



POLYTECH[°]
NICE SOPHIA



UNIVERSITÉ
CÔTE D'AZUR

R&D PROJECT REPORT

DRONE DESIGN & CONTROL

Prepared By

Nino Mulac - nino.mulac@etu.unice.fr

Jaime Alba - jaime.alba-pastor@etu.unice.fr

Table des matières

1. Introduction.....	3
2. Modélisation et simulation.....	4
2.1. Définition du système.....	4
2.2. Design du contrôleur sur Simulink.....	5
3. Drone de test.....	8
3.1. SpeedyBee V4.....	8
3.2. ESP32 contrôleur à travers SBUS.....	8
4. Design du drone.....	10
4.1. Conception physique.....	10
4.2. PCB sur Kicad.....	11
4.3. Résultats.....	11
5. Librairie python ftprci.....	12
5.1. Avantages de la librairie.....	12
5.2. Progrès.....	13
6. Contrôle par caméras.....	14
6.1. Caméras utilisées.....	14
6.2. Calibration des caméras.....	14
6.3. Identification des leds.....	15
6.4. Identification des leds.....	15

1. Introduction

Dans le cadre de notre projet universitaire, nous avons choisi de nous attaquer à un défi ambitieux : concevoir et contrôler un drone autonome, entièrement personnalisé. Ce projet vise à réunir des compétences variées allant de la simulation des systèmes à l'élaboration d'un PCB pour le contrôleur de vol, ainsi qu'au développement de boucles de contrôle avancées basées sur des caméras infrarouges pour la localisation. L'objectif final est double : construire un drone 100% personnalisé et parvenir à le contrôler de manière autonome dans un environnement clos équipé de caméras sur les murs.

Réaliser un tel projet pose plusieurs défis techniques et conceptuels. Tout d'abord, la conception d'un drone sur mesure nécessite une coordination précise entre les aspects mécaniques, électroniques et logiciels. Chaque composant doit être soigneusement sélectionné et intégré pour garantir une compatibilité et une performance optimale. La conception du PCB pour le contrôleur de vol implique, par exemple, de résoudre des problèmes complexes d'agencement de circuits, de gestion de l'alimentation et de traitement des signaux provenant des capteurs.

Ensuite, la mise en place d'un contrôle autonome basé sur des caméras infrarouges ajoute un niveau supplémentaire de complexité. Les caméras doivent être calibrées avec précision pour capturer et traiter en temps réel la position du drone dans l'espace. Cela nécessite l'intégration d'algorithmes robustes capables de traiter des données bruitées et d'intégrer ces informations dans une boucle de contrôle stable.

Ce projet, à la fois théorique et pratique, constitue une opportunité unique d'appliquer nos connaissances en robotique, électronique et traitement de l'image. Il illustre à quel point la collaboration interdisciplinaire est essentielle pour résoudre des problèmes complexes et réaliser des systèmes autonomes avancés.

2. Modélisation et simulation

2.1. Définition du système

Pour modéliser le mouvement d'un drone, il est nécessaire de prendre en compte les forces et moments agissant sur celui-ci. Ces équations permettent de simuler précisément le comportement du drone et constituent la base pour concevoir un contrôleur efficace.

2.1.1. Somme des forces

Le mouvement translatoire du drone est régi par la deuxième loi de Newton:

$$m\ddot{\vec{r}} = \vec{F}_{grav} + \vec{F}_{drag} + \vec{F}_{motors}$$

Avec:

- \vec{r} la position du drone dans le référentiel terrestre
- m la masse du drone
- $\vec{F}_{grav} = m\vec{g}$ la force gravitationnelle
- \vec{F}_{drag} la force de traînée, proportionnelle à la vitesse
- \vec{F}_{motors} la force générée par les moteurs et les hélices

La force générée par les moteurs dans le référentiel terrestre est exprimée comme:

$$\vec{F}_{motors} = R_{b2e} \cdot T\hat{z}_b$$

Avec R_{b2e} la matrice de rotation entre le repère du drone (body frame), $T\hat{z}_b$ la poussée totale produite par les moteurs dans le repère du drone.

Ainsi, l'accélération dans le référentiel terrestre est donnée par :

$$\ddot{\vec{r}} = \frac{1}{m}(R_{b2e}T\hat{z}_b - \vec{F}_{drag}(v)) - \vec{g}$$

Cette équation montre que le contrôle de la poussée T et de l'orientation du drone (via R_{b2e}) permet de modifier sa position et sa vitesse.

2.1.2. Équations de rotation

En plus de la translation, la dynamique du drone comprend également des rotations. Ces rotations sont décrites par les vitesses angulaires (p, q, r), qui sont les composantes des mouvements autour des axes du repère du drone. Les vitesses angulaires dans le repère du drone sont liées aux taux de changement des angles d'Euler (ϕ, θ, ψ) par la relation :

$$\begin{bmatrix} p \\ q \\ r \end{bmatrix} = \begin{bmatrix} 1 & 0 & -\sin\theta \\ 0 & \cos\phi & \sin\phi \cdot \cos\theta \\ 0 & -\sin\phi & \cos\phi \cos\theta \end{bmatrix} \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix}$$

2.1.3. Dérivation de l'équation de vitesse dans un référentiel en rotation

Le drone évolue dans un référentiel tournant (body frame), ce qui affecte la vitesse relative mesurée. L'équation générale de la vitesse dans un référentiel inertiel est :

$$\frac{dv}{dt} \Big|_{inertial} = \frac{dv}{dt} \Big|_{body} + \omega \times V_i$$

En développant le produit vectoriel nous obtenons:

$$\omega \times V_i = \begin{bmatrix} p \\ q \\ r \end{bmatrix} \times \begin{bmatrix} V_x \\ V_y \\ V_z \end{bmatrix} = \begin{bmatrix} qV_z - rV_y \\ rV_x - pV_z \\ pV_y - qV_x \end{bmatrix} = \begin{bmatrix} 0 & -r & q \\ r & 0 & -p \\ -q & p & 0 \end{bmatrix} \begin{bmatrix} V_x \\ V_y \\ V_z \end{bmatrix}$$

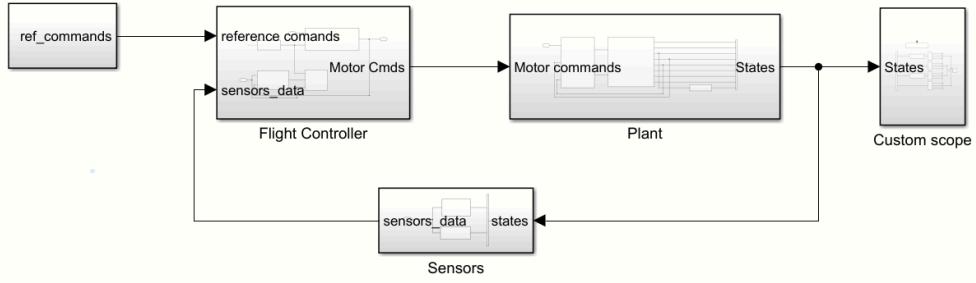
Pour simplifier les calculs, on représente cette opération comme: $\omega \times V_i = [\omega \times] V_i$, où $[\times]$ est une matrice antisymétrique représentant le produit vectoriel: $a \times b = [a \times] b$. Dans ce cas précis on a :

$$[\omega \times] = \begin{bmatrix} 0 & -r & q \\ r & 0 & -p \\ -q & p & 0 \end{bmatrix}$$

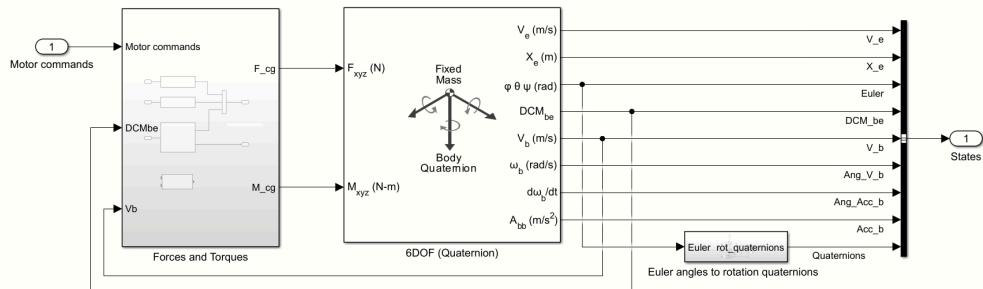
2.2. Design du contrôleur sur Simulink

La boucle principale de contrôle commence par le modèle du système, appelé "Plant", qui représente la dynamique physique complète du drone. Ce modèle fournit en sortie l'état réel du drone, exprimé en termes de position, orientation et vitesses. Ces informations sont ensuite simulées comme étant détectées par les capteurs embarqués du drone (incluant les erreurs et le bruit de mesure).

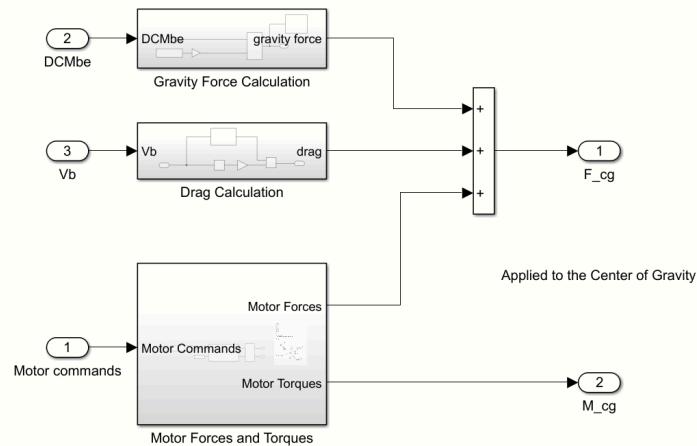
En combinant ces données détectées avec les instructions de référence fournies par l'utilisateur (comme atteindre une certaine position ou orientation), nous générerons les commandes moteur nécessaires pour atteindre ces objectifs.



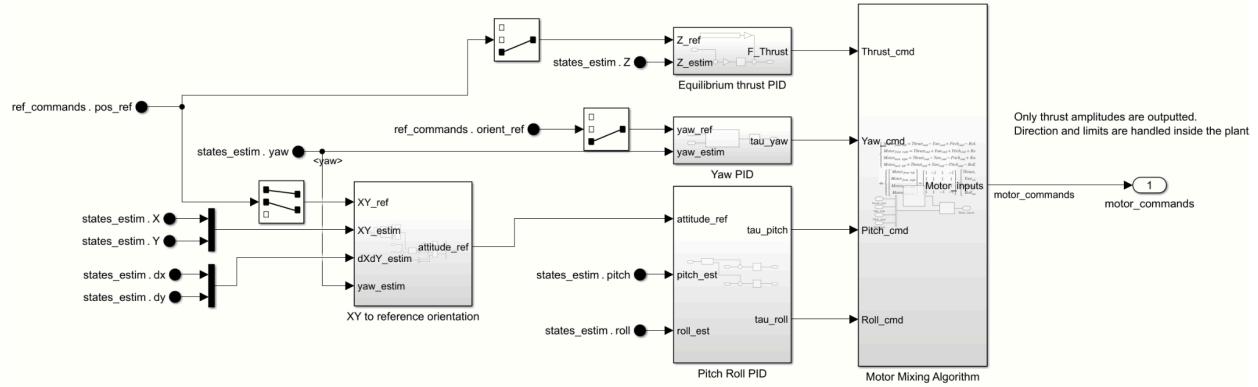
Regardons plus en détail le modèle dynamique du système. Pour simuler le comportement physique du drone, nous calculons toutes les forces qui lui sont appliquées. Ces forces sont exprimées dans le repère du corps (body frame), puis transformées dans le repère terrestre à l'aide des matrices de rotation. Ces calculs permettent de déterminer l'accélération globale du drone, qui est ensuite intégrée pour mettre à jour sa position et son orientation à la prochaine unité de temps.



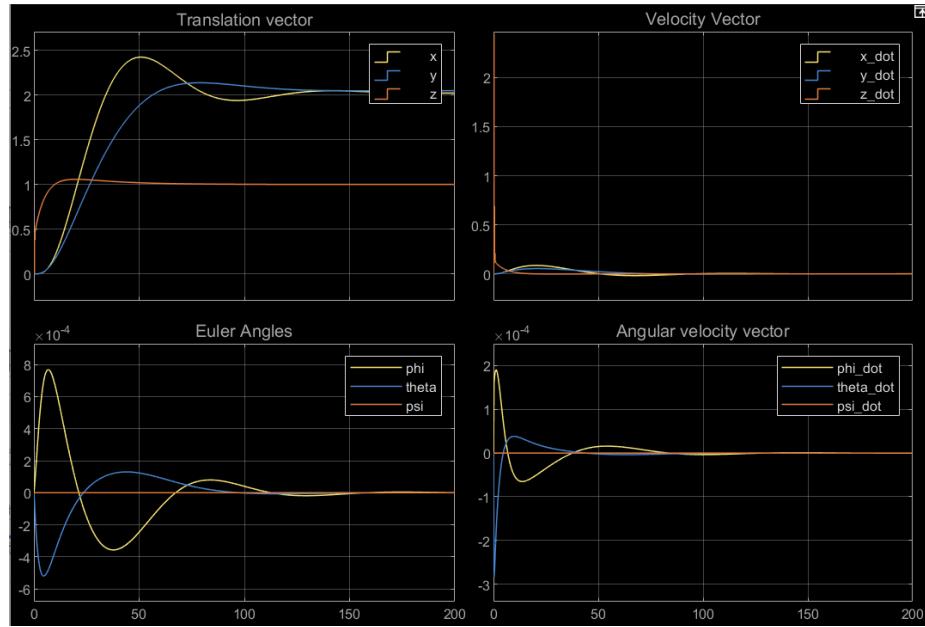
Lors de la génération des forces, nous prenons en compte non seulement les forces externes, comme la gravité et le vent, mais également les forces internes générées par les moteurs comme la poussée totale et le torque. Ces contributions permettent de modéliser fidèlement l'effet des commandes moteur sur la dynamique du drone. Les calculs sont réalisés dans le body frame avant d'être intégrés dans la dynamique globale.



Finalement, lors du contrôle des moteurs nous utilisons plusieurs PID pour contrôler d'abord la hauteur, le yaw, le pitch et le roll. Puis par dessus ça on applique un autre PID pour contrôler les coordonnées x et y à travers le pitch et roll (XY to reference orientation).



Pour illustrer l'efficacité de ce design, considérons une commande simple: moteur à une hauteur de 1 mètre et se déplacer à la position $(x,y) = (2,2)$. En sortie, on obtient une réponse simulée qui montre la trajectoire du drone et les efforts des moteurs pour compenser les forces externes et les perturbations.



3. Drone de test

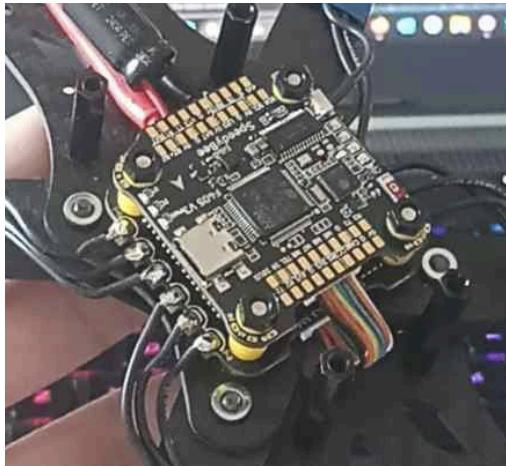
Pour valider nos concepts et approches, nous avons construit un drone de test simplifié, regroupant plusieurs éléments clés adaptés à nos besoins de simulation et de contrôle. Voici les principaux composants utilisés et leur rôle dans le système.

3.1. SpeedyBee V4

Le contrôleur de vol central de notre drone est une carte SpeedyBee V4, sélectionnée pour sa polyvalence et sa compatibilité avec de nombreux capteurs et périphériques. Cette carte offre :

- Une prise en charge des protocoles de communication modernes comme SBUS et UART.
- Une interface simplifiée pour la configuration avec Betaflight, permettant un réglage rapide des paramètres de vol.
- Une intégration facile avec des ESCs pour contrôler la vitesse des moteurs.

La SpeedyBee V4 a été configurée pour fonctionner en mode manuel à l'aide d'une télécommande, tout en gardant la possibilité d'implémenter des modes autonomes à travers des commandes SBUS.



3.2. ESP32 contrôleur à travers SBUS

Pour transmettre les commandes de contrôle au drone, nous avons utilisé un système basé sur deux microcontrôleurs ESP32, tirant parti de leur flexibilité et de leurs capacités de communication.

L'ESP32 principal est relié directement à la SpeedyBee via le protocole SBUS. Ce protocole est largement utilisé dans les contrôleurs de vol pour recevoir des commandes précises concernant les axes de contrôle (tangage, roulis, lacet) et la poussée verticale. Elle reçoit les commandes et les convertit en signaux SBUS adaptés pour la SpeedyBee.

Les commandes de contrôle proviennent d'un second ESP32, qui communique avec le premier via le protocole ESP-NOW qui offre une faible latence et une excellente portée, ce qui le rend parfait pour des communications sans fil en temps réel.

Le deuxième ESP32 est utilisée comme une station de contrôle distante, transmettant les commandes générées par une manette, un ordinateur, ou même par des algorithmes autonomes en cours de développement. La modularité de cette configuration facilite également l'ajout futur de fonctionnalités, comme le contrôle autonome basé sur des capteurs ou des données environnementales.



4. Design du drone

Une version plus avancée du drone étant nécessaire, nous avons commencé à développer un autre drone, plus puissant, plus solide, et embarquant un Raspberry pi pour nous permettre de le contrôler plus facilement. Après avoir choisi les composants, il a fallu adapter le design afin d'avoir un système résistant et fonctionnel.

4.1. Conception physique

Dans le but de résister aux forces importantes(200N en pic) générées par les moteurs, le drone se doit d'être très résistant. Les pièces critiques sont donc en métal, usinées dans des tiges métalliques de 5 mm de diamètre. Elles sont disposées pour avoir des liaisons légèrement désaxées, évitant les rotations parasites et les flexions.

Par ailleurs, pour effectuer des tests sans relâcher le drone et éviter des crashs, nous avons modélisé et fabriqué un bras articulé, avec des contraintes réglables. Il contient un gros ressort pour réduire l'intensité des chocs. Voilà un bref aperçu des pièces :



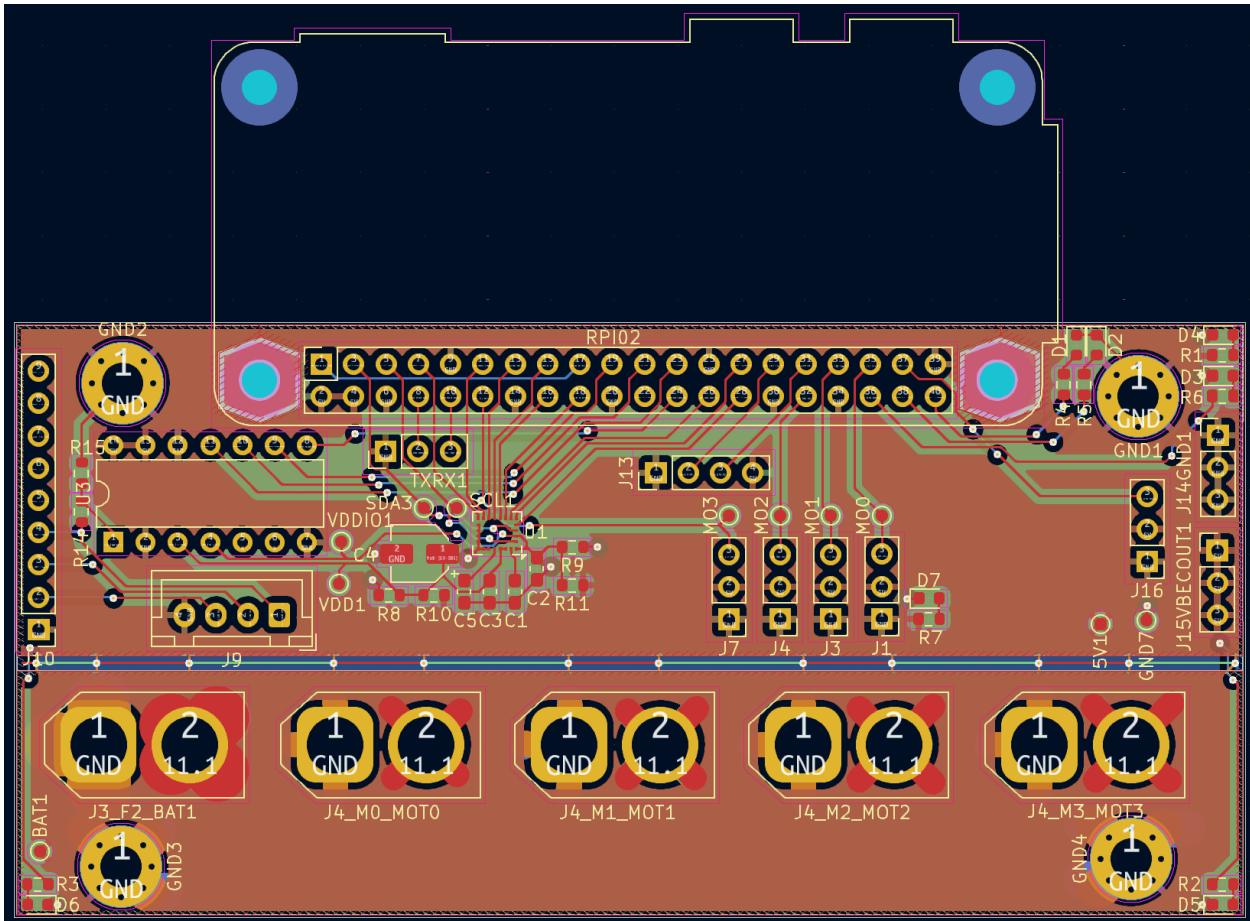
Pièces du bras. Des roulements à billes permettent un mouvement au frottement réduit.



Pièces du drone. Les barres métalliques de 8 cm relient les pièces 1 (qui tient les moteurs) et 2 (avant et arrière du drone), et les deux exemplaires de la 2 en passant par la 3 au centre.

4.2. PCB sur Kicad

Les moteurs ayant besoin d'énormément de courant (60A par moteur en pic), le design d'un PCB dédié a été nécessaire. Il a aussi ensuite permis l'addition de plusieurs capteurs, ainsi que des leds nécessaires à la détection par caméra (voir parties suivantes). La dernière version en date (pas encore fabriquée) est un PCB de quatre étages, faisant particulièrement attention aux perturbations et aux pertes par effet Joule, et bien plus compact que les précédentes. Elle contient un système de mesure de batterie, les connections aux ESCs des moteurs, quatre leds infrarouges, trois leds de signalement, une IMU 9DOF, une interface pour un capteur de distance ultrasonique, et différentes interfaces séries pour ajouter des composants si besoin, le tout piloté par un Raspberry Pi.



4.3. Résultats

L'assemblage final a fonctionné lorsque nous avons testé avec la version de PCB précédente, nous pouvons controller les moteurs, bien qu'il y ait un problème d'initialisation avec un des

ESCs. Par contre, les moteurs vident la batterie en moins de temps que prévu, elle ne tient que quelques minutes, ce qui peut poser problème pour un drone.

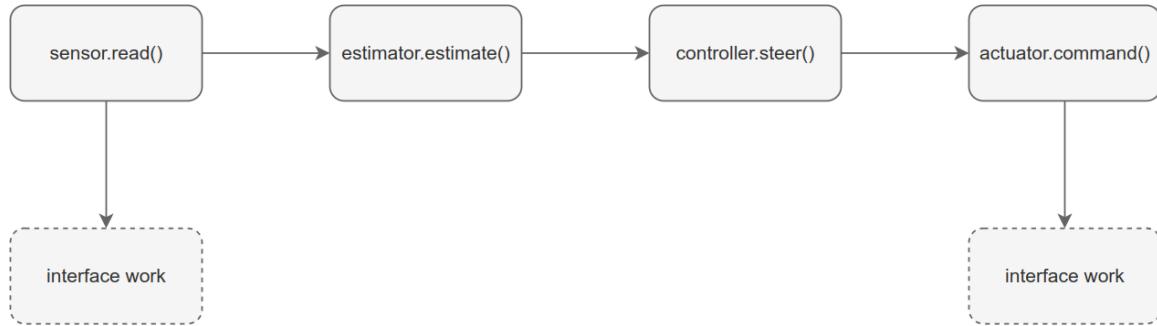


5. Librairie python ftprci

5.1. Avantages de la librairie

ftprci est une librairie Python ayant pour objectif de standardiser les interfaces de programmation de robots. Dans notre cas, ses fonctionnalités intéressantes sont les suivantes :

- Langage Python : Langage simple de très haut niveau facilitant la programmation
- Interfaces simplifiées : Les interfaces sont représentées par des objets, définis de façon claire, permettant des extensions faciles de la librairie, et rendant le code très lisible.



Voici un diagramme de fonctionnement de ftprci. Il est tellement simple qu'un projet ftprci se codera en général en quelques lignes :

```

1 import ftprci as fci
2
3 # Create an instance of the LSM9DS1 sensor
4 sensor: Any = fci.LSM9DS1()
5
6 estimator = fci.ComplementaryFilter()
7
8 pid = fci.PIDController(10, 0, 3, fci.DiscreteIntegral.Tustin())
9
10 actuators = fci.PololuAstar()
11
12 th = fci.RunnerThread(frequency=100)
13
14 th.callback | sensor.read | estimator.estimate | pid.steer | actuators.command

```

- Comme vu dans le code précédent, le code est exécuté dans des threads séparés (`RunnerThread`), il est possible d'exploiter les fonctionnalités de parallélisme des ordinateurs.
- Les boucles de contrôle sont précises en termes de timings (à environ 10ns près), ce qui permet d'avoir des intégrations stables dans les modèles de robots, et donc d'éviter les divergences, malgré la présence du garbage collector et de l'OS qui peuvent préempter et perturber les boucles.

De plus, la librairie est compatible avec les interfaces ROS, bien que nous n'en ayons pas encore besoin. Elle peut aussi tourner non seulement sur raspberry pi, mais aussi sur les cartes arduino récentes, sur les STM32, et sur les ESP32.

5.2. Progrès

Nous avons réalisé une interface pour le capteur LSM9DS1, l'IMU que nous utilisons sur le PCB, permettant de configurer ses registres internes et de récupérer ses données pour la partie accéléromètre/gyroscope (le magnétomètre est séparé). Cependant, un problème de communication nous empêche d'écrire dans d'autres registres que le sixième, donc les données du gyroscope ne sont pas lisibles (sleep mode par défaut tant que le registre 1 n'est pas modifié).

6. Contrôle par caméras

6.1. Caméras utilisées

Pour la partie perception visuelle, nous avons sélectionné les caméras PS3 Eye en raison de leur excellent rapport qualité/prix, leur capacité à capturer des images à haute fréquence, et leur facilité de modification.

Les PS3 Eye offrent les avantages d'une haute fréquence d'image (60 à 120 fps), ce qui est crucial pour capter les mouvements rapides du drone sans perte de précision ; une résolution suffisante (640x480 à 60 fps) pour identifier les marqueurs lumineux (LED IR) tout en maintenant des calculs temps réel rapides ; leur structure permet de retirer et remplacer le filtre IR sans nécessiter d'équipement sophistiqué ; et un coût très abordable, ce qui les rend parfaites pour un projet académique.

Cette modification permet de rendre les caméras sensibles aux LED infrarouges tout en supprimant les interférences des autres sources lumineuses. En plus de ça nous avons installé des LED infrarouges autour de chaque caméra, diffusant une lumière uniforme dans le spectre infrarouge. Ces LED réduisent le bruit blanc en minimisant les perturbations causées par des variations de lumière ambiante.

Cependant, l'utilisation de caméras modifiées présente également certains défis. Le remplacement du filtre IR modifie légèrement les propriétés optiques des caméras, ce qui nécessite une recalibration précise pour garantir une localisation exacte. Les LED infrarouges ont une intensité limitée, ce qui peut restreindre la portée opérationnelle des caméras. Ce problème peut être résolu en ajoutant plus de caméras pour couvrir plus d'espace.

6.2. Calibration des caméras

La calibration des caméras est une étape clé pour garantir une localisation précise du drone. Elle permet de déterminer les paramètres intrinsèques (comme la focale et le centre optique) ainsi que les paramètres extrinsèques (position et orientation de la caméra dans l'espace).

Nous avons utilisé un tableau de carreaux noirs et blancs (calibration checkerboard) pour effectuer cette calibration. Ce processus consiste à capturer des images du tableau depuis différents angles avec chaque caméra, puis identifier les coins du tableau dans chaque image et finalement appliquer l'algorithme de calibration pour récupérer les paramètres intrinsèques.

Après la modification des caméras pour intégrer un filtre IR, les propriétés optiques changent légèrement, rendant cette étape essentielle.

6.3. Identification des leds

Pour suivre le drone, nous avons équipé celui-ci de trois LED infrarouges placées de manière stratégique. Ces LED servent de marqueurs pour identifier la position et l'orientation du drone dans l'espace 3D.

Chaque caméra, grâce à son filtre IR, capture les LED infrarouges sous forme de points lumineux facilement isolables sur l'image. Pour déterminer quelle LED correspond à laquelle sur les images capturées par différentes caméras, nous utilisons les principes de la géométrie épipolaire. Les points détectés sont projetés sur une épipolaire pour limiter les correspondances possibles entre deux images. Une fois les correspondances établies, nous utilisons la triangulation pour calculer la position 3D des LED. En connaissant la disposition géométrique des LED sur le drone, il est possible d'estimer à la fois sa position et son orientation.

Cette partie n'a malheureusement pas abouti du fait qu'on n'avait pas encore fait le cours de computer vision, et donc comprendre et se familiariser avec ces concepts a été très ardu, et a pris trop longtemps.

6.4. Identification des leds

Pour connecter le système de localisation avec le reste des fonctionnalités du drone, nous avons utilisé le framework ROS2. Les positions et orientations calculées sont publiées en temps réel sur des topics ROS, permettant au contrôleur du drone d'accéder aux informations nécessaires pour effectuer les ajustements de trajectoire.

Ça nous permet aussi d'utiliser des outils de visualisation déjà existants pour améliorer et mieux comprendre le système.