différents essais, notre interprétation des résultats montre que tous les critères de performance ont été respectés. Les positions étaient cohérentes avec l'ensemble des déplacements, le rafraîchissement était rapide et sans valeurs aberrantes.



Figure 80 : Essai d'armement et de désarmement

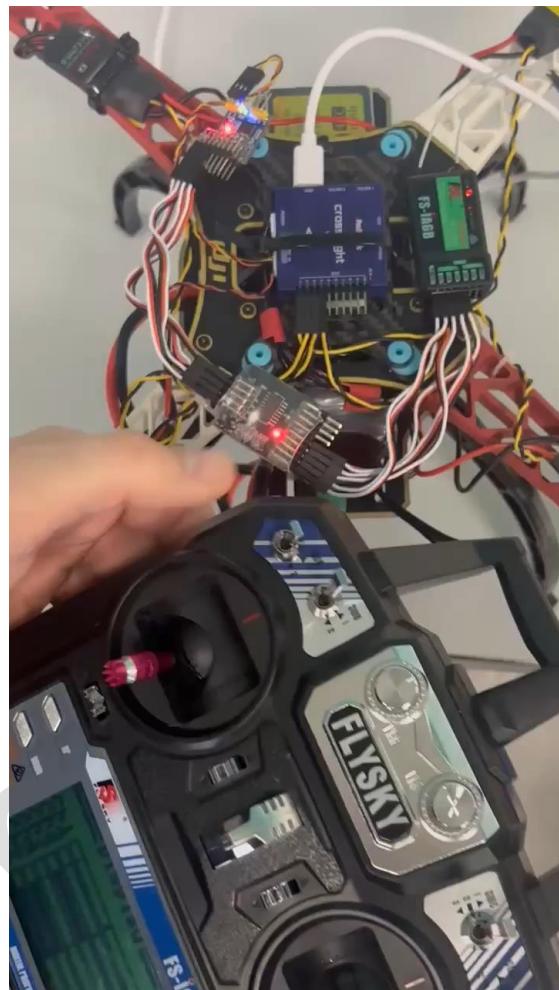


Figure 81 : Essai de commutation



Figure 82 : Essai de compréhension des commandes de l'opérateur par le contrôleur de vol



Test d'autonomie et de vitesse de stabilisation en mode manuel

Objectif d'Essai

L'objectif de cet essai est double. Il s'agit de tester de manière opérationnelle l'autonomie de la batterie et également de déterminer les commandes de consigne de vitesse de rotation des moteurs pour les différentes phases de vol, que ce soit au décollage ou en stabilisation. Cela nous servira pour implémenter des plages de valeurs cohérentes dans l'asservissement automatique en fonction de l'état de la batterie.

Les validations attendues sont :

- Autonomie de la batterie cohérente par rapport aux calculs théoriques
- Extraction des valeurs de stabilisation aux différentes phases de vol via une interface Web

L'hypothèse considérée sera que l'environnement n'est pas altéré par des interférences quelconques.

Exigences et Critères de Réussite

L'exigence technique que doit respecter le système est la suivante : le système stocke l'énergie électrique afin de pouvoir réaliser des vols sans fils et en autonomie énergétique (EOT9).

Les critères de performance permettent d'évaluer la qualité de la mise en œuvre. La cohérence des tensions au fur et à mesure des tests est primordiale et l'allure global de l'état de charge de la batterie doit pouvoir être semblable à celle du batterie LiPo classique. Enfin, le retour d'information de la commande de poussée doit être immédiat et cohérent avec les observations visuelles faites du drone en mouvement. Ceci étant fait via une interface Web appelé « WebSerial » qui permettra de récupérer des informations avec le drone en mouvement sans avoir à le câbler, ce qui poserait d'importants problèmes de sécurité.

Composants Testés

Différents composants essentiels au bon déroulement de cet essai seront testés :

- La batterie, qui alimente l'ensemble du système volant
- L'ESP32-WROOM-32, qui transmettra les informations reçues sur l'interface Web

Méthode d'Essai

La méthodologie adoptée pour cet essai repose sur plusieurs étapes. Le système volant effectuera différents vols consécutifs en entamant le test avec la batterie chargée à son plein potentiel. La tension de la batterie sera mesurée avant et après chaque vol, durant lesquels les valeurs de commande de poussée permettant au drone de se stabiliser et de décoller seront récupérées.

Cette méthode est justifiée par le fait en condition réel du système. Cette étape est indispensable afin de valider la conception faite jusqu'à présent et d'être en mesure d'approcher les tests en mode automatique avec le moins d'inconnus possibles



Conditions de l'Essai

L'environnement de test doit répondre à des conditions spécifiques. L'essai sera réalisé dans une salle fermée et au repos, à une température ambiante de 20°C et avec une humidité relative de 50 %. Aucune interférence radio ou perturbation ne devra être présente.

Les paramètres variables au cours de l'essai sont les suivants : la tension en sortie de la batterie ainsi que la valeur de la consigne de poussée, qui variera en fonction en de l'état de charge de la batterie.

Procédure de test

La procédure de test se déroule en plusieurs phases :

1. Mesure de la tension de sortie de la batterie
2. Initialisation
 - Connecter tous les composants du système volant entre eux
 - Mettre la radiocommande et la base volante sous tension
3. Vol et observation des valeurs de poussée sur l'interface Web
4. Mesure de la tension de sortie de la batterie
4. Répétition du processus jusqu'à épuisement de la batterie

Collecte des données

Les données recueillies seront directement retransmises sur l'interface Web « WebSerial », qui offre la possibilité, via une connexion WiFi, d'avoir un accès au moniteur série en ligne de la carte de programmation considérée, ici l'ESP32-WROOM-32.

Validation des résultats

Les résultats de cet essai seront considérés comme validés si plusieurs critères sont remplis :

- La décharge de la batterie se produit de manière attendue et cohérente
- Les valeurs de consigne de poussée sont cohérentes avec les observations visuelles et l'état de charge de la batterie
- Les valeurs mesurées s'affichent correctement sur l'interface Web

Conclusion

Au final, 13 vols consécutifs ont été fait durant ce test et notre interprétation des résultats montre que tous les critères de performance ont été respectés. La décharge de la batterie se fait correctement, les valeurs s'affichent correctement sur l'interface Web et sont cohérentes avec les observations visuelles du drone. On peut alors en extraire les courbes suivantes qui nous seront nécessaires afin de déterminer les plages des valeurs de poussée à prendre en compte ainsi que pour déterminer l'autonomie réelle de la batterie.

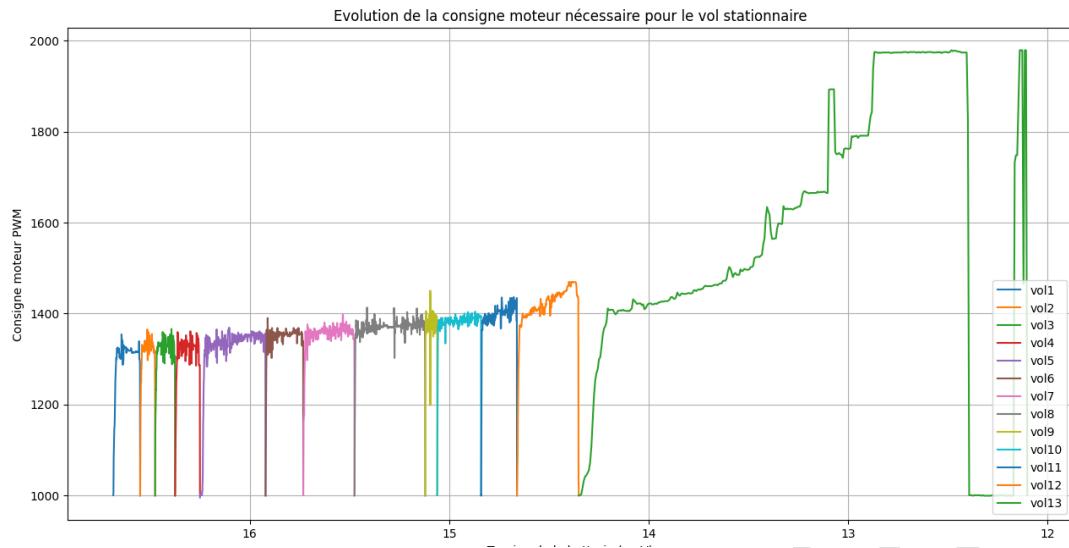


Figure 83 : Visualisation de la consigne de poussée requise pour une stabilisation en fonction de la décharge de la batterie

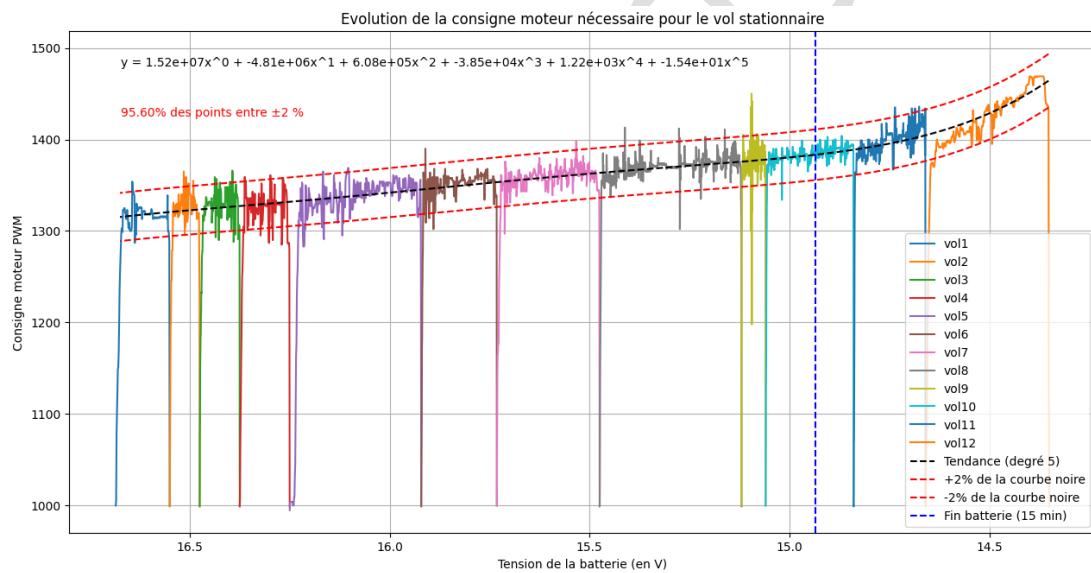


Figure 84 : Détermination de l'autonomie de la batterie et des plages de poussée requises

Les résultats obtenus sur les figures précédentes permettent de mettre en évidence une courbe de décharge qui correspond bien à celle d'une batterie LiPo avec une consigne de poussée qui augmente naturellement au fur et à mesure de la décharge de la batterie. En effet, de plus en plus de poussée est nécessaire pour maintenir le drone à altitude constante. On peut alors déterminer le moment où la courbe devient exponentielle et n'est plus linéaire, marquant le début de la zone critique dans laquelle on se refusera d'aller pour des questions de sécurité. Elle se définit ainsi à exactement 15min, correspondant donc bien aux estimations théoriques faites auparavant. De plus, on peut définir, sur cette d'autonomie, une plage de valeur pour la consigne de poussée comprise entre 1300 et 1400, permettant d'englober plus de 95% des points mesurés. Ainsi le mode automatique pourra être brider et ne pourra jamais excéder une valeur de consigne supérieure à 1400, pour des questions de sécurité.

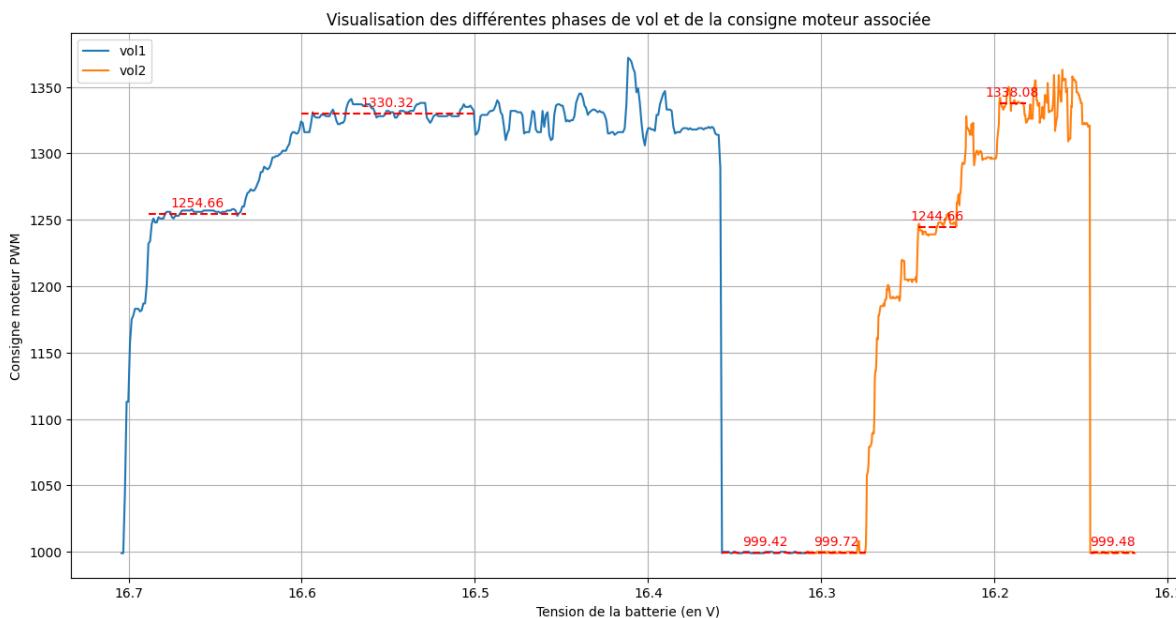


Figure 85 : Visualisation des paliers de stabilisation à pleine charge

De plus, les résultats permettent également de visualiser les différents paliers de stabilisation du drone lorsque la batterie est à son plein potentiel. On constate que la consigne nécessaire pour permettre au drone de tout juste compenser son propre poids et d'être à la limite du décollage se trouve aux alentours de 1250 et que la consigne permettant une stabilisation en l'air est aux alentours de 1330 avec évidemment, une tendance à augmenter au fur et à mesure de la consommation de la charge de la batterie. Ces données seront essentielles afin que le pilote automatique puisse exécuter les phases de stabilisation, de décollage et d'atterrissement de la meilleure manière possible.

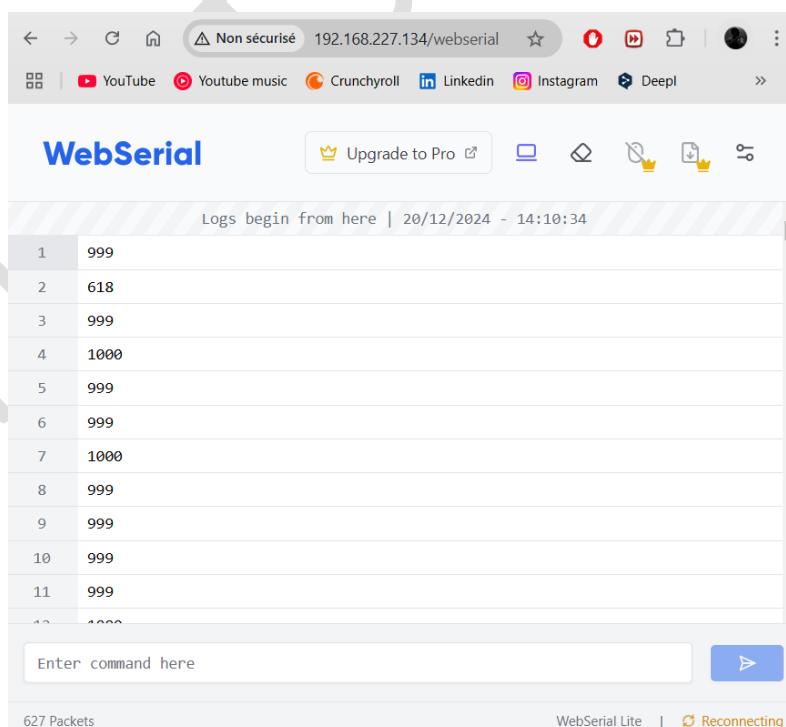


Figure 86 : Visualisation de l'interface Web utilisée



Tests opérationnels

Test d'une séquence de vol complète sans les hélices

Objectif d'Essai

L'objectif de l'essai est de tester le fonctionnement du module automatique pour une séquence de vol sans déplacement horizontal (montée, stabilisation, descente) sans hélices. Cela permettra d'observer facilement l'ensemble des données issus du pilote automatique, avec notamment l'acquisition de la position et les consignes générées en conséquence, en étant directement branché à celui-ci via le port USB de l'ordinateur.

Les validations attendues sont :

- Acquisition de la position cohérente par rapport à la position réel
- Consignes moteurs cohérentes par rapport au comportement attendu
- Mise en défaut automatique du système à la fin de la séquence de vol

Exigences et Critères de Réussite

Les exigences techniques que doit respecter le système sont les suivantes :

- Le système doit acquérir sa position avec précision (EOT1)
- Le système doit traiter l'information pour générer des consignes de vitesses de rotation des moteurs cohérentes (EOT2)
- Le système doit pouvoir correctement afficher des données cohérentes par rapport à ses tâches en cours de traitement (EOT5)
- La communication entre la radiocommande et le système volant est fiable, permanente, cohérente et les informations envoyées par la radiocommande sont correctement interprétées (EOT11 & EOT12)

Les critères de performance permettent d'évaluer la qualité de la mise en œuvre. La qualité de l'acquisition est primordiale, les données doivent être précise et il ne doit pas y avoir de valeur aberrante entre deux positions, ce qui donnerait de mauvaises valeurs de consigne aux moteurs. De plus, les consignes doivent permettre aux moteurs de générer la poussée nécessaire pour diriger le drone dans la direction adaptée.



Composants Testés

Différents composants essentiels au bon déroulement de cet essai seront testés :

- Le contrôleur de vol, qui transmet les consignes aux moteurs
- Les ESP32 UWB DW1000, dont trois qui formeront les balises au sol et une sur le drone assurant l'acquisition de la position
- Le commutateur, qui permet de basculer entre le mode manuel et le mode automatique
- L'encodeur, qui a pour rôle de convertir les signaux de l'ESP32-WROOM-32 sous forme PWM en un signal PPM compréhensible par le contrôleur de vol
- L'ESP32-WROOM-32, qui permet de traiter l'acquisition afin de générer des consignes de vol
- Le haut-parleur intégré, qui avertit l'opérateur lors de l'activation ou de la désactivation de l'armement
- L'écran OLED, qui affiche les différentes informations relatives aux tâches en cours de traitement par le système

Méthode d'Essai

La méthodologie adoptée pour cet essai repose sur plusieurs étapes. Le système volant sera d'abord alimenté de manière autonome, sans connexion au sol. Ensuite, les hélices seront retirées afin de garantir la sécurité des opérateurs et du matériel. L'ESP32-WROOM-32 sera relié à un ordinateur afin d'observer l'évolution des commandes envoyées vers le contrôleur de vol sur le moniteur série. Les trois balises ESP32 UWB DW1000, seront placées au sol et configurées. Une séquence de vol sera ensuite définie puis initié par l'opérateur. Il déplacera alors le drone manuellement selon la trajectoire et les étapes du vol. L'évolution des consignes de l'ESP32 sera alors observée sur l'ordinateur ainsi que la réponse des moteurs. Enfin, la mise en défaut du système une fois la séquence de vol terminé, sera observé.

Cette méthode est justifiée par plusieurs raisons. En effet, tester le système sans hélices permet de se concentrer sur la fiabilité de la réception et du traitement des signaux sans risque de dommage. De plus, simuler manuellement les scénarios permet de vérifier la cohérence entre les signaux générés et les actions réalisées.

Conditions de l'Essai

L'environnement de test doit répondre à des conditions spécifiques. L'essai sera réalisé dans une salle fermée et au repos, à une température ambiante de 20°C et avec une humidité relative de 50 %.

Les paramètres variables au cours de l'essai sont les suivants : la position du drone ainsi que les consignes selon les mouvements du drone.



Procédure de test

La procédure de test se déroule en plusieurs phases :

1. Initialisation

- Connecter tous les composants du système volant entre eux
- Mettre la base volante et les balises au sol sous tension

2. Test de vol

- Initialiser la séquence de vol
- Déplacer progressivement la base volante verticalement, observer les consignes de vol et le comportement des moteurs
- Positionner la base volante au point de stabilisation définie, observer le comportement des consignes, qui doivent permettre de maintenir le drone à ce point fixe pendant 10 secondes
- Descendre progressivement la base volante jusqu'au sol, observer l'évolution des consignes et la mise en défaut automatique

Collecte des données

Les données recueillies au cours de cet essai seront les suivantes. D'une part, les consignes de vol affichées sur le moniteur série de l'ordinateur permettant d'observer l'augmentation des consignes de poussées lors de l'ascension, leur stabilisation au point le plus élevé de la trajectoire et finalement leur diminution durant la descente. D'autre part, l'écoute qualitatives de la réponse des moteurs permettra de valider ou non les réponses aux consignes. Enfin, l'observation du signal lumineux permettra de valider la mise en défaut.

Validation des résultats

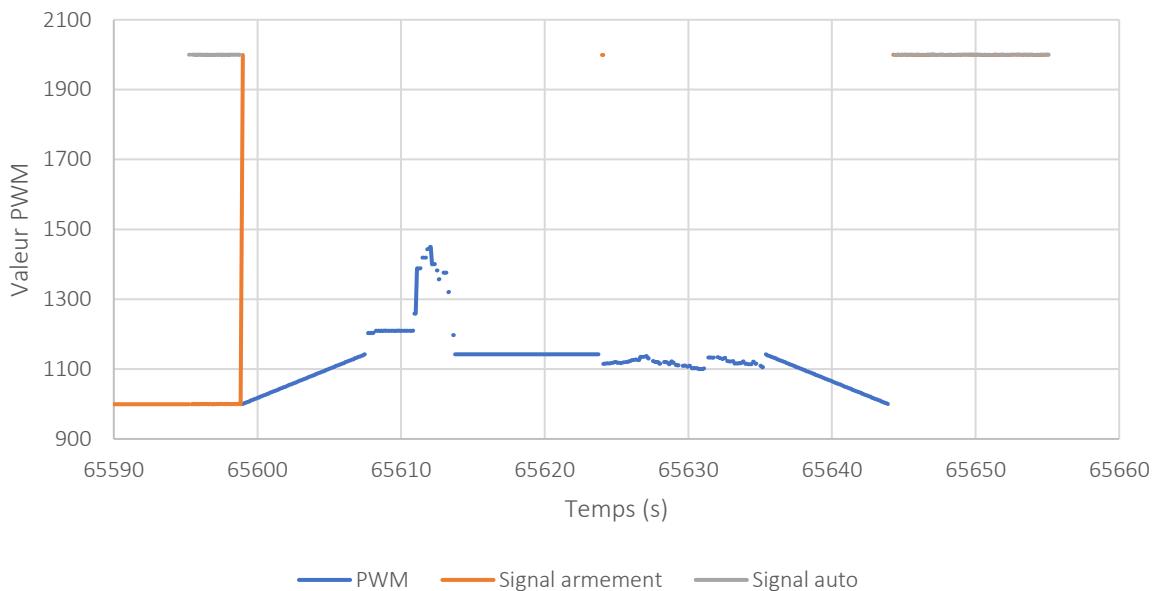
Les résultats de cet essai seront considérés comme validés si plusieurs critères sont remplis :

- Les consignes sont en adéquation avec la position du drone et la phase de vol requise
- La vitesse de rotation des moteurs évolue conformément aux consignes
- Le drone se met bien en défaut une fois la séquence de vol terminée

Conclusion

Après avoir effectué l'essai, notre interprétation des résultats montre que tous les critères de performance ont été correctement respectés. Les consignes envoyées par L'ESP32-WROOM-32 étaient cohérentes et fiables. En effet, les différentes phases de vol sont bien respectées et les consignes envoyées aux moteurs sont cohérentes comme le montre la figure suivante :

Résultats vol test sans hélices 16/12/2024



C'est uniquement au moment où le signal d'armement s'enclenche (valeur à 2000) que les moteurs commencent à tourner. La séquence qui s'en suit est une succession de différentes phases de vol avec tout d'abord la phase de démarrage linéaire des moteurs jusqu'à atteindre la vitesse qualifiée de neutre, correspondant à celle qui permet de tout juste compenser le poids du drone. Viens ensuite la phase de montée avec une accélération jusqu'au maximum de vitesse autorisé avec un ralentissement amenant le drone sur sa phase de stabilisation durant laquelle la vitesse reste constante de telle sorte à compenser le poids du système volant. Enfin arrive les dernières phases, qui sont la descente avec un plafond minimal qui est bien respecté et un arrêt progressif et linéaire des moteurs une fois le sol atteint.

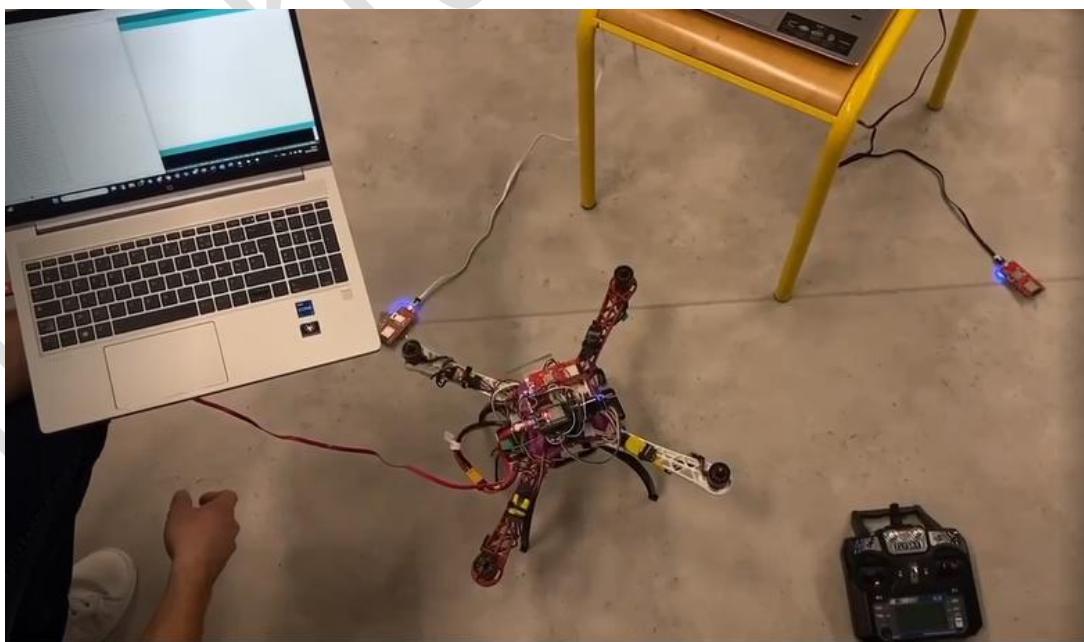


Figure 87 : Dispositif de test avec le drone au sol durant la préparation au vol

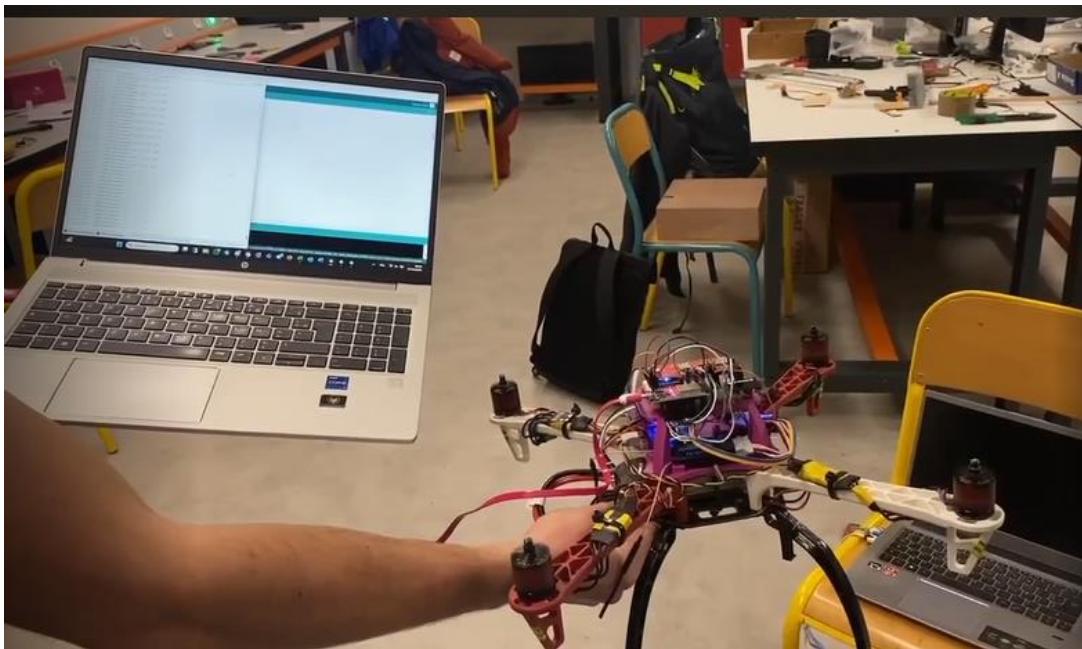


Figure 88 : Dispositif de test avec le déplacement du drone durant la séquence de vol



Test d'une séquence de vol complète avec les hélices semi-automatique

Objectif d'Essai

L'objectif de l'essai est de tester le fonctionnement du module automatique pour une séquence de vol complète sans déplacement horizontal (montée, stabilisation, descente) avec hélices. La particularité de ce test est que seule la commande de poussée sera automatisée tandis que l'ensemble des autres commandes de tangage, roulis et lacet seront laissés à la charge de l'opérateur.

Les validations attendues sont :

- Respect de plan de vol
- Consignes moteurs cohérentes par rapport au comportement attendu

Exigences et Critères de Réussite

Les exigences techniques que doit respecter le système sont les suivantes :

- Le système doit acquérir sa position avec précision (EOT1)
- Le système doit traiter l'information pour générer des consignes de vitesses de rotation des moteurs cohérentes (EOT2)
- Le système doit pouvoir correctement afficher des données cohérentes par rapport à ses tâches en cours de traitement (EOT5)
- La communication entre la radiocommande et le système volant est fiable, permanente, cohérente et les informations envoyées par la radiocommande sont correctement interprétées (EOT11 & EOT12)

Les critères de performance permettent d'évaluer la qualité de la mise en œuvre. La qualité de l'acquisition est primordiale, ce qui permet le respect du plan de vol définit.



Composants Testés

Différents composants essentiels au bon déroulement de cet essai seront testés :

- Le contrôleur de vol, qui transmet les consignes aux moteurs
- Les ESP32 UWB DW1000, dont trois qui formeront les balises au sol et une sur le drone assurant l'acquisition de la position
- Le commutateur, qui permet de basculer entre le mode manuel et le mode automatique
- L'encodeur, qui a pour rôle de convertir les signaux de l'ESP32-WROOM-32 sous forme PWM en un signal PPM compréhensible par le contrôleur de vol
- L'ESP32-WROOM-32, qui permet de traiter l'acquisition afin de générer des consignes de vol
- Le haut-parleur intégré, qui avertit l'opérateur lors de l'activation ou de la désactivation de l'armement
- L'écran OLED, qui affiche les différentes informations relatives aux tâches en cours de traitement par le système

Méthode d'Essai

La méthodologie adoptée pour cet essai repose sur plusieurs étapes. Le système volant sera d'abord alimenté de manière autonome, sans connexion au sol. Ensuite, les hélices seront placées sur le système volant afin de garantir la sécurité des opérateurs et du matériel. L'ESP32-WROOM-32 transmettra via WiFi à notre ordinateur les données essentielles de positionnement afin d'observer si l'évolution du système est correcte au cours du vol. Les trois balises ESP32 UWB DW1000, seront placées au sol et configurées. Une séquence de vol sera ensuite définie puis initié par l'opérateur.

Cette méthode est justifiée par plusieurs raisons. En effet, tester le système avec les hélices est indispensable pour la phase finale de test du système global. Cependant, pour une question de sécurité, il est préférable d'uniquement laissé l'automatisation sur la poussée dans un premier temps afin de s'assurer que cela fonctionne correctement.

Conditions de l'Essai

L'environnement de test doit répondre à des conditions spécifiques. L'essai sera réalisé dans une salle fermée et au repos, à une température ambiante de 20°C et avec une humidité relative de 50 %.



Procédure de test

La procédure de test se déroule en plusieurs phases :

1. Initialisation
 - Connecter tous les composants du système volant entre eux
 - Fixer les hélices sur les moteurs
 - Mettre la base volante et les balises au sol sous tension
2. Test de vol
 - Activer le mode automatique
 - Initialiser la séquence de vol en entrant une distance de 0m dans le calculateur
 - Activer l'armement
 - Vérifier l'ensemble des paramètres et suivre la progression du système

Collecte des données

Les données recueillies au cours de cet essai seront les suivantes. D'une part, les coordonnées de vol seront stockées sur l'ordinateur afin d'en faire un post-traitement. D'autre part, l'approbation global de ce test se fera sur les observations visuelles du respect de la trajectoire.

Validation des résultats

Les résultats de cet essai seront considérés comme validés si le drone effectue correctement sa séquence de vol.

Conclusion

Après avoir effectué l'essai, notre interprétation des résultats montre que tous les critères de performance ont été correctement respectés. La séquence de vol a bien été respecté malgré le faible niveau de batterie qui a empêché le système de décoller davantage.



Figure 89 : Photo du drone en vol semi-automatique

Mode d'emploi

Mentions dangereuses

- Toujours opérer le drone dans un espace dégagé, intérieur et sans personnes ou obstacles à proximité
- Ne jamais approcher le drone lorsque les moteurs sont en marche
- En cas de perte de contrôle ou de situation dangereuse, couper immédiatement l'armement via le switch dédié (cf. section [Procédure à respecter en cas de problème](#))
- Ne jamais utiliser le drone sous l'effet de substances altérant la vigilance
- Toujours vérifier l'état de la batterie qui doit être à pleine charge et des composants avant chaque vol
- Ne pas tenter de réparer le drone lorsqu'il est sous tension

Procédure à respecter pour le vol manuel

- Disposer les balises au sol dans la pièce conformément au plan de disposition fournit (cf. section [Disposition au sol des balises](#)). Attention à bien respecter l'ordre de mise en place des balises. Une fois placé, les mettre sous tension à l'aide des batteries externes
- Allumer la radiocommande et bien faire attention à mettre tous les switch et joysticks en position basse ou nulle



Figure 90 : Actionner l'armement sur la radiocommande

- Mettre le drone sous tension en branchant la batterie
- Attendre quelques secondes le temps que le système volant s'allume et qu'il n'y ait plus de signal sonore et que la mention : « Bonjour » apparaissent sur l'écran supérieur
- Eloignez-vous à bonne distance du drone afin d'éviter tout contact physique avec celui-ci
- Une fois éloigné, activé l'armement avec le switch dédié de la télécommande. Un signal sonore retentit et les moteurs se mettent alors en marche. Il ne reste plus qu'à vous faire plaisir
- Une fois le vol terminé, désactiver l'armement à l'aide du switch dédié de la radiocommande. Un signal sonore s'activera alors
- Veuillez le débrancher physiquement le système volant de son alimentation en débranchant la batterie
- Recommencez ensuite l'ensemble de la procédure autant de fois que vous le voulez aussi longtemps que le niveau de batterie le permet

Procédure à respecter pour le vol automatique

- Disposer les balises aux sols dans la pièce conformément au plan de disposition fournit en veillant à bien respecter la numération des balises indiquée. Attention à bien respecter l'ordre de mise en place des balises. Une fois placé, les mettre sous tension à l'aide des batteries externes
- Allumer la radiocommande et bien faire attention à mettre tous les switch et joysticks en position basse ou nulle
- Mettre le drone sous tension en branchant la batterie
- Attendre quelques secondes le temps que le système volant s'allume et qu'il n'y ait plus de signal sonore et que la mention : « Bonjour ! Prêt à embarquer ?» apparaisse sur l'écran supérieur
- Activer le mode automatique à l'aide du switch dédié de la radiocommande. La mention « Veuillez indiquer la distance à parcourir... » apparaît alors sur l'écran



Figure 91 : Actionner le mode automatique sur la radiocommande



- Indiquer, en mètres, la distance à franchir via le Keypad. Une fois la distance saisie à l'aide des chiffres, appuyer sur la touche « D » du Keypad. En cas d'erreur, presser la touche « A » pour effacer la saisie et recommencer. A noter que ce système prend en compte uniquement des chiffres entiers et aucune virgule. De plus, si la distance saisie est de 0, le drone fera uniquement un décollage, une stabilisation puis une descente.
- Une fois la touche « D » saisie, veuillez attendre quelques secondes que le système embarqué se prépare au décollage.
- Une fois prêt, la mention : « Prêt au décollage Cap'tain » apparait. Eloignez-vous à bonne distance du drone afin d'éviter tout contact physique avec celui-ci
- Une fois éloigné, activé l'armement avec le switch dédié de la télécommande. Un signal sonore retentit et les moteurs se mettent alors en marche. Il ne reste plus qu'à profiter du spectacle
- Une fois la séquence de vol terminé, le drone se mettra automatique en mode par défaut. Veuillez le débrancher physiquement de son alimentation en débranchant la batterie
- Recommencez ensuite l'ensemble de la procédure autant de fois que vous le voulez et aussi longtemps que le niveau de batterie le permet

Procédure à respecter en cas de problème

Durant le vol automatique

- Si une anomalie visible est détectée durant le vol, repassez immédiatement en mode manuel à l'aide du switch dédié de la radiocommande. Le drone passera alors dans son mode dégradé et sera désormais pilotable uniquement manuellement sans ne plus avoir la possibilité de basculer en mode automatique avant la mise hors tension complète du système par le débranchement de la batterie
- En cas d'incapacité à reprendre le contrôle du drone, désarmez immédiatement le drone. Les moteurs se couperont instantanément et le système volant s'écrasera par terre sans faire de dégât corporel ou matériel autre que lui-même

Durant le vol manuel

En cas d'incapacité à garder le contrôle du drone, désarmez immédiatement le drone. Les moteurs se couperont instantanément et le système volant s'écrasera par terre sans faire de dégât corporel ou matériel autre que lui-même

Entretien et stockage

- Batterie : Toujours débrancher les batteries après usage et les stocker dans un endroit sec et à température ambiante.
- Moteurs : Inspecter régulièrement les moteurs pour détecter toute usure ou obstruction.
- Hélices : Vérifier l'état des hélices avant chaque vol. Remplacer les hélices endommagées.
- Châssis : Contrôler l'intégrité du châssis après chaque vol.

Assistance et support technique

Pour toute question ou assistance technique, veuillez contacter notre équipe à l'adresse suivante :
marc.beauchet@epf.fr

Disposition au sol des balises

Avant tout utilisation du système automatique, veuillez disposer les balises au sol de la manière suivante :

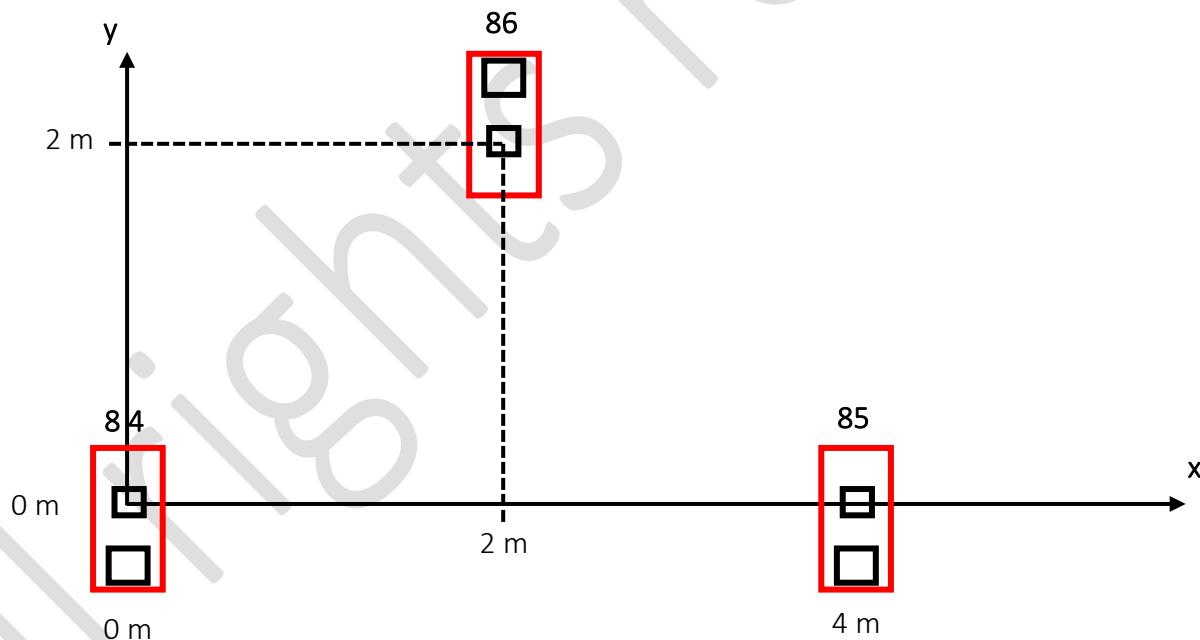


Figure 92 : Schéma de positionnement des balises au sol pour le vol automatique

Conclusion

Pour l'organisation globale du projet drone, nous avons planifié la phase de simulation sur une période de deux mois, de septembre à octobre, suivie de la phase de validation également sur deux mois, de novembre à décembre. Afin de garantir une gestion efficace des tâches et de minimiser les risques de retard, plusieurs équipes ont été mobilisées selon les compétences nécessaires à chaque étape. Pendant les deux premiers mois, trois équipes ont été actives, tandis que les deux derniers mois, deux équipes ont pris le relais pour finaliser le projet comme illustré sur la figure 93.



Figure 93 : Diagramme de Gantt du projet



Annexe

Code ESP32-WROOM-32

```
/// -----  
/// ----- CODE ESP32-WROOM-32 -----  
/// ----- PROJET DRONE GPD1 -----  
/// -----  
  
/// ----- Bibliothèques -----  
  
#include <Keypad.h>  
#include <Adafruit_GFX.h>  
#include <Adafruit_SSD1306.h>  
#include <cmath>  
#include <math.h>  
#include <WiFi.h>  
#include <WebSerial.h>  
#include <ESPAsyncWebServer.h> // Pour AsyncWebServer  
  
/// ----- Définitions -----  
  
// SDA & SCL  
#define SDA_PIN 21  
#define SCL_PIN 22  
  
// Ecran OLED  
#define SCREEN_I2C_ADDR 0x3C  
#define SCREEN_WIDTH 128  
#define SCREEN_HEIGHT 64  
#define OLED_RST_PIN -1  
Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, OLED_RST_PIN);  
#define FRAME_DELAY 42  
#define FRAME_WIDTH 64  
#define FRAME_HEIGHT 64  
#define FRAME_COUNTPos (sizeof(framespos) / sizeof(framespos[0]))  
  
// Moteurs, leds et channels de la radiocommande  
#define PWM_PIN_1 18 // Pin pour ESC 1  
#define PWM_PIN_2 19 // Pin pour ESC 2  
#define PWM_PIN_3 2 // Pin pour ESC 3  
#define PWM_PIN_4 23 // Pin pour ESC 4  
#define PWM_PIN_OUT 15 // Pin pour le mode auto  
#define SEQUENCE_PIN 35 // Pin pour déclencher la séquence  
#define LED_RED 5 // LED rouge  
#define LED_GREEN 4 // LED verte  
  
// Fréquences et résolutions  
#define ESC_PWM_FREQUENCY 50 // Fréquence PWM pour ESC (50 Hz)  
#define LED_PWM_FREQUENCY 50 // Fréquence PWM pour LEDs (50 ou 500 Hz)  
#define LED_PWM_RESOLUTION 255 // Résolution (0-255 pour la largeur d'impulsion)  
#define PWM_PERIOD (1000000 / ESC_PWM_FREQUENCY) // Période en microsecondes pour ESC (20 ms)  
  
// Keypad  
const byte ROWS = 4;  
const byte COLS = 4;
```













```
const float hauteur_du_vol = 0.7;
const float hauteur_au_sol = 0.3;
const float position_x_depart = 2.0;
const float position_y_depart = 0.0;

// Points de passage initiaux pour le parcours imposé
float A_z = hauteur_au_sol; // Position du point A
float A_y = position_y_depart; // Position du point A
float A_x = position_x_depart; // Position du point A
float B_z = hauteur_du_vol; // Position du point B
float B_y = position_y_depart; // Position du point B
float B_x = position_x_depart; // Position du point B
float C_z = hauteur_du_vol; // Position du point C
float C_y = position_y_depart; // Position du point C
float C_x = 0.0; // Position du point C
float D_z = hauteur_au_sol; // Position du point D
float D_y = position_y_depart; // Position du point D
float D_x = 0.0; // Position du point D

// Constantes pour la gestion du vol
const float radius_checkpoint = 0.50; // Rayon des sphères autour de A et B
const float V_min = 1200; // Valeur PWM pour la vitesse max de descente
const float V_neutre = 1240; // Valeur PWM pour la vitesse stabilisée
const float V_max = 1340; // 1400; // Valeur PWM pour la vitesse max de montée
const int duree_stabilisation = 10000; // Durée de stabilisation en micro secondes
float distancecouloirvolx = 0.0;
float distancecouloirvoly = 0.0;
const float V_reelle = 0.1;
const float V_reelle_stab = 0.01;
```



```
// PID vertical
const int Kp = 30;
float erreur = 0.0;

// PID horizontal
float erreurU = 0.0;
float erreurV = 0.0;
float U = 0.0;
float V = 0.0;
int KpU = 1;
int KpV = 1;

// Gestion des etats
enum Phase {
    START,
    PHASE_1,
    STABILIZE_AT_B,
    PHASE_2,
    STABILIZE_AT_C,
    PHASE_3,
    STOP
};

// Autres variables
Phase currentPhase = START; // État initial
bool missionComplete = false; // Drapeau global pour indiquer la fin de la mission
float xmes = 0.0, ymes = 0.0, zmes = 0.0, thetames = 0.0, phimes = 0.0, xmesn = 0.0, ymesn = 0.0, zmesn = 0.0;
bool Autorisation_preparation_decollage = false;
bool Preparation_decollage = false;
float Distance;
bool Ne_pas_executer_la_phase_2 = false;
int valeurRecue = 0; // Variable pour stocker la valeur reçue
int ledRedDutyCycle = 0; // Duty cycle pour la LED rouge
int ledGreenDutyCycle = 0; // Duty cycle pour la LED verte
bool ledState = false; // %tat de la LED verte pour le clignotement
String ligne = "";
int etatSaisie = 0; // 0 = message d'attente, 1 = saisie en cours, 2 = saisie validée
unsigned long startTime; // Variable globale pour stocker le temps de départ
bool stabilisationEnCours = false; // Indicateur d'état de stabilisation
const float rayon_couloir_de_vol = 0.2;
static bool conditionMet = false;
int pwm_output = V_neutre;
float pwm_output_theta_degres = 0.0;
float pwm_output_phi_degres = 0.0;
float pwm_output_theta = 0.0;
float pwm_output_phi = 0.0;
unsigned long previousTime = 0; // en microsecondes par exemple
float dt = 0.0;
float alpha = 0.0;
float zcons = 0.0;

// Vos identifiants WiFi
const char* ssid = "Joachim";
const char* password = "christa2";

// Création d'un serveur WiFi sur le port 80
AsyncWebServer server(80); // Création d'un serveur async sur le port 80

/// ----- Fonctions ----- ///
// Mise à jour de l'écran OLED
void updateDisplay(String message) {
```



```
display.clearDisplay();
display.setTextSize(1);
display.setTextColor(SSD1306_WHITE);
display.setCursor(0, 0);
display.println(message);
display.display();
}

// Smiley écran OLED
void showSmileyFace() {
    unsigned long startTime = millis(); // Temps de départ
    while (millis() - startTime < 5000) { // Boucle pendant 5 secondes
        display.clearDisplay();
        display.drawBitmap(32, 0, framespos[frame], FRAME_WIDTH, FRAME_HEIGHT, 1);
        display.display();
        frame = (frame + 1) % FRAME_COUNTpos;
        delay(FRAME_DELAY);
    }
    display.clearDisplay();
}

// Signal PWM pour la poussée
void generatePWM(int pin, int pulseWidthMicros) {
    digitalWrite(pin, HIGH);
    delayMicroseconds(pulseWidthMicros); // Maintenir à HIGH pendant la largeur d'impulsion
    digitalWrite(pin, LOW);
    delayMicroseconds(PWM_PERIOD - pulseWidthMicros); // Compléter le cycle
}

// Signal PWM pour les angles
void generatePWMMangle(int pin, int angle) {
    const int NEUTRAL_WIDTH_MICROS = 1500; // Valeur neutre à 1500 microsecondes
    const int MIN_WIDTH_MICROS = 1000; // Largeur minimale à 1000 microsecondes
    const int MAX_WIDTH_MICROS = 2000; // Largeur maximale à 2000 microsecondes
    const float MIN_ANGLE = -10.0; // Angle minimum en degrés
    const float MAX_ANGLE = 10.0; // Angle maximum en degrés
    if (abs(angle) > 10) {
        angle = 10;
    }
    int pulseWidthMicros = map((int)(angle * 1000), (int)(MIN_ANGLE*1000), (int)(MAX_ANGLE*1000), MIN_WIDTH_MICROS,
    MAX_WIDTH_MICROS);
    // Limiter la largeur d'impulsion pour éviter les dépassemens de bornes
    pulseWidthMicros = constrain(pulseWidthMicros, MIN_WIDTH_MICROS, MAX_WIDTH_MICROS);
    // Inversion de la consigne pour PWM_PIN_2
    if (pin == PWM_PIN_2) {
        pulseWidthMicros = MIN_WIDTH_MICROS + MAX_WIDTH_MICROS - pulseWidthMicros;
    }
    pulseWidthMicros = (int) ceil(pulseWidthMicros);
    digitalWrite(pin, HIGH);
    delayMicroseconds(pulseWidthMicros); // Maintenir à HIGH pendant la largeur d'impulsion
    digitalWrite(pin, LOW);
    delayMicroseconds(PWM_PERIOD - pulseWidthMicros); // Compléter le cycle
    WebSerial.print("Valeur d'angle pour le pin ");
    WebSerial.println(pin);
    WebSerial.println(pulseWidthMicros);
    Serial.print("Valeur d'angle pour le pin ");
    Serial.println(pin);
    Serial.println(pulseWidthMicros);
}
```



```
// Signal PWM pour les leds
void generatePWMLed(int pin, int dutyCycle) {
    int pulseWidthMicros = (1000000 / LED_PWM_FREQUENCY) * dutyCycle / LED_PWM_RESOLUTION;
    int offTime = (1000000 / LED_PWM_FREQUENCY) - pulseWidthMicros;
    digitalWrite(pin, HIGH);
    delayMicroseconds(pulseWidthMicros); // Maintenir HIGH pour la durée du duty cycle
    digitalWrite(pin, LOW);
    delayMicroseconds(offTime); // Compléter le cycle LOW
}

// Lecture signal PWM en entrée
uint16_t readPWM(int pin) {
    return pulseIn(pin, HIGH, PWM_PERIOD); // Lire la durée HIGH en µs
}

void checkEmergencyStop() {
    static unsigned long startTime = 0; // Variable statique pour mémoriser le début de la détection
    static bool conditionMet = false; // Indicateur pour savoir si la condition est remplie
    uint16_t autoSignal = readPWM(PWM_PIN_OUT);
    uint16_t sequenceSignal = readPWM(SEQUENCE_PIN);
    if (autoSignal <= 1200 || sequenceSignal <= 1200) {
        if (!conditionMet) {
            startTime = millis(); // On enregistre le temps de début
            conditionMet = true; // On marque que la condition est remplie
        } else if (millis() - startTime >= 500) { // On vérifie si la condition est restée vraie pendant au moins 500 ms
            Serial.println("Signal d'arrêt détecté. Activation de la LED rouge.");
            WebSerial.println("Signal d'arrêt détecté. Activation de la LED rouge.");
            ledRedDutyCycle = 255; // Activer la LED rouge à pleine intensité
            ledGreenDutyCycle = 0; // Désactiver la LED verte
            // Boucle infinie simulant l'arrêt d'urgence
            while (true) {
                generatePWMLed(LED_RED, ledRedDutyCycle);
                generatePWMLed(LED_GREEN, ledGreenDutyCycle);
            }
        }
    } else {
        conditionMet = false; // Réinitialiser si la condition n'est plus remplie
    }
}

void Demarrage() {
    for (int pulseWidth = 1000; pulseWidth <= V_neutre; pulseWidth++) {
        checkEmergencyStop();
        generatePWM(PWM_PIN_3, pulseWidth);
        generatePWMLed(LED_GREEN, ledGreenDutyCycle);
    }
}

void Phase_1() {
    alpha = sign(B_z - zmes);
    zcons = zcons + alpha*dt*V_reelle;
    if (alpha == 1) {
        zcons = max(zcons, B_z);
    }
    else if (alpha == -1) {
        zcons = min(zcons, B_z);
    }
    erreur = zcons - zmes;
    pwm_output = (int) ceil(V_neutre + Kp*erreur);
    if (pwm_output > V_max) {
```



```
pwm_output = V_max;
}
generatePWM(PWM_PIN_3, pwm_output);
erreurU = A_x - xmes;
erreurV = A_y - ymes;
U = KpU*erreurU;
V = KpV*erreurV;
pwm_output_theta = atan(U);
pwm_output_theta_degres = pwm_output_theta * 180.0/PI;
pwm_output_phi = -atan(V*cos(pwm_output_theta));
pwm_output_phi_degres = pwm_output_phi * 180.0/PI;
generatePWMAngle(PWM_PIN_2, pwm_output_theta_degres);
generatePWMAngle(PWM_PIN_1, pwm_output_phi_degres);
Serial.print("Pousée : ");
Serial.println(pwm_output);
Serial.print("Erreurs angles");
Serial.print(erreurU);
Serial.print(" ; ");
Serial.println(erreurV);
Serial.print("Angles : ");
Serial.print(pwm_output_theta_degres);
Serial.print(" ; ");
Serial.println(pwm_output_phi_degres);
checkEmergencyStop();
}

void debutStabilisation() {
    startTime = millis();
    stabilisationEnCours = true;
}

void Stabilisation_phase_1() {
    // Vérifie si on doit encore stabiliser
    if (!stabilisationEnCours) return;
    // Vérification du temps écoulé
    if (millis() - startTime < duree_stabilisation) {
        alpha = sign(B_z - zmes);
        zcons = zcons + alpha*dt*V_reelle_stab;
        if (alpha == 1) {
            zcons = max(zcons, B_z);
        }
        else if (alpha == -1) {
            zcons = min(zcons, B_z);
        }
        erreur = zcons - zmes;
        pwm_output = (int) ceil(V_neutre + Kp*erreur);
        if (alpha == 1) {
            if (pwm_output > V_max) {
                pwm_output = V_max;
            }
        }
        if (alpha == -1) {
            if (pwm_output < V_min) {
                pwm_output = V_min;
            }
        }
        generatePWM(PWM_PIN_3, pwm_output);
        erreurU = B_x - xmes;
        erreurV = B_y - ymes;
        U = KpU*erreurU;
        V = KpV*erreurV;
        pwm_output_theta = atan(U);
```



```
pwm_output_theta_degres = pwm_output_theta * 180.0/PI;
pwm_output_phi = -atan(V*cos(pwm_output_theta));
pwm_output_phi_degrees = pwm_output_phi * 180.0/PI;
generatePWMangle(PWM_PIN_2, pwm_output_theta_degrees);
generatePWMangle(PWM_PIN_1, pwm_output_phi_degrees);
Serial.print("Pousée : ");
Serial.println(pwm_output);
Serial.print("Erreurs angles");
Serial.print(erreurU);
Serial.print(" ; ");
Serial.println(erreurV);
Serial.print("Angles : ");
Serial.print(pwm_output_theta_degrees);
Serial.print(" ; ");
Serial.println(pwm_output_phi_degrees);
checkEmergencyStop();
} else {
    // Durée de stabilisation écoulée
    stabilisationEnCours = false;
    Serial.println("Stabilisation terminée");
}
}

// Phase de vol horizontal
void Phase_2(){
    alpha = sign(C_z - zmes);
    zcons = zcons + alpha*dt*V_reelle;
    if (alpha == 1) {
        zcons = max(zcons, C_z);
    }
    else if (alpha == -1) {
        zcons = min(zcons, C_z);
    }
    erreur = zcons - zmes;
    pwm_output = (int) ceil(V_neutre + Kp*erreur);
    if (alpha == 1) {
        if (pwm_output > V_max) {
            pwm_output = V_max;
        }
    }
    if (alpha == -1) {
        if (pwm_output < V_min) {
            pwm_output = V_min;
        }
    }
    generatePWM(PWM_PIN_3, pwm_output);
    erreurU = C_x - xmes;
    erreurV = C_y - ymes;
    U = KpU*erreurU;
    V = KpV*erreurV;
    pwm_output_theta = atan(U);
    pwm_output_theta_degrees = pwm_output_theta * 180.0/PI;
    pwm_output_phi = -atan(V*cos(pwm_output_theta));
    pwm_output_phi_degrees = pwm_output_phi * 180.0/PI;
    generatePWMangle(PWM_PIN_2, pwm_output_theta_degrees);
    generatePWMangle(PWM_PIN_1, pwm_output_phi_degrees);
    Serial.print("Pousée : ");
    Serial.println(pwm_output);
    Serial.print("Erreurs angles");
    Serial.print(erreurU);
    Serial.print(" ; ");
    Serial.println(erreurV);
    Serial.print("Angles : ");
```



```
Serial.print(pwm_output_theta_degrees);
Serial.print(" ; ");
Serial.println(pwm_output_phi_degrees);
checkEmergencyStop();
}

void Stabilisation_phase_2() {
    // Vérifie si on doit encore stabiliser
    if (!stabilisationEnCours) return;
    // Vérification du temps écoulé
    if (millis() - startTime < duree_stabilisation) {
        alpha = sign(C_z - zmes);
        zcons = zcons + alpha*dt*V_reelle_stab;
        if (alpha == 1) {
            zcons = max(zcons, C_z);
        }
        else if (alpha == -1) {
            zcons = min(zcons, C_z);
        }
        erreur = zcons - zmes;
        pwm_output = (int) ceil(V_neutre + Kp*erreur);
        if (alpha == 1) {
            if (pwm_output > V_max) {
                pwm_output = V_max;
            }
        }
        if (alpha == -1) {
            if (pwm_output < V_min) {
                pwm_output = V_min;
            }
        }
        generatePWM(PWM_PIN_3, pwm_output);
        erreurU = C_x - xmes;
        erreurV = C_y - ymes;
        U = KpU*erreurU;
        V = KpV*erreurV;
        pwm_output_theta = atan(U);
        pwm_output_theta_degrees = pwm_output_theta * 180.0/PI;
        pwm_output_phi = -atan(V*cos(pwm_output_theta));
        pwm_output_phi_degrees = pwm_output_phi * 180.0/PI;
        generatePWMMangle(PWM_PIN_2, pwm_output_theta_degrees);
        generatePWMMangle(PWM_PIN_1, pwm_output_phi_degrees);
        Serial.print("Pousée : ");
        Serial.println(pwm_output);
        Serial.print("Erreurs angles");
        Serial.print(erreurU);
        Serial.print(" ; ");
        Serial.println(erreurV);
        Serial.print("Angles : ");
        Serial.print(pwm_output_theta_degrees);
        Serial.print(" ; ");
        Serial.println(pwm_output_phi_degrees);
        checkEmergencyStop();
    } else {
        // Durée de stabilisation écoulée
        stabilisationEnCours = false;
        Serial.println("Stabilisation terminée");
    }
}

// Phase d'atterrissage
void Phase_3() {
```



```
alpha = sign(D_z - zmes);
zcons = zcons + alpha*dt*V_reelle;
if (alpha == 1) {
    zcons = max(zcons, D_z);
}
else if (alpha == -1) {
    zcons = min(zcons, D_z);
}
erreur = zcons - zmes;
pwm_output = (int) ceil(V_neutre + Kp*erreur);
if (pwm_output < V_min) {
    pwm_output = V_min;
}
generatePWM(PWM_PIN_3, pwm_output);
erreurU = D_x - xmes;
erreurV = D_y - ymes;
U = KpU*erreurU;
V = KpV*erreurV;
pwm_output_theta = atan(U);
pwm_output_theta_degrees = pwm_output_theta * 180.0/PI;
pwm_output_phi = -atan(V*cos(pwm_output_theta));
pwm_output_phi_degrees = pwm_output_phi * 180.0/PI;
generatePWMAngle(PWM_PIN_2, pwm_output_theta_degrees);
generatePWMAngle(PWM_PIN_1, pwm_output_phi_degrees);
Serial.print("Pousée : ");
Serial.println(pwm_output);
Serial.print("Erreurs angles");
Serial.print(erreurU);
Serial.print(" ; ");
Serial.println(erreurV);
Serial.print("Angles : ");
Serial.print(pwm_output_theta_degrees);
Serial.print(" ; ");
Serial.println(pwm_output_phi_degrees);
checkEmergencyStop();
}

int sign(float x) {
    if (x >= 0) return 1;
    else if (x < 0) return -1;
    else return 0;
}

// Arrêt des moteurs
void Arret() {
    for (int pulseWidth = V_neutre; pulseWidth >= 1000; pulseWidth--) {
        checkEmergencyStop();
        generatePWM(PWM_PIN_3, pulseWidth);
        generatePWMLed(LED_GREEN, ledGreenDutyCycle);
    }
}

// Fonction de callback pour WebSerial
void recvMsg(uint8_t *data, size_t len) {
    // Ici vous gérez les données reçues via WebSerial
    Serial.print("Reçu sur WebSerial: ");
    for (size_t i = 0; i < len; i++) {
        Serial.write(data[i]);
    }
    Serial.println();
}
```



/// ----- Boucle principale ----- ///

```
void setup() {
    // Initialisation de la communication série
    Serial.begin(115200);
    Serial2.begin(115200, SERIAL_8N1, 16, 17);
    Wire.begin(SDA_PIN, SCL_PIN);
    WebSerial.println("Setup started.");
    Serial.println("Setup started.");
    delay(5000);

    // Configuration des broches ESC comme sortie
    pinMode(PWM_PIN_1, OUTPUT); //AIL (Roulis)
    pinMode(PWM_PIN_2, OUTPUT); //ELE (Tangage)
    pinMode(PWM_PIN_3, OUTPUT); //THR (Poussée)
    pinMode(PWM_PIN_4, OUTPUT); //RUD (Lacet)
    pinMode(PWM_PIN_OUT, INPUT); // Mode auto
    pinMode(SEQUENCE_PIN, INPUT); // Armement
    pinMode(LED_RED, OUTPUT); // Led rouge
    pinMode(LED_GREEN, OUTPUT); // Led verte
    WebSerial.println("Pins configurés. Prêt au départ !");
    Serial.println("Pins configurés. Prêt au départ !");

    // Connexion WiFi en mode station
    WiFi.mode(WIFI_STA);
    WiFi.begin(ssid, password);
    // Attente de la connexion WiFi
    if (WiFi.waitForConnectResult() != WL_CONNECTED) {
        Serial.println("Echec de la connexion WiFi !");
        return;
    }
    Serial.print("Adresse IP: ");
    Serial.println(WiFi.localIP());
    // Initialisation de WebSerial
    WebSerial.begin(&server); // passer le serveur créé
    //WebSerial.setReceiveCallback(recvMsg);
    WebSerial.onMessage(recvMsg);

    server.begin();

    WebSerial.println("Hello from ESP!");

    // Initialisation de l'écran OLED
    if (!display.begin(SSD1306_SWITCHCAPVCC, SCREEN_I2C_ADDR)) {
        Serial.println("Echec de l'initialisation de l'écran OLED");
        for (;;) {
    }
    display.clearDisplay();
    display.display();
    inputValue = "";
    updateDisplay("Bonjour ! Pret a embarquer ?\n");
}

void loop() {
    unsigned long currentTime = micros();
    dt = (currentTime - previousTime) / 1000000.0; // Conversion en secondes
    previousTime = currentTime;

    // Lecture du signal sur PWM_PIN_OUT
    uint16_t autoSignal = pulseIn(PWM_PIN_OUT, HIGH, 25000); // Mesure impulsion haute sur 25ms max
```



```
// Lecture du signal sur SEQUENCE_PIN
uint16_t sequenceSignal = pulseIn(SEQUENCE_PIN, HIGH, 25000); // Idem mais sur SEQUENCE_PIN

// Lecture des données UART de la position
while (Serial2.available() > 0) {
    char c = (char)Serial2.read();
    if (c == '\n') {
        // On a reçu une ligne complète, par ex: "1.50,2.20,3.40"
        // Découpage de la ligne sur les virgules
        int firstComma = ligne.indexOf(',');
        int secondComma = ligne.indexOf(',', firstComma + 1);

        if (firstComma > 0 && secondComma > 0) {
            String xStr = ligne.substring(0, firstComma);
            String yStr = ligne.substring(firstComma + 1, secondComma);
            String zStr = ligne.substring(secondComma + 1);

            xmes = xStr.toFloat();
            ymes = yStr.toFloat();
            zmes = zStr.toFloat();

            WebSerial.println("Valeurs reçues pour dt :");
            WebSerial.print("dt = "); WebSerial.println(dt, 2);
            WebSerial.print("X = "); WebSerial.println(xmes, 2);
            WebSerial.print("Y = "); WebSerial.println(ymes, 2);
            WebSerial.print("Z = "); WebSerial.println(zmes, 2);

            Serial.println("Valeurs reçues :");
            Serial.print("X = "); Serial.println(xmes, 2);
            Serial.print("Y = "); Serial.println(ymes, 2);
            Serial.print("Z = "); Serial.println(zmes, 2);

        } else {
            WebSerial.println("Erreur : la ligne ne contient pas les virgules attendues.");
            Serial.println("Erreur : la ligne ne contient pas les virgules attendues.");
        }
    }

    // Réinitialisation de la ligne pour la prochaine réception
    ligne = "";
} else if (c != '\r') {
    // Accumuler les caractères (ignorer '\r')
    ligne += c;
}

if (autoSignal >= 1500 && Autorisation_preparation_decollage == false) {
    switch (etatSaisie) {
        case 0:{ // État d'affichage initial
            display.clearDisplay();
            updateDisplay("Mode automatique détecté.\n");
            updateDisplay("Veuillez indiquer la distance à parcourir.\n");

            ledRedDutyCycle = 255; // LED rouge à pleine intensité
            ledGreenDutyCycle = 0; // LED verte éteinte
            generatePWMLed(LED_RED, ledRedDutyCycle);
            generatePWMLed(LED_GREEN, ledGreenDutyCycle);

            etatSaisie = 1; // Passer à l'état de saisie
            break;
        }
        case 1:{ // Saisie de la distance
            char key = customKeypad.getKey();
            if (key) {
                if (key >= '0' && key <= '9') {
```



```
// Ajout du chiffre à la saisie
inputValue += key;
display.clearDisplay(); // On efface pour afficher la nouvelle valeur
updateDisplay("Saisissez la distance :\n" + inputValue);
delay(100); // Petit délai pour éviter le double appui
} else if (key == 'A') {
    // Réinitialisation de la saisie
    inputValue = "";
    etatSaisie = 0; // Revenir à l'état d'affichage initial
} else if (key == 'D') {
    // Validation de la saisie
    if (inputValue.length() > 0) {
        Distance = inputValue.toFloat(); // Convertir la distance en float
        display.clearDisplay();
        updateDisplay("Valeur enregistrée :\n" + inputValue);
        delay(1000);
        showSmileyFace();
        etatSaisie = 2; // Passer à l'étape suivante
    }
}
break;
}
case 2: { // Préparation au décollage
    display.clearDisplay();
    updateDisplay("Début de la préparation au vol.\nVeuillez vous écartez du drone et armer le.");
    Autorisation_preparation_decollage = true;
    delay(2000);
    break;
}
}
}

// Recalcul du plan de vol
if (Autorisation_preparation_decollage == true && Preparation_decollage == false) {
    ledRedDutyCycle = 255; // LED rouge à pleine intensité
    ledGreenDutyCycle = 0; // LED verte éteinte
    generatePWMLed(LED_RED, ledRedDutyCycle);
    generatePWMLed(LED_GREEN, ledGreenDutyCycle);

    // Recalcul des coordonnées
    C_x = Distance;
    D_x = Distance;

    if (C_x == 0 && D_x == 0){
        Ne_pas_executer_la_phase_2 = true;
    }
    display.clearDisplay();
    inputValue = "";
    updateDisplay("Recalcul du plan de vol...\n");
    delay(5000);
    Preparation_decollage = true;
}

// Indication pour prévenir que le drone est prêt à partir
if (Autorisation_preparation_decollage == true && Preparation_decollage == true) {
    ledRedDutyCycle = 0; // LED rouge éteinte
    ledGreenDutyCycle = 255; // LED verte à pleine intensité
    generatePWMLed(LED_RED, ledRedDutyCycle);
    generatePWMLed(LED_GREEN, ledGreenDutyCycle);

    display.clearDisplay();
    inputValue = "";
    updateDisplay("Prêt au décollage Captain !\n");
```



```
}
```

```
// Vol automatique
if (Autorisation_preparation_decollage == true && Preparation_decollage == true && sequenceSignal > 1500) {
    checkEmergencyStop();
    if (!missionComplete) {
        switch (currentPhase) {
            case START:
                Demarrage();
                currentPhase = PHASE_1;
                break;
            case PHASE_1:
                WebSerial.println("Phase 1");
                Serial.println("Phase 1");
                Phase_1();
                if (abs(zmes - hauteur_du_vol) <= 0.1) {
                    currentPhase = STABILIZE_AT_B;
                    Serial.println("Fin de la phase 1");
                    debutStabilisation();
                }
                break;
            case STABILIZE_AT_B:
                WebSerial.println("Vol stationnaire...");
                Serial.println("Vol stationnaire...");
                Stabilisation_phase_1(); // Appel à chaque itération
                // Ne changez de phase que lorsque la stabilisation est terminée
                if (!stabilisationEnCours) {
                    Serial.println("Fin de la stabilisation à B");
                    currentPhase = PHASE_2;
                }
                break;
            case PHASE_2:
                if (Ne_pas_executer_la_phase_2 == true) {
                    currentPhase = PHASE_3;
                    break;
                } else {
                    WebSerial.println("Phase 2...");
                    Serial.println("Phase 2...");
                    Phase_2();
                    if (abs(xmes - Distance) <= 0.1) {
                        currentPhase = STABILIZE_AT_C;
                        debutStabilisation();
                    }
                }
                break;
            case STABILIZE_AT_C:
                WebSerial.println("Vol stationnaire...");
                Serial.println("Vol stationnaire...");
                Stabilisation_phase_2();
                // Ne changez de phase que lorsque la stabilisation est terminée
                if (!stabilisationEnCours) {
                    Serial.println("Fin de la stabilisation à B");
                    currentPhase = PHASE_3;
                }
                break;
            case PHASE_3:
                WebSerial.println("Phase 3...");
                Serial.println("Phase 3...");
                Phase_3();
                if (abs(zmes - D_z) <= 0.1) {
                    currentPhase = STOP;
                }
                break;
            case STOP:

```



```
WebSerial.println("Mise hors tension...");  
Serial.println("Mise hors tension...");  
Arret();  
missionComplete = true; // La mission est terminée  
break;  
}  
}  
else {  
    WebSerial.println("Vol terminé.");  
    Serial.println("Vol terminé.");  
    ledGreenDutyCycle = 0; // LED verte éteinte  
    ledRedDutyCycle = 255; // LED rouge à pleine intensité  
    generatePWMLed(LED_RED, ledRedDutyCycle);  
    display.clearDisplay();  
    updateDisplay("Vol termine. Veuillez me debrancher. A bientot !\n");  
}  
}  
}
```

Code ESP UWB DW1000

```
#include <SPI.h>  
#include "DW1000.h"  
#include "DW1000Ranging.h"  
  
// Configuration des broches pour DW1000  
#define SPI_SCK 18  
#define SPI_MISO 19  
#define SPI_MOSI 23  
#define PIN_RST 27  
#define PIN_IRQ 34  
#define PIN_SS 4  
  
// Adresse du TAG  
char tag_addr[] = "7D:00:22:EA:82:60:3B:9C";  
  
// Coordonnées des ANCHORS  
struct Anchor {  
    uint8_t id;  
    float x, y, z;  
};  
  
Anchor anchors[3] = {  
    {0x84, 0.0, 0.0, 0.0},  
    {0x85, 2.0, 0.0, 0.0},  
    {0x86, 1.0, 1.0, 0.0}  
};  
  
// Variables globales DW1000  
float distances[3] = {0.0, 0.0, 0.0};  
float xmes = 0.0, ymes = 0.0, zmes = 0.0;  
  
// Tampons pour la moyenne glissante  
#define FILTER_WINDOW_SIZE 10  
float x_filter[FILTER_WINDOW_SIZE] = {0};  
float y_filter[FILTER_WINDOW_SIZE] = {0};  
float z_filter[FILTER_WINDOW_SIZE] = {0};  
int filter_index = 0;  
  
// Variables pour garder en mémoire la dernière position validée  
static float last_valid_x = 0.0;  
static float last_valid_y = 0.0;
```



```
static float last_valid_z = 0.0;

void setup() {
    Serial.begin(115200);

    // Initialisation DW1000
    SPI.begin(SPI_SCK, SPI_MISO, SPI_MOSI);
    DW1000Ranging.initCommunication(PIN_RST, PIN_SS, PIN_IRQ);
    DW1000Ranging.attachNewRange(newRange);
    DW1000Ranging.startAsTag(tag_addr, DW1000.MODE_LONGDATA_RANGE_LOWPOWER, false);

    Serial.println("DW1000 prêt. Calcul de la position (x, y, z) ...");

    // Initialisation de l'UART2
    Serial2.begin(115200, SERIAL_8N1, 16, 17);

    Serial.println("Émetteur prêt !");
}

void loop() {
    // Gestion DW1000
    DW1000Ranging.loop();

    static unsigned long lastDisplayTime = 0;
    if (millis() - lastDisplayTime > 100) {
        lastDisplayTime = millis();
        calculateTagPosition();
    }
}

void newRange() {
    DW1000Device *device = DW1000Ranging.getDistantDevice();
    uint8_t shortAddr = device->getShortAddress();

    // Associe la distance à l'ANCHOR correspondante
    for (uint8_t i = 0; i < 3; i++) {
        if (anchors[i].id == shortAddr) {
            distances[i] = device->getRange();
            break;
        }
    }
}

float applyMovingAverage(float *buffer, float newValue) {
    buffer[filter_index % FILTER_WINDOW_SIZE] = newValue;
    filter_index++;
    float sum = 0.0;
    for (int i = 0; i < FILTER_WINDOW_SIZE; i++) {
        sum += buffer[i];
    }
    return sum / FILTER_WINDOW_SIZE;
}

void calculateTagPosition() {
    float r1 = distances[0];
    float r2 = distances[1];
    float r3 = distances[2];

    // Positions des anchors
    float x1 = anchors[0].x;
    float y1 = anchors[0].y;
    float z1 = anchors[0].z;

    float x2 = anchors[1].x;
```



```
float y2 = anchors[1].y;
float z2 = anchors[1].z;

float x3 = anchors[2].x;
float y3 = anchors[2].y;
float z3 = anchors[2].z;

// Calcul de xmes, ymes (projection 2D)
xmes = (r1 * r1 - r2 * r2 + x2 * x2) / (2 * x2);
if (fabs(xmes) < 1e-10) {
    xmes = 0.0;
}
ymes = (r1 * r1 - r3 * r3 + x3 * x3 + y3 * y3 - 2 * x3 * xmes) / (2 * y3);
if (fabs(ymes) < 1e-10) {
    ymes = 0.0;
}

// Calcul du z
if (r1 * r1 - xmes * xmes - ymes * ymes < 0) {
    // Si la valeur est négative, garder la dernière valeur filtrée de z
    zmes = z_filter[(filter_index - 1) % FILTER_WINDOW_SIZE];
} else {
    zmes = sqrt(r1 * r1 - xmes * xmes - ymes * ymes);
}

// Appliquer la moyenne glissante
xmes = applyMovingAverage(x_filter, xmes);
ymes = applyMovingAverage(y_filter, ymes);
zmes = applyMovingAverage(z_filter, zmes);

// Mémoriser les anciennes valeurs validées
float old_x = last_valid_x;
float old_y = last_valid_y;
float old_z = last_valid_z;

// Vérifier si l'écart entre les nouvelles valeurs et les précédentes est > 0.7
if (fabs(xmes - old_x) > 0.7 || fabs(ymes - old_y) > 0.7) {
    // Trop de différence, on rétablit les anciennes valeurs validées
    xmes = old_x;
    ymes = old_y;
    zmes = old_z;
} else {
    // Mise à jour des dernières valeurs validées
    last_valid_x = xmes;
    last_valid_y = ymes;
    last_valid_z = zmes;
}

// Afficher les valeurs
Serial.print("Position filtrée : r1 = ");
Serial.print(r1, 3);
Serial.print(", r2 = ");
Serial.print(r2, 3);
Serial.print(", r3 = ");
Serial.println(r3, 3);
Serial.print("Position filtrée : x = ");
Serial.print(xmes, 3);
Serial.print(", y = ");
Serial.print(ymes, 3);
Serial.print(", z = ");
Serial.println(zmes, 3);

Serial2.print(xmes, 2); // x avec 2 décimales
Serial2.print(",");
```



```
Serial2.print(ymes, 2); // y avec 2 décimales
Serial2.print(",");
Serial2.println(zmes, 2);
}
```

All rights reserved

Listes des figures

Figure 1 : Schéma du parcours devant être réalisé par le drone en mode automatique	6
Figure 2 : DCU en phase d'utilisation	8
Figure 3 : DCU en phase de charge.....	8
Figure 4 : DCS.....	9
Figure 5 : Diagramme des transitions.....	10
Figure 6 : Scénario opérationnel en situation normal de fonctionnement.....	11
Figure 7 : Scénario opérationnel en cas de problème détecté durant le vol	11
Figure 8 : Logigramme de fonctionnement général.....	16
Figure 9 : Logigramme des définitions	18
Figure 10 : Logigramme de l'initialisation du système	18
Figure 11 : Logigramme de l'acquisition des paramètres	18
Figure 12 : Logigramme de la lecture des données de position reçues	19
Figure 13 : Logigramme de la détection du mode automatique et de la saisie de la distance à parcourir.....	20
Figure 14 : Logigramme du recalcul du plan de vol.....	21
Figure 15 : Logigramme de l'indication de la préparation au décollage	21
Figure 16 : Logigramme de la phase de vol automatique	22
Figure 17 : Logigramme des phases de déplacement en vol	23
Figure 18 : Logigramme des phases de stabilisation en vol	23
Figure 19 : Vue latérale du plan de vol.....	24
Figure 20 : Vue 3D du plan de vol.....	24
Figure 21 : Modèle 3D initial du système volant.....	26
Figure 22 : Modèle réel du système volant initial	26
Figure 23 : Envergure du système	27
Figure 24 : Pesée du système volant réel.....	28
Figure 25 : Matrices d'inertie déterminés numériquement grâce au modèle 3D	28
Figure 26 : Surfaces projetées vue du dessous	29
Figure 27 : Valeur des surfaces projetées vue de côté.....	29
Figure 28 : Modèle Simulink complet.....	32
Figure 29 : Résultats du respect du couloir de vol	33
Figure 30 : Résultats télémétriques du respect du couloir de vol.....	34
Figure 31 : Résultats du respect du couloir de vol avec bruit	35
Figure 32 : Résultats télémétriques du respect du couloir de vol avec bruit.....	36
Figure 33 : Boîte grise générale.....	39
Figure 34 : Boîte grise des sous-systèmes terrestre et balise	40
Figure 35 : Boîte grise du sous-système volant.....	41
Figure 36 : Courbes des critères de pilotage en phase de montée	43
Figure 37 : Courbes des critères de pilotage en phase de descente	43
Figure 38 : Moteur AIR2216 KV880	44
Figure 39 : Données constructeur des moteurs AIR2216 KV880	44
Figure 40 : Hélices T-motor T1045	45
Figure 41 : Batterie LiPo 3700mAh, 16,8V.....	45
Figure 42 : Simulation de l'autonomie de la batterie dans la configuration initiale	46

Figure 43 : Distributeur Holybro PM06 V2	46
Figure 44 : FlySky FS-i6 et FS-iA6B.....	46
Figure 45 : Mesure de la fréquence de rafraîchissement de la communication radio	47
Figure 46 : ESP32 UWB DW1000	52
Figure 47 : Ecran OLED 0,96 pouces	52
Figure 48 : Keypad 4x4	52
Figure 49 : ESP32-WROOM-32	52
Figure 50 : Encodeur 8ch PWM vers PPM	52
Figure 51 : LED verte.....	53
Figure 52 : Commutateur 4ch trainer.....	53
Figure 53 : Vue de face du banc d'essai	55
Figure 54 : Vue de côté du banc d'essai	55
Figure 55 : Modèle 3D de la première version du support sous Fusion 360.....	56
Figure 56 : Impression de la première version du support	57
Figure 57 : Modèle 3D de la deuxième version du support sous Fusion 360	58
Figure 58 : Impression de la deuxième version du support	58
Figure 59 : Modélisation 3D de la version finale du support sous Fusion 360.....	59
Figure 60 : Impression de la version finale du support	59
Figure 61 : Plan industriel du support	59
Figure 62 : Vue du support avec l'assemblage complet sur la base volante.....	60
Figure 63 : Modélisation complète du système volant sous Catia V5.....	61
Figure 64 : Résultats numériques du centre de gravité et des matrices d'inerties avec l'assemblage complet.....	61
Figure 65 : Mesure expérimentale du poids du système volant	62
Figure 66 : Plan industriel du châssis de la base volante	64
Figure 67 : Premier plan industriel des pieds de la base volante.....	64
Figure 68 : Deuxième plan industriel des pieds de la base volante	65
Figure 69 : Simulation de l'autonomie de la batterie dans la configuration finale.....	66
Figure 70 : Schéma électrique autour de l'ESP32-WROOM-32.....	68
Figure 71 : Schéma électrique du récepteur de position ESP32 UWB DW1000	69
Figure 72 : Test de la base volante sous la supervision de nos autorités techniques.....	72
Figure 73 : Assemblage complet de la base volante	73
Figure 74 : Essai de repérage au sol rapproché.....	75
Figure 75 : Essai de repérage au sol éloigné	75
Figure 76 : Essai de repérage dans l'espace avec l'ensemble du système volant	76
Figure 77 : Acquisition du positionnement et traitement de l'incidence de correction pour une position en (0, 0, 0).....	76
Figure 78 : Essai de repérage dans l'espace avec incidence de correction pour une position en (0, 0, 0)	76
Figure 79 : Montage du test avec l'apparition des engrenages pour montrer l'enregistrement de la distance saisie.....	78
Figure 80 : Essai d'armement et de désarmement	82
Figure 81 : Essai de commutation	82
Figure 82 : Essai de compréhension des commandes de l'opérateur par le contrôleur de vol	82
Figure 83 : Visualisation de la consigne de poussée requise pour une stabilisation en fonction de la décharge de la batterie	85



Figure 84 : Détermination de l'autonomie de la batterie et des plages de poussée requises.....	85
Figure 85 : Visualisation des paliers de stabilisation à pleine charge	86
Figure 86 : Visualisation de l'interface Web utilisée	86
Figure 87 : Dispositif de test avec le drone au sol durant la préparation au vol	90
Figure 88 : Dispositif de test avec le déplacement du drone durant la séquence de vol	91
Figure 89 : Photo du drone en vol semi-automatique	95
Figure 90 : Actionner l'armement sur la radiocommande	96
Figure 91 : Actionner le mode automatique sur la radiocommande	97
Figure 92 : Schéma de positionnement des balises au sol pour le vol automatique	99
Figure 93 : Diagramme de Gantt du projet	100