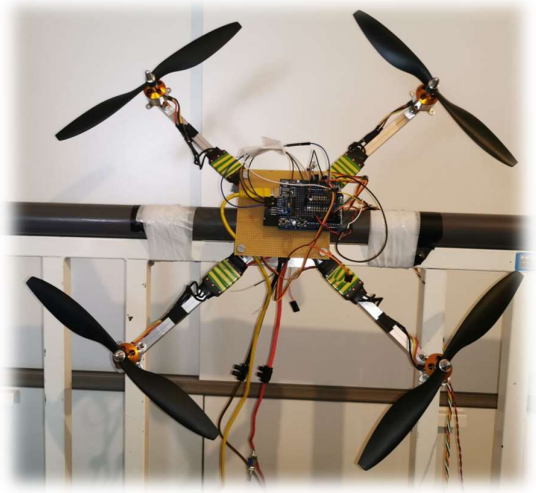


RAPPORT

PROJET 4

Construction d'un drone volant

RESUME

Rapport de projet concernant le semestre 8. Ce rapport va vous présenter les différentes tâches que nous avons mené pour réaliser notre projet.

Richard Simonin, Lukas Fauchois, Evan Gury et Maxime Claudel

TABLE DES MATIERES

TABLE DES ILLUSTRATIONS

INTRODUCTION

Pour répondre à la problématique de notre projet “Construction d’un drone volant”, nous avions dans un premier temps, au semestre 6, opté pour la régulation d’un châssis de drone de type quadcopter déjà existant. Pour ce faire nous utilisions un régulateur de vol existant sur le marché (ArduPilot 2.8) que nous contrôlions avec une connexion en Xbee.

Pour l’année d’IMA4 nous sommes repartis du début en construisant un nouveau châssis, le but étant de n’utiliser que des composants indépendants pour parvenir à un système de contrôle du drone qui nous est propre. La description de nos avancées concernant le S7 et nos objectifs pour le S8 se trouvent dans la partie suivante.

Après une bref récapitulatif des avancées, nous évoquerons la partie « Communication et connectiques » du drone en détaillant la fabrication d’un PCB nécessaire à la fonctionnalité du montage et en expliquant le mode de fonctionnement de l’optimisation de la puissance délivrée par les ESC (Electronic Speed Controllers) ainsi que la communication par ESP32.

Dans un deuxième temps nous évoquerons l’aspect « Régulation » de ce projet, en détaillant l’implémentation d’un régulateur PID ainsi que les montages qui nous ont permis de tester cette régulation du vol du drone. Finalement, nous parlerons des problèmes rencontrés durant ce projet.

RAPPEL DES AVANCEES (S7)

Après avoir réussi à faire voler notre drone grâce à un contrôleur de vol préprogrammé (ArduPilot 2.8) au S6, nous avions décidé de s'occuper nous-mêmes du contrôle et de la régulation du drone et de changer de mode de communication.

Concrètement, l'objectif pour la partie communication était de s'approprier une nouvelle commande de drone à l'aide d'une télécommande bien plus adaptée. Pour la partie régulation, l'objectif était de programmer notre contrôle et notre régulation à partir d'un Arduino UNO et d'une centrale inertielle.

Partie communication

Nous avons, cette année, décidé d'utiliser une télécommande de modélisme Flysky FS-I6X accompagné de son récepteur FS-X6B. Ces composants nous permettent de récupérer, en PWM, les valeurs de nos joysticks directement dans notre programme Arduino.

Nous avons, au S7, effectué des tests pour déterminer les extrema de nos joysticks ainsi qu'une suite de commandes à effectuer sur la manette avant de démarrer les moteurs par souci de sécurité. De plus, nous avons constaté que la quantité de connectiques sur notre drone devenait légèrement complexe, c'est pourquoi nous avons évoqué la possibilité de concevoir un PCB au S8.

Partie Régulation

Pour mener à terme notre objectif, nous avons décidé de travailler avec la centrale inertielle MPU6050 qui comprend un accéléromètre 3 axes et un gyroscope 3 axes, ce qui est nécessaire pour notre drone quadcopter.

Au cours du semestre 7 nous avons donc implémenté un programme Arduino nous permettant de lire les données renvoyées par la gyroscope et l'accéléromètre pour ensuite pouvoir les exploiter. Ainsi, à la fin de ce semestre, nous étions en capacité de gérer les données du MPU6050 et d'appliquer aux moteurs du drone les consignes envoyées par la télécommande grâce au travail de la partie communication. Cependant, la partie de régulation PID n'était pas finalisée, c'est sur quoi nous nous sommes concentrés au semestre 8.

Objectifs pour le semestre 8

Après analyse du travail réalisé au semestre 7, nous en avons fait ressortir quelques objectifs :

- Conception d'un PCB (pour l'ensemble du système)
- Optimiser la distribution de la puissance (esthétiquement)
- Finition de la partie communication envoi/réception de trame validé
- Tests et réglages de la régulation PID
- Implémenter un système de guidage GPS (en fonction du temps dont on dispose)

COMMUNICATION & CONNECTIQUES

Création d'un PCB

Au début de ce semestre, nous avons premièrement voulu fabriquer un PCB afin de rendre les branchements plus simples et le montage plus fonctionnel. L'idée première est de créer un shield que l'on posera sur notre carte Arduino. Nous nous sommes lancés dans la création de ce PCB sur *Altium Designer*. Ce PCB a pour but d'être un shield que l'on posera sur notre carte Arduino. Cependant nous avons rencontré quelques soucis lors de la fabrication de notre PCB ce qui nous a mené à le reproduire sur *Freezing*, afin de disposer des deux versions pour l'impression.

Comme l'indique la capture, nous avons séparé chaque partie, communication, ESC, I2C et nous avons créé des pins pour un capteur de position. Ce shield nous a beaucoup servi lors de nos manipulations car il simplifie les branchements sur la carte.

Ce PCB comporte 4 parties distinctes, la première est pour la communication.

Nous avons en premier lieu la partie alimentation 5V et 0V, et nous avons fait le choix de récupérer 6 canaux (PWM) alors que nous n'en utilisons que 4 pour le moment. Nous avons laissé la possibilité d'ajouter deux boutons situés sur la télécommande pour ajouter des options au drone.

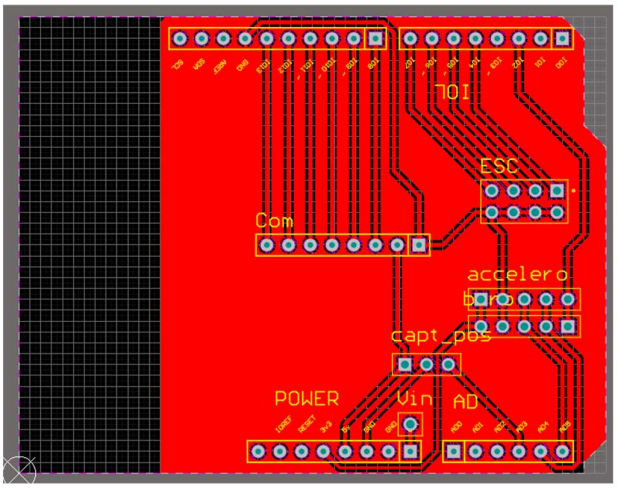


Figure 1 : PCB (shield Arduino) facilitant les branchements

La deuxième partie concerne le contrôle des ESC. Cette partie est ainsi composée de 4 pistes pour la commande des moteurs et du potentiel 0V.

La troisième partie correspond à la communication I2C. Pour cela nous avons besoin du SDA (pin A4) ainsi que du SCL (pin A5) avec une alimentation en 3,3V et 0V. Le bus I2C nous est utile pour le MPU6050 qui est le composant central du projet. Nous avons aussi laissé l'éventualité d'ajouter un baromètre qui fonctionne généralement en I2C. C'est pour cela que le PCB dispose de deux emplacements I2C.

La dernière partie a été ajoutée dans l'optique d'y connecter un capteur de position. Nous avons réalisé ce PCB en début de semestre et nous avons envisagé toutes les possibilités d'améliorations. Nous avions pensé à améliorer l'atterrissage à l'aide d'un tel capteur situé sous le drone. Pour éviter la création de multiples prototypes de PCB nous avons préféré envisager toutes les solutions directement.

Nous avons consacré beaucoup de temps sur cette première partie, car nous avons rencontré des difficultés à la création de la carte ainsi qu'à l'impression comme expliqué précédemment. Malgré ces quelques problèmes, le résultat obtenu correspond à nos attentes et nous avons rapidement constaté la praticité d'un tel composant.

Optimisation de la distribution de puissance

Pendant nos tests avec les hélices, nous avons remarqué des petites baisses de tension aux bornes de nos moteurs. En cherchant, nous avons constaté que ce souci vient des batteries lorsque beaucoup de courant est tiré en même temps. C'est pour cela que nous avons fait le choix d'acheter un PDB afin de mieux réguler la tension et de rendre notre montage plus esthétique.

Figure 2 : PDB pour réguler la tension des ESC

REGULATION DU DRONE

Programme de régulation

Le contrôle du drone repose sur le programme suivant :

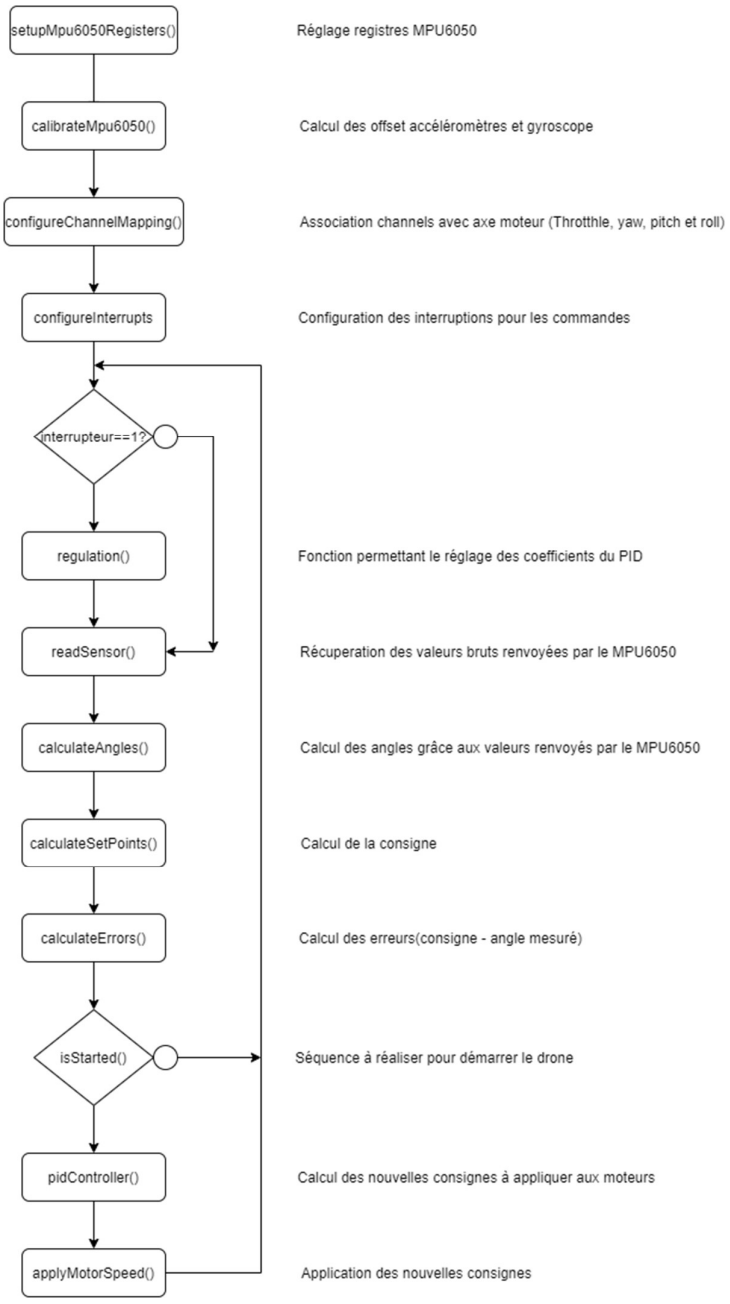


Figure 3 : Algorigramme de la régulation

Ce semestre, côté régulation, nous avons donc dû travailler sur la fonction calculateSetPoints, qui permet de convertir les signaux PWM envoyés par la commande en une consigne en angle,

calculateErrors qui permet de récupérer les erreurs et d'effectuer un premier traitement sur ces dernières tel que la somme des erreurs pour le calcul de la composante intégrale ou bien la variation des erreurs pour le calcul de la composante dérivée, pidController qui permet de calculer les nouvelles consignes à appliquer au moteur et applyMotorSpeed qui se charge d'appliquer les consignes aux escs.

Régulation en suspension

Pour ce test nous nous sommes inspirés d'une méthode trouvée sur internet. Nous avons fixé chaque bras du drone au plafond ainsi qu'au sol en lui laissant quelques centimètres de liberté.

Lors de ce test, nous avons testé la régulation et l'efficacité du PID. Nous avons pour cela fixé les hélices au drone. Les premiers dysfonctionnements sur deux ESC ont commencé à apparaître. Nous avons observé que le PID fonctionnait car la consigne des moteurs montrait bien une compensation mais cette compensation était inadaptée à notre maquette. Nous avons essayé de faire varier quelques valeurs du PID dans la ROM du programme mais sans réussite car un ESC coupait régulièrement au moment où le drone commençait à porter.

Figure 4 : Régulation en suspension

Ce test nous a donc permis de nous rendre compte que nos valeurs de PID n'étaient pas encore parfaites pour réguler convenablement mais que le PID fonctionnait. Ensuite, la variation du PID était lente et pas approprié à un test en temps réel. Nous avons donc pensé à utiliser une breadboard avec 3 potentiomètres et 2 boutons pour modifier les valeurs. Enfin, ce test ne nous assurait pas une grande sécurité c'est pour cela que nous avons décidé de passé sur un autre banc d'essai.

Conception d'un boîtier de contrôle (PID)

Comme pour la conception du PCB, après avoir testé notre circuit pour régler le PID sur la régulation en suspension, nous avons décidé de réaliser un boitier pour obtenir un système de réglage plus maniable et également augmenter la longueur de câble pour limiter les risques liés aux hélices. Nous allons ici entrer plus en détail dans la conception de ce boîtier et expliciter son fonctionnement.

L'objectif de cet outil est de pouvoir modifier les différentes constantes du PID pour réguler le drone en fonctionnement. A l'aide d'un bouton, d'un switch, de trois potentiomètres et de deux résistances, il est possible pour nous de modifier les valeurs de notre PID en interrompant simplement le programme le temps du réglage.

(Schéma électrique)

Le schéma électrique ci-dessus détaille le système, d'un point de vue technique le fonctionnement est assez simple : lorsque l'on veut modifier la valeur de notre PID il suffit de passer l'interrupteur (switch) sur la position ON pour passer l'entrée (pin 12) à l'état haut ce qui interrompra le programme. Ensuite les valeurs sont réglables à l'aide des potentiomètres sur des intervalles suffisamment large ($P \in [0\,;\,2], I \in [0\,;\,0{,}2]\ et\ D \in [0\,;\,39]$). Un fois les valeurs choisies, en appuyant sur le bouton poussoir, l'état du pin 3 passe à 5V et les valeurs sont remplacées dans le programme (en appuyant un deuxième fois on peut afficher ces valeurs dans le moniteur série).

Pour pouvoir utiliser notre système de réglage dans de bonnes circonstances nous avons décidé de le réaliser à l'aide d'un boîtier étanche pas plus grand qu'une main (maniabilité) et avec un câble tressé de 3 mètres de longueur pour éviter la proximité avec le drone lors des tests. Ci-après une photo du rendu :

Figure 5 : Boîtier de réglage du PID

Régulation autour d'un axe

Le premier test de régulation n'ayant pas été très concluant et risqué d'un point de vue sécurité, nous avons décidé de fabriquer un banc d'essai afin de limiter les mouvements du drone sur un seul axe et ainsi rendre la mise en place de la régulation plus simple.

Pour ce faire, nous avons imaginé un système à base de tréteaux et de tube de PVC. Deux tubes de PVC viennent pincer le drone de part et d'autre de la plateforme centrale, et ces deux tubes sont insérés dans deux autres tubes de PVC d'un diamètre légèrement supérieur. En fixant les tubes plus larges aux tréteaux, nous limitons la liberté de mouvement du drone à un seul axe.

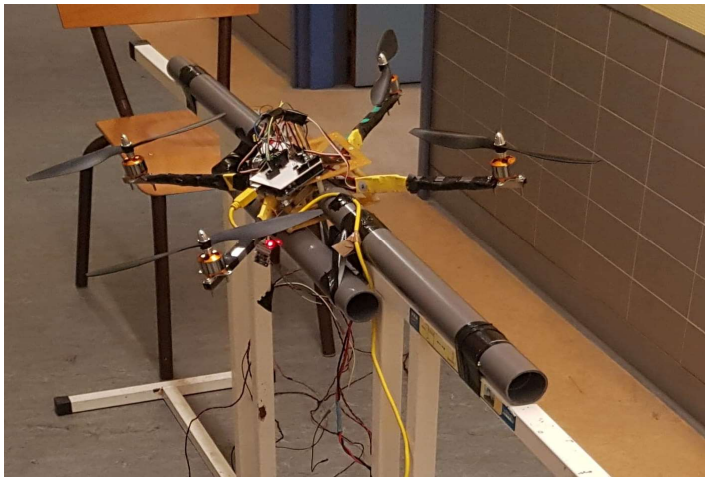


Figure 6 : Régulation autour d'un axe

Si toute la partie software semblait être en place pour finaliser le projet, la partie hardware nous a fait défaut. Nous avons été dans l'obligation de commander de nouveaux ESC. Malheureusement, compte tenu des circonstances nous n'avons pas pu les monter sur le drone et effectuer de nouveaux tests.

Toutefois, même si nous n'avons pas pu finaliser le réglage du PID, les tests effectués malgré le matériel défaillant nous ont tout de même conforté dans l'idée du bon fonctionnement du PID.

En effet, même si la régulation n'était pas bonne, nous avons tout de même observé une compensation ou un ralentissement des moteurs pour atteindre la consigne. Pour effectuer notre test nous avons imposé une consigne constante au drone, ce qui revient à observer sa réponse indicielle.

Visuellement, lors d'un tel test, nous avons observé le résultat suivant :

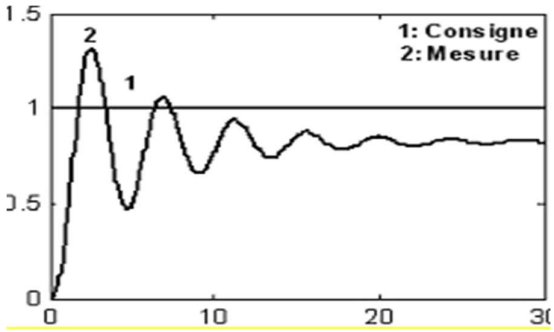


Figure 7 : Consigne envoyée aux moteurs au cours du temps en fonction du PID

Il est donc logique dans un tel cas d'augmenter la composante intégrale afin de corriger l'erreur statique mais aussi d'augmenter la composante dérivée afin de limiter les oscillations. Malheureusement nous n'avons pas pu mener à terme nos tests comme expliqué précédemment.

Afin d'obtenir les meilleurs réglages PID, plutôt que de passer par une méthode "à tâtons", nous avions pour objectif de passer par la méthode de Ziegler–Nichols qui est une démarche plus scientifique même si cela reste une régulation pour expérimentation et non par modélisation.

CREATION APPLICATION MOBILE

La création de l'application mobile n'était initialement pas prévue dans le projet, mais compte tenu des circonstances, il nous a été proposé de réaliser cette dernière.

Pour se faire l'application a été développé sous Android Studio. La partie affichage en XML est relativement simple, elle ne contient que quelques zones de textes pour afficher la valeur des commandes de gaz, deux images pour délimiter la zone de mouvement des joysticks et deux autres images pour représenter les joysticks.

Pour la partie interactive, nous avons utilisés deux listeners, un pour chaque joystick. Nous avons distingué deux cas de figure pour chaque joystick, un premier qui correspond au premier toucher du joystick, à partir de là nous effectuons diverses opérations, tel que le calcul de la référence pour le déplacement joystick. Le deuxième cas de figure correspond au mouvement du joystick. Ce cas est appelé en boucle tant que le joystick n'a pas été relâché. A chaque appel, nous stockons les nouvelles coordonnées du joystick, nous les comparons à celle de l'appel précédent et ainsi nous pouvons avoir une valeur qui représente le déplacement du doigt et donc in fine les nouvelles consignes à appliquer au moteur.

Nous avons aussi rajouté quelques animations pour rendre l'application plus agréable visuellement. (Voir vidéo)

Communication ESP32

Cette partie concerne le projet à compter du 17 mars 2020 (confinement) est se sépare en deux parties.

La première est la prise en main de l'ESP32, pour cela nous avons testé différents modes proposés par l'ESP32. Dans un premier temps, nous avons créé un réseau wifi sur lequel un appareil peut se connecter s'il dispose de clé WAP. Une fois la prise en main de la création du réseau, nous avons voulu commencer la communication avec notre smartphone.

Pour cela nous avons créé un serveur web avec une socket (80) et nous avons créé une page HTML avec deux lignes qui nous permettent d'éteindre et d'allumer la LED 5 sur l'ESP32.

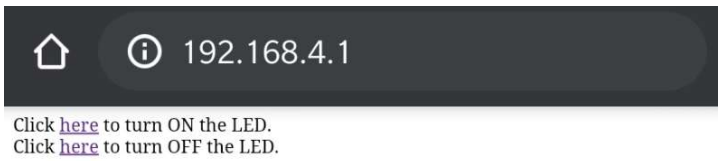


Figure 8 : Page web pour contrôler une LED à l'aide de l'ESP32

Ainsi, pour le clic correspondant à la ligne, la LED va s'allumer ou s'éteindre. Cette partie est bien fonctionnelle et nous a permis de prendre en main l'ESP32, il reste maintenant à faire communiquer l'application avec l'ESP32 et essayer de récupérer les valeurs des commandes.

Dans cette seconde partie, le visuel de l'application mobile est fini. Dans un premier temps, nous nous connectons manuellement au réseau wifi créé précédemment par l'ESP32 puis nous essayons d'envoyer des trames sur la socket créée sur le réseau mais nous ne récupérons aucune information.

Dans un second temps, nous avons utilisé la connexion de l'ESP32 à un réseau wifi existant. Nous créons donc une socket 4000 et nous écoutons la réception TCP sans émettre.

En ce qui concerne l'application mobile, nous avons modifié des fonctions pour permettre l'envoi de messages en TCP. Nous vérifions donc que la socket est bien ouverte en envoyant une trame TCP sur l'adresse IP et le port de notre ESP et nous recevons bien. Il reste donc à émettre correctement avec l'application mobile.

Nous avons créé les fonctions JAVA pour toute la communication TCP c'est-à-dire :

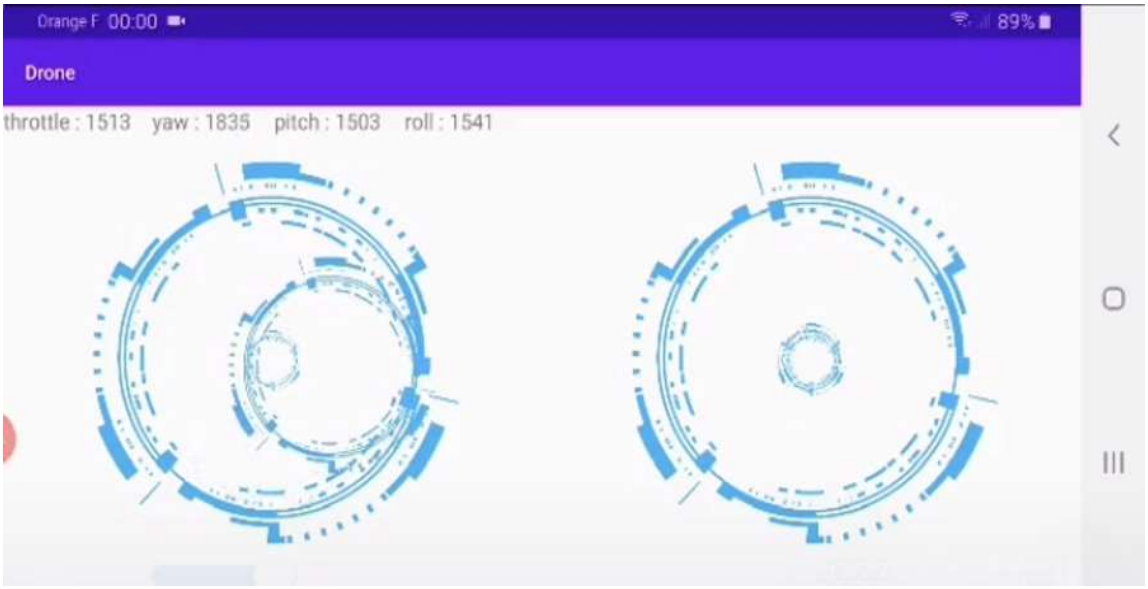
- Démarrer la connexion
- Arrêter la connexion
- Connecter la socket
- Déconnecter la socket
- Émission et réception de message

La simulation de la connexion sur l'ordinateur marche bien mais la mise en place de la communication TCP sur l'application mobile a été périlleuse, ce problème va être expliqué dans la partie liée aux problèmes.

Nous recevons bien nos 4 valeurs de nos 2 joysticks, ainsi notre application Android répond aux exigences, nous avons deux joysticks et nous recevons les valeurs souhaitées.

https://github.com/richardsimonin/projet_p4_drone.git

Le git contient le code, de notre application mobile. Il y a deux classes pour la gestion de la communication qui permet une gestion asynchrone en multi-threadé.



Des captures pour montrer la fonctionnalité de l'application.

```
throttle:1520,pitch:1626,roll:1626,yaw:1439.
throttle:1534,pitch:1638,roll:1631,yaw:1397.
throttle:1549,pitch:1660,roll:1639,yaw:1365.
throttle:1558,pitch:1695,roll:1648,yaw:1341.
throttle:1564,pitch:1712,roll:1650,yaw:1327.
throttle:1569,pitch:1719,roll:1648,yaw:1316.
throttle:1573,pitch:1721,roll:1645,yaw:1307.
throttle:1574,pitch:1721,roll:1638,yaw:1301.
throttle:1575,pitch:1718,roll:1624,yaw:1297.
throttle:1575,pitch:1708,roll:1604,yaw:1294.
throttle:1572,pitch:1695,roll:1574,yaw:1292.
throttle:1551,pitch:1684,roll:1545,yaw:1286.
throttle:1520,pitch:1668,roll:1501,yaw:1282.
throttle:1467,pitch:1639,roll:1453,yaw:1291.
throttle:1429,pitch:1595,roll:1405,yaw:1308.
throttle:1373,pitch:1563,roll:1361,yaw:1330.
throttle:1336,pitch:1529,roll:1331,yaw:1346.
throttle:1305,pitch:1505,roll:1311,yaw:1362.
```

Figure 10 : capture terminale ESP32 durant la communication avec l'application Android

Ainsi nous observons bien une variation des valeurs de nos joysticks sur le terminal de l'ESP32.

PROBLEMES RENCONTRES

Problème lié à l'ATmega328P

Lors de nos différents tests nous avons rencontré un problème majeur qui était l'arrêt du microcontrôleur ATmega328P. Il nous a fallu débuguer le programme pour résoudre ce problème.

Dans un premier temps nous avons suspecté un problème venant de l'ISR. En effet nous avions pensé que les routines d'interruptions étaient plus grandes que la fréquence des interruptions. Nous avons raccourci au maximum nos temps d'interruption pour régler ce problème mais malheureusement cela n'a pas permis de résoudre le plantage du microcontrôleur. Nous avons donc tu continuer nos investigations pour trouver l'origine du problème.

Dans un deuxième temps, nous avons modifier le programme en ajoutant l'allumage de LED dans différentes parties du programme afin de trouver la fonction faisant planter le microcontrôleur. Après plusieurs heures de débogage nous n'avons rien trouvé de cohérent au niveau de la programmation, le programme s'arrêtant à des endroits totalement aléatoires.

Nous avons ensuite utilisé un multimètre pour vérifier toutes les tensions et essayé de trouver une tension anormale mais là encore aucun problème.

En revanche lors de test de régulation nous nous sommes rendu compte que certains mouvements brusques avaient tendance à faire apparaître notre plantage. Après enquête nous avons fini par trouver un câble défectueux qui entrainait la coupure du MPU6050, de ce fait nous supposons que le MPU6050 ne renvoyant aucune valeur, des divisions par zéro devait se produire dans le programme entraînant un plantage du microcontrôleur. Ce possible problème nous avait d'ailleurs été suggéré par nos encadrants.

Nous avons donc changé le câble pour ne plus avoir ce problème et ainsi pouvoir réguler sans interruption mais aussi vérifié de nombreux autre câbles avec des tests de continuité par prévention.

Problème lié aux ESC

Pendant le premier test, suspendu, nous avons détecté un disfonctionnement d'un ESC pendant la portance du drone et un deuxième ESC posait un problème lorsque la consigne était trop grande.

Nous avons voulu savoir d'où venaient ces deux problèmes, et nous avons donc utilisé un oscilloscope afin d'observer les courants lors des différentes phases.

Ainsi nous avons analysé les courants de commande des ESC dans la phase de démarrage, accélération et enfin nous avons envoyé une consigne fixe.

Nous avons constaté que pendant la phase de démarrage un pic de courant est observable, ce qui est normal, et nous ne détectons aucune anomalie.

Pendant la phase d'accélération, un ESC « décroche » alors que son courant de commande et sa tension d'alimentation sont correctes, nous suspectons un disfonctionnement interne au composant.

Enfin lors de la phase où la consigne est constante, nous rencontrons un souci uniquement lorsque les hélices sont montées sur les moteurs. Quand le drone commence à porter, un moteur coupe et ne redémarre plus. Pour cela nous avons essayé de tout reprendre depuis le début, c'est-à-dire de réinitialiser les ESC. Nous avons ainsi modifié notre programme d'initialisation de base pour mieux comprendre les différents modes et essayer ainsi de trouver une erreur. Malheureusement, nous n'avons pas réussi à trouver d'où venait cette erreur, c'est pourquoi nous avons fait le choix de commander 4 nouveaux ESC.

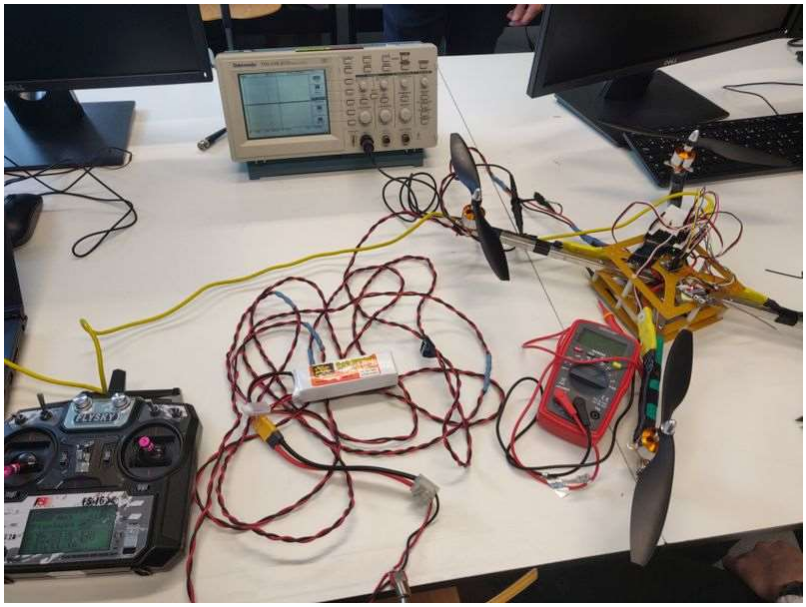


Figure 11 : Test des valeurs de courant reçues par les ESC

Problème lié à la programmation mobile

Lors de la création de l'application mobile, il nous fallait communiquer avec un esp32 en TCP, pour l'échange d'information. Sur le logiciel Android studio dans un premier temps nous pouvons simuler sur un téléphone simuler puis le transférer sur un téléphone physique afin de tester l'application dans le cas réel. Pour la communication les fonctions fonctionnaient sur la simulation mais une fois les tests sur le téléphone physique l'application crash à chaque fois. Après du débogage, la source du problème est identifiée, la connexion à la socket. Après de la documentation sur Android, il est possible de faire du TCP uniquement en multi-threadé. Il a fallu ainsi changer tout le code et créer deux classes JAVA, les fonctions TCP et les threads. Après des recherches sur les possibilités en JAVA nous sommes parvenus à la bonne exécution des fonctions TCP sur l'application Android.

CONCLUSION

Au semestre 6, nous étions parvenus à faire voler un drone à l'aide d'un contrôleur de vol acheté dans le commerce. Pour cette année d'IMA4 et ce semestre 8 en particulier, nous ne sommes pas parvenus totalement à nos objectifs bien que nous en fussions proches.

Les problèmes techniques liés aux ESC et aux moteurs ainsi que le manque de temps dû à la fermeture des établissements d'enseignement nous ont empêchés de finaliser notre régulation.

Cependant, la partie commande et communication fonctionne correctement et bien que la régulation ne soit pas finalisée, suites aux derniers tests que nous avons pu effectuer, nous sommes très optimistes concernant le fonctionnement final du drone.

De plus, si le confinement nous a empêché de finaliser le projet, il a tout de même favorisé le développement de nouveaux axes tel que la conception d'une application permettant de piloter le drone depuis un smartphone. Actuellement nous disposons de toutes les pièces du puzzle pour y parvenir et à l'instar de la régulation, nous comptons finaliser tout cela lors de la reprise des cours.