REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE

MINISTÈRE DE L'ENSEIGNEMENT SUPÉRIEUR ET DE LA RECHERCHE SCIENTIFIQUE

Université Abou Bekr Belkaïd de Tlemcen

Faculté de Technologie

N° attribué par la bibliothèque

MEMOIRE DE FIN D'ETUDE POUR L'OBTENTION DU DIPLOME DE MASTER ACADEMIQUE

Spécialité : « Automatique »
Option : « Automatique et informatique industrielle »

préparé au Département de Génie Électrique et Électronique (GEE) et au Laboratoire d'Automatique de Tlemcen (LAT)

et présenté par

SEDINI Chahrazed et CHERIGUI Nasre-Eddine

Intitulé du mémoire

Conception et commande d'un quadrotor UAV à base d'Arduino

sous la direction du Dr A.CHOUKCHOU-BRAHAM

soutenu publiquement le 24 Juin 2019 devant la commission d'examen composée de :

B.CHERKI	Président	Professeur	U.A.B - Tlemcen
A.CHOUKCHOU-BRAHAM	Encadrante	Maître de Conférences	U.A.B - Tlemcen
A.HADJ ABDELKADER	Examinateur	Professeur	U.A.B - Tlemcen
B.BENYAHIA	Examinateur	Maître de Conférences	U.A.B - Tlemcen

Année universitaire 2018 - 2019

Dédicaces

Nous dédions ce travail :

- à Nos très chers parents
- à Nos frères et Soeurs
- à Nos familles
- à Nos amis

SEDINI Chahrazed et CHERIGUI Nasre-Eddine
Tlemcen, le 24 juin 2019

Remerciements

Nous tenons à exprimer notre profonde gratitude à ALLAH Azza Wa Jal pour nous avoir donné le souffle de vie, la force, la santé et l'intelligence nécessaires pour accomplir ce travail.

Nous tenons sincèrement à remercier les personnes qui grâce à eux, nous n'aurions pas pu être là, nos parents. Qui nous ont encouragés tout au long de ces cinq années, Ainsi que toute ma famille.

Ce document présente les travaux effectués dans le cadre de notre projet de fin d'étude de Master au Département de Génie Électrique et Électronique de la Faculté de Technologie de l'Université Abou Bekr Belkaïd de Tlemcen.

Ce projet est le résultat de cinq mois de travail, et de recherche sous la supervision du notre encadrante Mme Amal CHOUKCHOU-BRAHAM, que nous remercions profondément pour ses instructions et qui nous a aidé à progresser dans notre réflexion grâce à ses conseils, son esprit critique et son soutien tout au long de la réalisation de ce mémoire..... Ainsi que pour nous avoir motivé pour travailler et poursuivre le projet.

Nous voudrons aussi remercier également les membres du jury Messieurs B.CHERKI, A.HADJ ABDELKADER et B.BENYAHIA qui nous rendent honneur en acceptant d'examiner notre modeste travail. Leurs présences, l'intérêt qu'ils portent à notre projet, les conseils et remarques qu'ils nous auront apportés ne pourrant que enrichir ce travail.

Sans oublier de fortement remercier Mr Ismat MESLOULI qui nous a aidé à comprendre les différentes notions durant notre projet de fin d'étude.

Nous remerciement vont aussi à tous nos collègues de l'option Automatique et Informatique Industriel.

Enfin, nous remercions tous ceux qui ont contribué de près ou de loin à la réalisation de ce travail.

Table des matières

Table des figures

Notations et abréviations

Les principales notations et abréviations utilisées dans ce mémoire sont explicitées ci-dessous, sous leur forme la plus couramment employée dans le domaine du génie électrique.

Symbole	**Signification**
P	Pesanteur
m	La masse
g	Accélération de la pesanteur (9,81 m/s^2)
F	Force de poussée
F_g	Force de gravité
F_f	Force de portance
V	Vitesse linéaire
Ω	Vitesse angulaire dans le repère fixe
ω	Vitesse angulaire
T_h	Traînée de l'hélice
F_t	Traînée selon les axes
$K_f t$	Coefficient de traînée
d	Coefficient de drag
b	Coefficient de portance
l	Longueur du segment

Symbole	**Signification**
M_f	Matrice de moment de portance
M_x, M_y, M_z	Moment de rotation selon les axes X,Y et Z
M_a	Moment de frottement aérodynamique
$K_f a$	Coefficient de frottement aérodynamique
J	Inertie du système
J_r	Inertie des rotors
$\wedge$	Produit vectoriel
$M_g h$	Moment gyroscopique selon les hélices
$M_g m$	Moment gyroscopique selon les mouvements du drone
J	Inertie du système
ϕ	Angle de rotation du couple créé selon X
θ	Angle de rotation du couple créé selon Y
ψ	Angle de rotation du couple créé selon Z
R	Matrice de rotation
Rot_x, Rot_y, Rot_z	Matrice de rotation autour de X, Y et Z
c, s, tan, atan	Cos, sin, tangente, arc-tangente
BEC	Battery Eliminator circuit
PID	Proportionnel Integral Dérivée
DDL	degré de liberté
UAV	Unmanned Aerial Vehicles
VTOL	Vertical Take OFF and Landing

Introduction générale

Durant les dix dernières années, un intérêt croissant est porté aux engins volants sans pilote humain â bord que l'on appelle drone. Un drone ou Unmanned Aerial Vehicle (UAV) est un aéronef sans passager ni pilote, capable de voler de façon autonome ou pilotés à distance. Le mot -drone- est extrait d'un terme anglais qui signifie -faux-bourdon-. En français, le terme est employé pour désigner des véhicules aériens, terrestres, de surface ou sous-marins. Ces appareils sont capables de porter différents types de charges : caméras, radars, missiles, équipements de communication, etc. Ces charges les rendent capables d'accomplir des tâches spécifiques, civiles et militaires.

Le véhicule aérien sans pilote connaît un essor croissant ces dernières années grâce à leurs différents domaines d'application, ces appareils ont notamment l'avantage de moins exposer le personnel au danger tout en couvrant une large gamme de missions. Au début, les applications du drone ont été orientées uniquement vers le domaine militaire. Aujourd'hui, ils sont aussi utilisés dans divers domaines tels que la recherche, le sauvetage, la cartographie, etc...

De plus, un quadrirotor se présente en quelque sorte comme un hélicoptère miniature à quatre moteurs. C'est un système à six degrés de liberté, qui se pilote généralement avec des consignes en puissance, et en angle (tangage, roulis et lacet). Le pilotage d'un tel engin s'avère difficile, d'autant plus que sa structure est très légère, ce qui le rend très sensible aux perturbations extérieures. Le but de notre projet est donc de stabiliser le quadrirotor, et de s'assurer que son comportement est conforme aux consignes que l'on lui donne.

L'objectif de ce travail et de faire la conception et la commande à base de PID d'un drone quadrotor en utilisant une carte de programmation Arduino uno. Ce travail s'inscrit dans le cadre de la continuation des travaux précédents menés au sein de l'équipe -commande- du laboratoire d'Automatique de Tlemcen.
Notre travail consiste à améliorer la partie identification des paramètres pour rapprocher la pratique de la simulation. A l'inverse du mémoire de master précédent où la carte de vol performante Pixhawk a été utilisée, nous avons choisi de travailler avec une carte très simple qui permet non seulement de réduire les coûts mais aussi

de faciliter l'accès à la partie programmation de lois de commande en vue de l'implémentation de d'autres lois de commande développées dans cette équipe.

Ce mémoire s'articule autour de quatre chapitres :

Dans le premier chapitre, nous présenton un état de l'art sur les drones en général et sur les quadrirotors en particulier où nous passons en revue leurs différents domaines d'application et leur fonctionnement.

Le deuxième chapitre est consacré à la modélisation dynamique du quadrirotor, d'abord nous faisons une description de ce système, de point de vue structure générale et principe de vol. Puis, nous mettons en évidence la modélisation dynamique du quadrirotor à travers le formalisme de Newton-Euler. Enfin, nous donnons le modèle d'état.

Dans le troisième chapitre, nous entamons en premier l'identification des paramètres de quadrirotor. Après nous synthétisons des régulateurs à base de PID pour le contr´le des différents mouvements tout en testant le modèle de simulation réalisé sous MATLAB, ensuite nous discutons les différents résultats obtenus.

Dans le quatrième chapitre, nous présentons les composants que nous avons utilisé pour la conception de notre quadrirotor. Nous passons ensuite au montage mécanique et électronique moyennent ces composants. La partie intelligente est implémentée à travers une carte Arduino uno sur laquelle nous programmons 3 PID pour le contrôle des angles d'Euler, l'environnement entrées-sorties et la gestion des capteurs. Le drone est augmanté d'élements de transmission de petite portée pour permettre la récuperation des données de vol en vue de les tracer. Des tests de vol en indoor et outdoor seront effectués pour évaluer la conception et la commande de ce drone quadrotor.

Drones : état de l'art

Sommaire

Résumé

Le premier chapitre consiste à donner un bref état de l'art sur les drones, leur historie, et leurs différentes classification.Parmi ces classifications il existe les drones de type quadrirotor que nous avons aussi définit dans ce chapitre, et nous avons également citer leur avantage et domaines d'application.

I.1 Introduction

Un drone ou Unmanned Aerial Vehicle (UAV) est un aéronef sans passager ni pilote, capable de voler de façon autonome ou être contrôlé à distance depuis le sol. Le mot « drone » est extrait d'un terme anglais qui signifie « faux-bourdon ». En français, le terme est employé pour désigner des véhicules aériens, terrestres, de surface ou sous-marins.

La taille de ces véhicules aérien peut aller de quelques centimètres pour les modèles miniatures à plusieurs mètres pour les drones spécialisés (surveillance, renseignement, combat, loisirs). Leur autonomie en vol va de quelques minutes à plus de 40 heures pour les drones de longue endurance.

I.2 Historique

La conception des drones a commencé pendant la Première Guerre mondiale : des prototypes ont était proposés, avec des tentatives de « torpilles aériennes » (telle le Kettering Bug) télécommandées par télégraphie sans fil et embarquant un gyroscope, mais qui n'a jamais été opérationnel sur le terrain. En 1916, au Royaume-Uni, fut conçu l'Aerial Target, un projet d'avion-cible, par l'ingénieur Archibald Low.

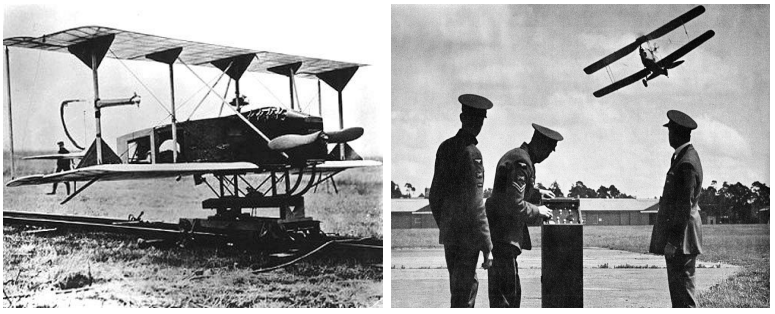

FIGURE I.1 – Kettering Bug(à gauche) Aerial Target(à droite)

Le 2 juillet 1917 en France le pilote Max Boucher, fait voler un avion Voisin « sans l'intervention de l'homme » sur 1 km. Au début de l'année 1918, un projet d'« avions sans pilotes » a été lancé. Après l'amélioration du système de pilotage automatique, l'avion Voisin BN3 a volé pendant 51 min sur un parcours de 100 km.

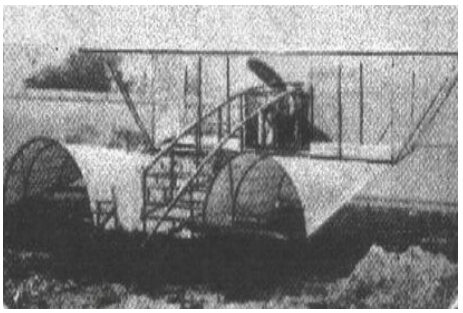

FIGURE I.2 – avion Voisin BN3

En 1941, l'US Navy passa commande de plus de mille exemplaires d'un nouveau modèle baptisé Target Drone Denny 1 (TDD-1). En 1938 l'armée allemande développa des recherches sur des vecteurs guidés à distance et prenant la forme de bombes planantes anti-navires, de bombes antichar radioguidées et surtout de véhicules à chenilles filoguidés.

FIGURE I.3 – Drone Denny 1 (TDD-1)

Dans les années 1990, la doctrine de la guerre « zéro mort » conduit à développer les projets de drones armés à travers le monde mais la toute première utilisation de ceux-ci a eu lieu durant la guerre Iran-Irak où l'Iran a déployé un drone armé de six RPG-7.

FIGURE I.4 – Predator-and-hellfire

Dans les années 2000, le drone est de tous les conflits et opérations de maintien de la paix, dont au Kosovo ou au Tchad, lors des attaques aériennes américaines au Pakistan ou contre la piraterie maritime, par les Américains qui l'ont introduit en

2009 [15].

I.3 Classification des drones

I.3.1 Selon la taille

Il n'est pas facile de classer les systèmes aériens, car il existe plusieurs catégories de drones aériens partant du nano-drone de quelques grammes jusqu'au drone lourd capable d'effectuer des missions de plus de 24 heures à plusieurs milliers de kilomètres de sa base.

La figure ci dessous illustre les différents types de drones selon les principaux critères : la taille et les performances (autonomie en vol, portée ou altitude d'opération). Trois grandes familles de drones se détachent alors en fonction du type de mission :

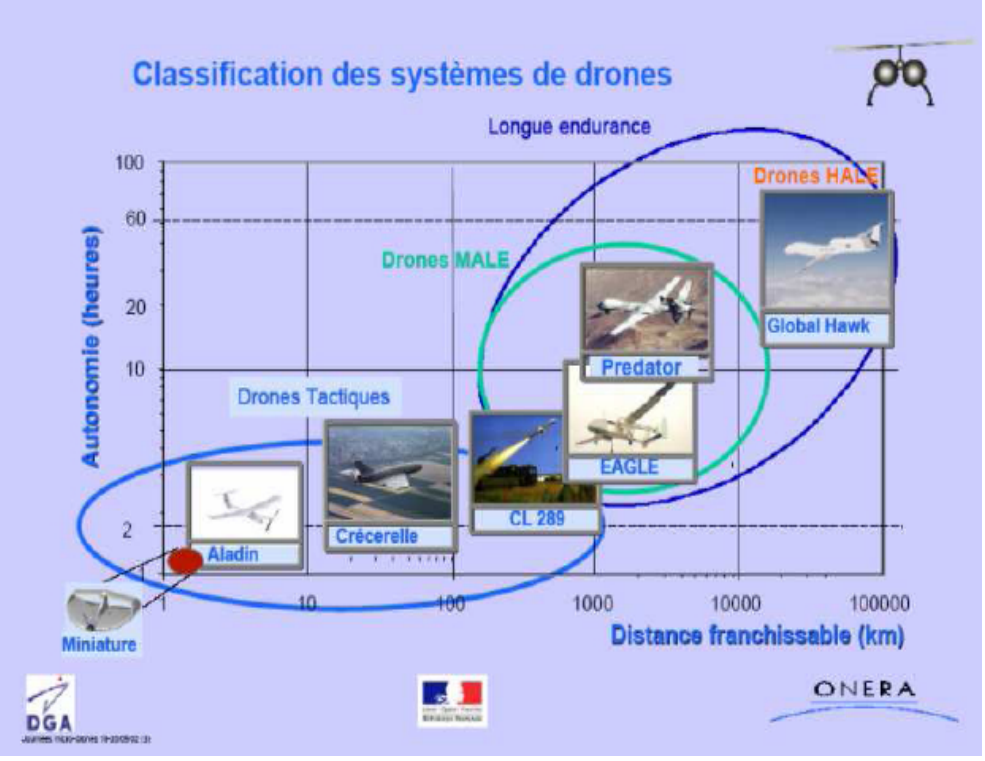

FIGURE I.5 – Classification des drones

-Les drones de Moyenne Altitude et Longue Endurance (MALE) .
-Les drones de Haute Altitude et Longues Endurances (HALE) .
-Les drones tactiques.

MALE (Moyenne Altitude Longue Endurance) : Caractérisés par leurs longues durées de vol à moyenne altitude opérationnelle, ayants une grande autonomie. Tels que le Eagle 1 (EADS), les drones **MALE** peuvent voler à des altitudes

comprises entre 5 km et 12 km pour un rayon d'action allant jusqu'à 1000 km.

FIGURE I.6 – Eagle 1 (EADS)

HALE (Haute Altitude Longue Endurance) : Ce sont des drones à grande taille, le plus souvent à voilure fixe. Caractérisés par leurs très longue durée en vol et collecte des informations sur de très longues périodes (12 à 48 heures) à haute altitude. Tels que le Global Hawk (Northrop Gumman), de la taille d'un avion de ligne et volants à des altitudes pouvant atteindre 20 km pour un rayon d'action de plusieurs milliers de kilomètres.

FIGURE I.7 – Global Hawk

Les drones tactiques, tels que le Sperwer (SAGEM) ou l'Aerostar (Aeronautics Défense Systèmes), utilisés pour des missions de reconnaissance ou de supervision du champ de bataille et volant à une altitude comprise entre 200 m et 5 km, pour un rayon d'action de 30 à 500 km [1].

FIGURE I.8 – Sperwer (SAGEM)

Mini drones MAV (Mini Air Vehicule) : Avec un temps de vol de quelques heures et des dimensions de l'ordre du mètre, les minidrones peuvent voler jusqu'à une altitude de 300 mètres, avec un diamètre d'environ 30 kilomètres en emportant une charge utile très légère. Ce type d'appareil est en général propulsé électriquement. La figure I.9 présente le mini drone Hovereye [Pflimlin2007], développé par Bertin Technologies.

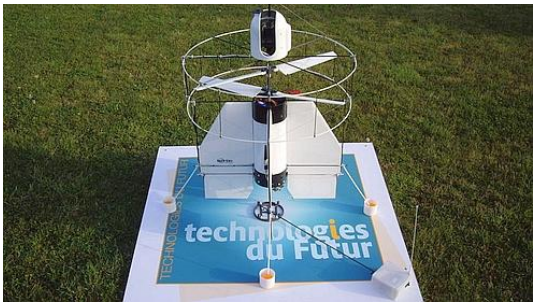

FIGURE I.9 – Hovereye [Pflimlin2007]

Micro-drones : Un MAV (Micro Air Vehicle) est un drone de taille réduite qui représentent la 3^{eme} génération de drones. De plus, ces petits véhicules aériens peuvent accomplir des tâches dont des engins plus gros sont incapables.

FIGURE I.10 – Micro drone

I.3.2 Selon la propulsion

Le type de propulsion fournit une autre catégorie de classification, dont on peut distinguer trois familles.

Drones à voilures fixes : Ce type de véhicules utilisent des ailes fixes pour leurs déplacement, donc ils ont besoin d'une piste de décollage et atterrissage. Ils peuvent être :

- Plus lourds que l'air : type avion.
- Plus légers que l'air : type dirigeable qui utilisent de l'hélium pour générer une poussée verticale et des rotors pour générer des couples.

FIGURE I.11 – Drone à voilures fixes

Drones à ailes battantes : Inspirés par les insectes, la construction de cette famille de drones consiste au battements des ailes qui permettent de faire des vols stationnaires et d'imiter les trajectoires des insectes.

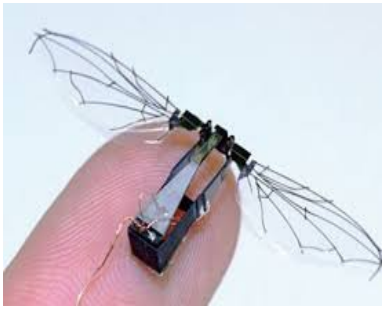

FIGURE I.12 – Drone à ailes battantes

Drones à voilures tournantes : Les drones à voilures tournantes sont caractérisés par leurs décollage et atterrissage verticaux (ADAV, en anglais Vertical Take-off and Landing aircraft ou VTOL), donc ils n'ont pas besoin de piste pour décoller ou atterrir.
Ils sont utilisés dans plusieurs applications telles que la surveillance et la récolte de données grâce à leur vol stationnaire à basse vitesse et à faible altitude.

Ce type de drone, aussi diffère selon le nombre de ses propulsions.

I.3.3 Selon le nombre de rotors

Les types de drones les plus courants dans cette catégorie sont :

- Mono-rotors : L'exemple le plus commun dans cette configuration est l'avion appelés 3D, qui a un seul moteur comme actionneur principal, et qui doit être suffisamment puissant pour décoller verticalement, et des ailerons d'une grande surface qui assurent des couples de commande suffisamment grands afin de piloter l'appareil facilement.

FIGURE I.13 – Avion 3D

-Birotors : Ce type de configuration inclue deux type de drones, ceux avec un ou deux plateaux cycliques comme l'hélicoptère classique et ceux avec des pales à pas fixe.
Le cas de l'hélicoptère nous avons un rotor principal et un rotor de queue.

Il existe aussi l'hélicoptère en tandem qui possède deux rotors qui tournent en contre sens mais dans des axes différents.

FIGURE I.14 – Hélicoptère classique(à gauche) Hélicoptère en tandem(à droite)

Nous pouvons aussi distinguer la configuration à deux rotors, coaxiales, qui participent les deux à la poussée. En tournant dans des sens opposés les effets de couple générés par la rotation des rotors, sont annulés.

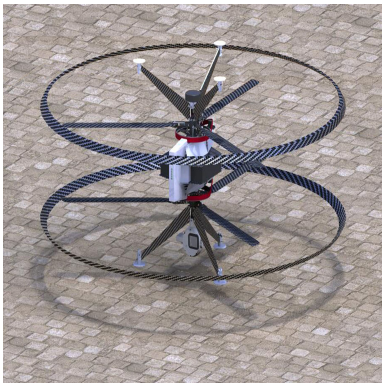

FIGURE I.15 – Drone à deux hélices co-axiale

-Multirotors : Ce type de véhicules aériens se compose de plusieurs rotors avec un sens de rotation inversé deux à deux pour compenser le couple de réaction, le changement des vitesses des moteurs convenablement assure les déplacements du véhicule.

FIGURE I.16 – Drones multirotors

I.4 Quadriotor

I.4.1 Définition

Quadcopter, aussi appelé hélicoptère quadrirotor, est un drone multirotor qui est soulevé et propulsé par quatre moteurs. Les quad-coptères sont classés en giravions, par opposition aux drones à voilure fixe, car leur portance est générée par un ensemble de rotors.

Les quadrotors utilisent généralement deux paires d'hélices à pas fixe identiques, deux dans le sens des aiguilles d'une montre (CW) et deux dans le sens inverse (CCW). En modifiant la vitesse de chaque rotor, il est possible de générer spécifiquement une poussée totale souhaitée, localiser le centre de poussée latéralement et longitudinalement, et pour créer un couple total souhaité, ou force de rotation.

I.4.2 Historique

Le gyroplane Breguet-Richet est le premier quadrirotor qui a vu le jour, développé par la société Breguet en 1907, ce véhicule n'a pas pu décoller qu'à 60 cm du sol et quatre hommes maintenaient sa structure.

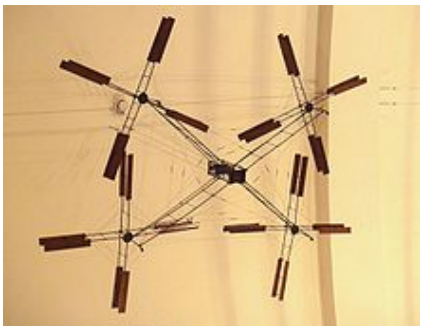

FIGURE I.17 – Gyroplane Breguet-Richet

En janvier 1921, l'Américain d'origine russe George de Bothezat est le premier qui a pu faire voler un appareil quadrirotor. En décembre 1922 il réalise un vol de 1 min 42 s à 1,8 m du sol. Le 19 janvier 1923, l'appareil emporte deux personnes à 1,2 m du sol.

FIGURE I.18 – Quad-coptère de Bothezat

En 1924, le quadrirotor du Français Étienne OEhmichen monte à plus de dix mètres d'altitude et effectue une boucle complète sur un kilomètre en sept minutes et quarante secondes . En plus des quatre rotors de sustentation, le type n^o2 d'OEhmichen était doté de huit hélices de direction.

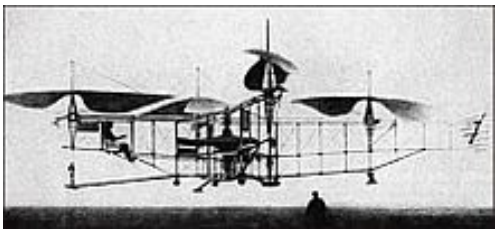

FIGURE I.19 – Appareille $n^{\circ 2}$ d'OEhmichen

En janvier 2016, la société chinoise Ehlang basé à Guangzhou présente un drone quadrirotor capable de transporter une personne jusqu'à 100 kg, à 500 mètres d'altitude.

Aujourd'hui, de petits véhicules aériens sans pilote ont été utilisés pour de nombreuses applications. Le besoin d'aéronefs dotés d'une plus grande maniabilité et

d'une plus grande capacité de vol stationnaire a entraîné une augmentation des recherches sur les quadricoptères.

Certains programmes actuels incluent :

-Le concept Bell Boeing Quad TiltRotor de Bell va plus loin dans le concept du quadricoptère fixe en le combinant avec le concept du rotor pendulaire pour un transport militaire de taille C-130 proposé.

-AeroQuad et ArduCopter sont des projets de logiciel et de matériel open-source basés sur Arduino pour la construction de quadricoptères.

-Parrot AR.Drone est un petit quadricoptère radiocommandé, équipé de caméras, construit par Parrot SA, conçu pour être contrôlé par un smartphone ou une tablette.

FIGURE I.20 – Prototype volant du Parrot AR.Drone (à gauche)
Décollage de Parrot AR.Drone 2.0, Nevada, 2012 (à droite)

-Nixie est un petit drone équipé d'une caméra qui peut être porté en poignet [16].

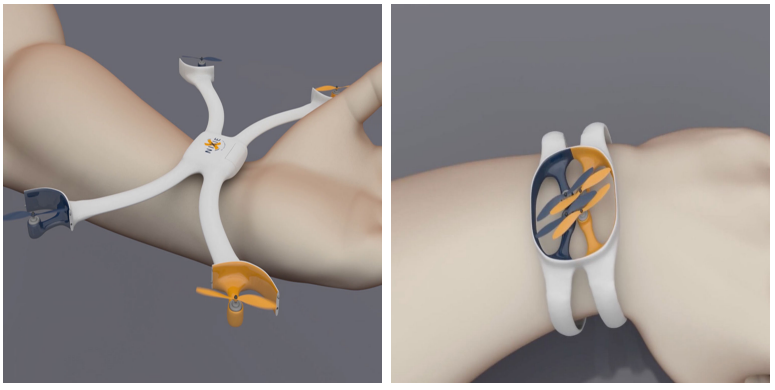

FIGURE I.21 – Drone Nixie

I.4.3 Avantages

La conception du quadrirotor offre de réels avantages par rapport à d'autres configurations [4] :

- Leurs tailles réduites et leur man ?uvrabilité permettent de se déplacer dans des environnements fermés (Indoor) ou ouverts (Outdoor) et en évitant les obstacles contrairement au autre véhicules aériens.
- La simplicité de sa mécanique facilite sa maintenance.
- Aucun embrayage n'est exigé entre le moteur et le rotor et aucune exigence n'est donnée sur l'angle d'attaque des rotors.
- Quatre petits rotors remplacent le grand rotor de l'hélicoptère ce qui réduit énormément l'énergie cinétique stockée et minimise les dégâts en cas d'accidents.
- Son décollage et atterrissage verticaux.
- Cette configuration est commandée en variant seulement la vitesse de rotation des quatre moteurs.
- Leur capacité de portance à cause de la présence de quatre rotors au lieu d'un qui peut être augmenté en rallongeant les pales d'un rotor ou en augmentant leur nombre.
- Sa dynamique est plus faible que celle de l'hélicoptère ce qui ne nécessite pas un temps de réaction rapide.

I.4.4 Applications

-Plateforme de recherche : Les quad-coptères sont un outil utile pour les chercheurs universitaires qui souhaitent tester et évaluer de nouvelles idées dans différents domaines.

-Militaire et maintien de l'ordre : Les véhicules aériens type quadrotor sont utilisés pour la surveillance et la reconnaissance par les forces armées et les forces de l'ordre, ainsi que pour les missions de recherche et de sauvetage en milieu urbain.

-Photographie : Les quad-coptères conviennent à cet emploi à cause de leur nature autonome et de leurs économies considérables.

-Journalisme : En 2014, le Guardian a annoncé que les principaux médias avaient commencé à déployé des efforts considérables pour explorer l'utilisation de drones afin de rapporter et de vérifier des informations sur des événements tels que les inondations, les manifestations et les guerres.

Certains médias et journaux utilisent des drones pour capturer des photographies de célébrités.

-Livraison par drone : En 2013, à l'aide d'un quadrotor Microdrones md4-1000, des colis ont été expédiés d'une pharmacie. C'était la première livraison de colis civil par drones[14].

I.5 Conclusion

Ce chapitre a permis d'avoir une vison générale sur les drones, leurs histoire et différents types. Nous avons également décrit les drones de type quadrotor, leur apparition et leurs avantages et applications.
Dans les chapitres suivants nous allons essayer d'établir un modèle dynamique décrivant les mouvements de notre véhicule aérien, et synthétiser une loi pour le commander.

modélisation dynamique du quadrirotor

Sommaire

Résumé

Dans ce chapitre nous allons définir le principe de fonctionnement d'un drone quadrirotor et décrire ses différents mouvements, pour pouvoir établir son modèle dynamique.

II.1 Introduction

La modélisation consiste à utiliser des techniques permettant de disposer d'une représentation mathématique d'un système, plus cette présentation est détallée plus elle traduit le comportement réel de ce système.

Le quad-coptère fait partie des systèmes volants les plus complexes à cause des phénomènes physiques qui influent sur sa dynamique. Afin de réaliser un contrôleur de vol robuste, nous devons d'abord comprendre les mouvements du système et sa dynamique. Cette compréhension est nécessaire pour assurer que les simulations de l'engin dépeindront un comportement aussi proche que possible de la réalité.

II.2 Principe de fonctionnement

Les quatre moteurs d'un quad-copter sont généralement placés aux extrémités du châssis, et l'électronique de contrôle est habituellement placée au centre.

Pour éviter au véhicule de tourner autour de son axe z (lacet), il est nécessaire que deux hélices tournent dans un sens, et les deux autres dans le sens contraire. Pour pouvoir diriger l'engin, il faut que chaque couple d'hélices tourne dans le même sens, soit placé aux extrémités opposées d'une branche du châssis.

Le fonctionnement d'un quadrotor est assez particulier. En variant astucieusement la puissance des moteurs, nous sommes capable de le faire monter/descendre, de l'incliner à gauche/ droite (roulis) ou en avant/arrière (tangage) ou de le faire pivoter sur son axe vertical (lacet), le quad-copter a six degrés de liberté, trois mouvements de rotation et trois mouvements de translation, ces six DDL sont commandés à partir de quatre consignes seulement.
Ainsi le drone qudrirotor est un système sous actionné (le nombre des entrées est inférieur au nombre des sorties).

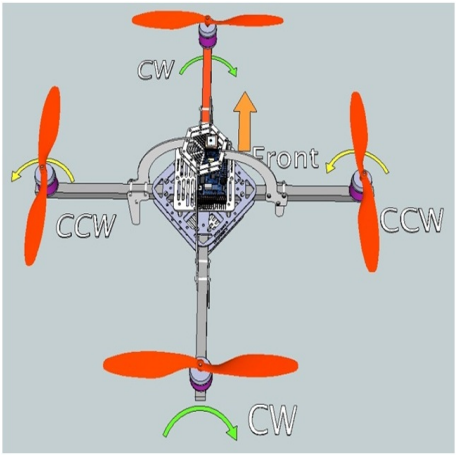

FIGURE II.22 – Configuration d'un drone quad-copter

II.3 Les mouvements du quadcopter

La variation de vitesse de chaque rotor va produire une poussée, et par conséquent réaliser un mouvement. Le quad-copter s'incline vers la direction du rotor plus lent, et effectue alors une translation le long de cet axe. Par conséquent, c'est l'inclinaison qui permet au quad-copter de réaliser une translation, ce qui signifie qu'un changement de la vitesse d'un rotor se traduit dans un mouvement en au moins trois DDL. Par exemple, l'augmentation de la vitesse du moteur avant aura comme conséquence un mouvement de tangage. Nous pouvons donc commander les six DDL de quad-copter avec les quatre mouvements principaux.

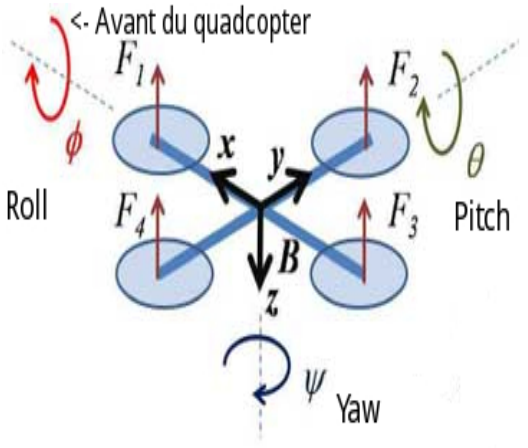

FIGURE II.23 – Mouvements de roulis, tangage et lacet

II.3.1 Mouvement de roulis (Roll)

Ce mouvement est réalisé par le changement des vitesses des propulseurs droite et gauche ce qui applique un couple autour de l'axe x. Ce mouvement est couplé avec un mouvement de translation selon l'axe y figure II.23.

II.3.2 Mouvement de tangage (Pitch)

Ce mouvement est réalisé par le changement des vitesses des propulseurs avant et arrière ce qui applique un couple autour de l'axe y. Ce mouvement est couplé avec un mouvement de translation selon l'axe x figure II.23.

II.3.3 Mouvement de lacet (Yaw)

Le mouvement de lacet est obtenu par le changement des vitesses des moteurs avant-arrière et les moteurs gauche-droit deux par deux. Il conduit à un couple autour de l'axe z qui fait tourner le quad-copter. Ainsi, lorsque le couple global est déséquilibré, le quad-copter tourne sur lui-même autour de z. Cette commande

conduit seulement à une accélération de l'angle de lacet figure II.23.

II.3.4 Mouvement vertical

Afin de planer, toute la force de portance devrait seulement être le long de l'axe z avec une grandeur exactement opposée à la force de pesanteur. Par conséquent, la poussée produite par chaque rotor doit être identique.

Les mouvements ascendant et descendant sont obtenus par la variation de la vitesse de rotation des moteurs par conséquence la poussée produite, si la force de portance est supérieure au poids du quad-copter le mouvement est ascendant, et si la force de portance est inférieure au poids du quad-copter le mouvement est descendant.[6]

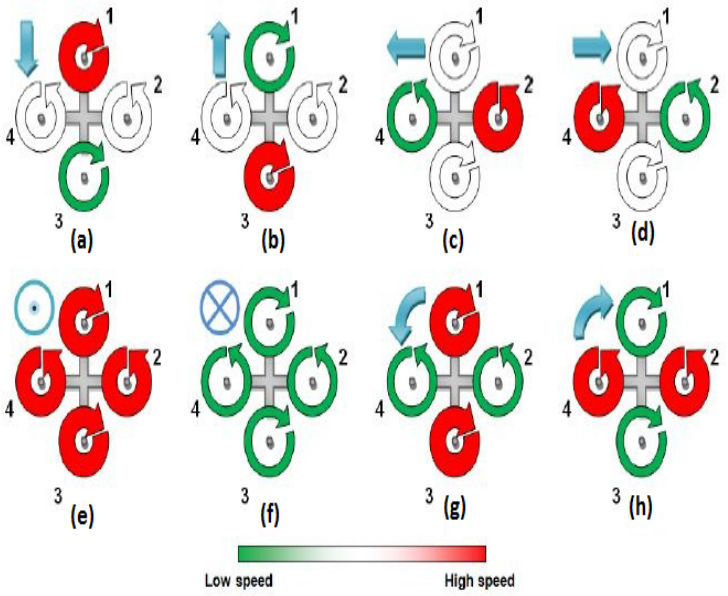

FIGURE II.24 – Mouvements d'un quad-copter

II.4 modèle dynamique

La modélisation des robots volants est une tâche délicate puisque la dynamique du système est fortement non linéaire et pleinement couplée. Afin de pouvoir comprendre au mieux le modèle dynamique, nous faisons quelques hypothèses :

- La structure du quadrirotor est supposée rigide et symétrique, ce qui induit que la matrice d'inertie sera supposée diagonale.

- Les hélices sont supposées rigides pour pouvoir négliger l'effet de leur déformation lors de la rotation.

- Le centre de masse et l'origine du repère lié à la structure coïncident.

- Les forces de portance et de traînée sont proportionnelles aux carrés de la vitesse de rotation des rotors, ce qui est une approximation très proche du compor-

tement aérodynamique [4].

II.4.1 Matrice de rotation

Pour décrire la position et l'orientation du quad-copter, nous avons besoin de deux repères. Le premier est nommé le repère inertiel (E-frame). Il s'agit d'un référentiel orthogonal fixe de type galiléen. La définition d'un deuxième repère est nécessaire pour décrire l'orientation du quad-copter. Celui-ci est attaché au châssis du quad-copter et se déplace donc avec celui-ci. Il est dénommé le repère mobile (B-frame).
Au début le repère mobile coïncide avec le repère fixe, après le repère mobile fait un mouvement de rotation autour de l'axe x d'un angle de roulis($\frac{-\pi}{2} < \phi < \frac{\pi}{2}$), suivi d'une rotation autour de l'axe y d'un angle de tangage ($\frac{-\pi}{2} < \theta < \frac{\pi}{2}$), suivi d'une rotation autour de l'axe z d'angle de lacet ($\frac{-\pi}{2} < \psi < \frac{\pi}{2}$) [11].

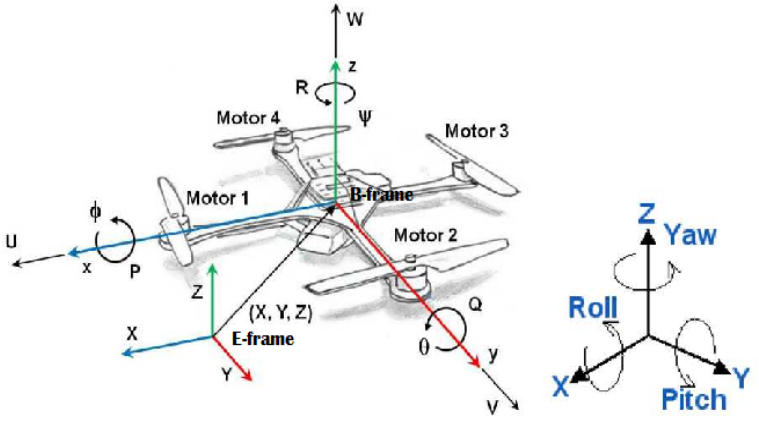

FIGURE II.25 – Repérage d'un quadrirotor

Donc on a la formule de la matrice de rotation **R** [8] :

$$\mathbf{R}=\mathrm{Rot}_z(\psi)\times Rot_y(\theta)\times Rot_x(\phi) = \begin{bmatrix} c\psi & -s\psi & 0 \\ s\psi & c\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} c\theta & 0 & s\theta \\ 0 & 1 & 0 \\ -s\theta & 0 & c\theta \end{bmatrix} \times \begin{bmatrix} 1 & 0 & 0 \\ 0 & c\phi & -s\phi \\ 0 & s\phi & c\phi \end{bmatrix} \quad \text{(II.1)}$$

$$\mathbf{R} = \begin{bmatrix} c\psi c\theta & s\phi s\theta c\psi - s\psi c\phi & c\phi s\theta c\psi + s\psi s\phi \\ s\psi c\theta & s\phi s\theta s\psi + c\psi c\phi & c\phi s\theta s\psi - s\phi c\psi \\ -s\theta & s\phi c\theta & c\phi c\theta \end{bmatrix} \quad \text{(II.2)}$$

Avec $c := cos$, $s := sin$.

II.4.2 Vitesses de rotation

Nous exprimons les vitesses de rotation $\Omega_1, \Omega_2, \Omega_3$ dans le repère fixe, en fonction des vitesses de rotation $\dot{\phi}, \dot{\theta}, \dot{\psi}$ dans le repère mobile :

$$\Omega = \begin{bmatrix} \Omega_1 \\ \Omega_2 \\ \Omega_3 \end{bmatrix} = \begin{bmatrix} \dot{\phi} \\ 0 \\ 0 \end{bmatrix} + Rot_x(\phi)^{-1} \begin{bmatrix} 0 \\ \dot{\theta} \\ 0 \end{bmatrix} + (Rot_y(\theta)Rot_x(\phi))^{-1} \begin{bmatrix} 0 \\ 0 \\ \dot{\psi} \end{bmatrix} \tag{II.3}$$

En effet, la rotation en roulis a lieu lorsque les repères sont encore confondus. Puis, en ce qui concerne le tangage, le vecteur représentant la rotation doit être exprimé dans le repère fixe : il est donc multiplié par $Rot(\phi)^{-1}$. De même, le vecteur représentant la rotation en lacet doit être exprimé dans le repère fixe qui a déjà subit deux rotations. On arrive ainsi à :

$$\Omega = \begin{bmatrix} \Omega_x \\ \Omega_y \\ \Omega_z \end{bmatrix} = \begin{bmatrix} \dot{\phi} \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ \dot{\theta}c\phi \\ -\dot{\theta}s\phi \end{bmatrix} + \begin{bmatrix} -\dot{\psi}s\theta \\ \dot{\psi}s\phi c\theta \\ \dot{\psi}c\phi c\theta \end{bmatrix} = \begin{bmatrix} \dot{\phi} - \dot{\psi}s\theta \\ \dot{\theta}c\phi + \dot{\psi}s\phi c\theta \\ \dot{\psi}c\phi c\theta - \dot{\theta}s\phi \end{bmatrix} \tag{II.4}$$

$$\Omega = \begin{bmatrix} 1 & 0 & -s\theta \\ 0 & c\phi & s\phi c\theta \\ 0 & -s\phi & c\phi c\theta \end{bmatrix} \times \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} \tag{II.5}$$

Donc :

$$\begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} 1 & s\phi tan\theta & c\phi tan\theta \\ 0 & c\phi & -s\phi \\ 0 & \frac{s\phi}{c\theta} & \frac{c\phi}{c\theta} \end{bmatrix} \times \begin{bmatrix} \Omega_x \\ \Omega_y \\ \Omega_z \end{bmatrix} \tag{II.6}$$

Lorsque le quadriritor fait des petites rotations, nous pouvons faire les approximations suivantes :

$$c\phi = c\theta = c\psi = 1, s\phi = s\theta = s\psi = 0$$

Nous obtenons donc :

$$\begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} \Omega_x & \Omega_y & \Omega_z \end{bmatrix}^T \tag{II.7}$$

II.4.3 Vitesses de translation

Les vitesses linéaires dans le repère fixe V_x^b, V_y^b, V_z^b sont exprimées en fonction des vitesses linéaires dans le repère mobile V_x^m, V_y^m, V_z^m par :

$$V = \begin{bmatrix} V_x^b \\ V_y^b \\ V_z^b \end{bmatrix} = R \times \begin{bmatrix} V_x^m \\ V_y^m \\ V_z^m \end{bmatrix} \tag{II.8}$$

II.4.4 Les effets physiques agissants sur le quad-copter

Une fois le drone est en mouvement, sa dynamique est soumise à certaines forces et moments [7] :

II.4.4-a Les forces

La force de pesanteur : qui est donnée par $P = m \times g$, où : m est la masse totale de quadrirotor et g la gravité.

Les forces de poussée : ces forces sont provoquées par la rotation des moteurs, elles sont perpendiculaires sur le plan des hélices, et proportionnelles au carrée de la vitesse de rotation des moteurs :

$$F_i = b\omega_i^2 \tag{II.9}$$

Avec : i le nombre des moteurs et b le coefficient de portance qui dépend de la forme et le nombre des pales et la densité de l'air.

Les forces de traînée : la traînée est la force qui s'oppose au mouvement d'un corps dans un fluide pesant et agit comme un frottement. Nous avons donc deux forces de traînée agissant sur le système :

-La trainée dans les hélices : elle agit sur les pales, elle est proportionnelle à la densité de l'air, à la forme des pales et au carré de la vitesse de rotation de l'hélice, elle est donnée par :

$$T_h = d\omega_i^2 \tag{II.10}$$

Avec : d est le coefficient de drag qui dépend de la fabrication de l'hélice.

-La traînée selon les axes (x, y, z) : elle est due au mouvement du quadrirotor :

$$F_t = K_{ft}V \tag{II.11}$$

Avec : K_{ft} le coefficient de traînée de translation et V la vitesse linéaire.

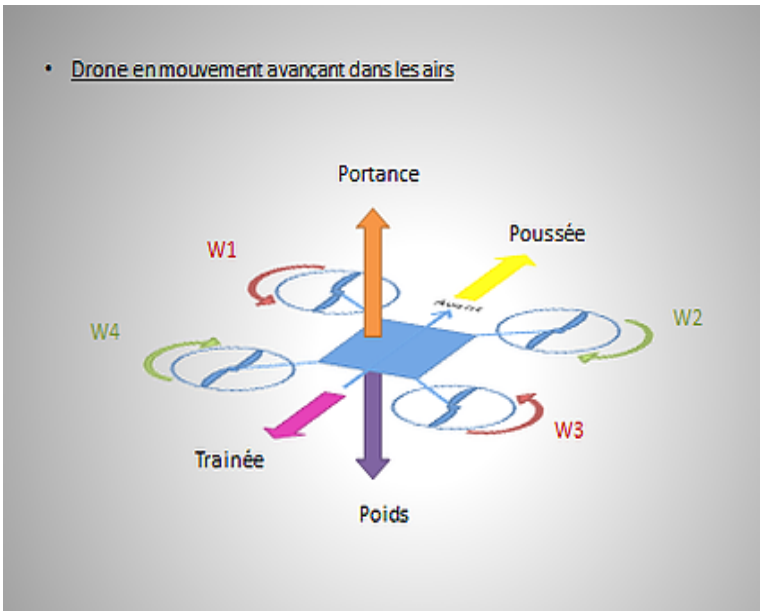

FIGURE II.26 – Forces exercées sur un drone

II.4.4-b Les moments

Il y a plusieurs moments agissants sur un quad-copter, qui sont dus aux forces de poussée et de traînée et aux effets gyroscopiques.

Moments dus aux forces de poussée :
-La rotation autour de l'axe x : due au moment créé par la différence entre les forces de portance des rotors droit et gauche, ce moment est donné par la relation suivante :

$$M_x = l(F_4 - F_2) = lb(\omega_4^2 - \omega_2^2) \tag{II.12}$$

Avec l est la longueur du bras entre le rotor et le centre de gravité du quadrirotor.

- La rotation autour de l'axe y : due au moment créé par la différence entre les forces de portance des rotors avant et arrière, ce moment est donné par la relation suivante :

$$M_y = l(F_3 - F_1) = lb(\omega_3^2 - \omega_1^2) \tag{II.13}$$

Moments dus aux forces de traînée :

- La rotation autour de l'axe z : due à un couple réactif provoqué par les couples de traînée dans chaque hélice, ce moment est donné par la relation suivante :

$$M_z = d(-\omega_1^2 + \omega_2^2 - \omega_3^2 + \omega_4^2) \tag{II.14}$$

- Moment résultant des frottements aérodynamiques, il est donné par :

$$M_a = K_{fa}\Omega^2 \tag{II.15}$$

Avec : K_{fa} Le coefficient des frottements aérodynamiques et Ω est la vitesse angulaire.

II.4.4-c Les effets gyroscopiques

Tout objet en rotation autour d'un axe est soumis à l'effet gyroscopique. C'est la capacité qu'a cet objet à conserver son axe de rotation ou, de façon plus précise, à conserver son moment angulaire.

Dans notre cas il y a deux moments gyroscopiques, le premier est le moment gyroscopique des hélices, l'autre est le moment gyroscopique dû aux mouvements du quadrirotor.

Moment gyroscopique des hélices : il est donné par la relation suivante :

$$M_{gh} = \sum_{i=1}^{4} \Omega \wedge Jr \begin{bmatrix} 0 \\ 0 \\ (-1)^{i+1}\omega_i \end{bmatrix} \tag{II.16}$$

Avec Jr est l'inertie des rotors.

Moment gyroscopique dû aux mouvements de quadrirotor : il est donné par la relation suivante :

$$M_{gm} = \Omega \wedge J\Omega \tag{II.17}$$

Avec J est l'inertie du système.

II.4.5 Développement du Modèle mathématique selon Newton-Euler

Nous avons négligé la force de traînée selon les axes et le moment dû aux frottements aérodynamiques pour développer le modèle dynamique.
En utilisant la formulation de Newton-Euler, les équations sont écrites sous la forme suivante [10] :

$$\begin{cases} \dot{\xi} = V \\ m\ddot{\xi} = F_f + Fg \\ \dot{R} = RS(\Omega) \\ J\dot{\Omega} = -\Omega \wedge J\Omega + M_f - M_{gh} \end{cases} \tag{II.18}$$

Avec :
ξ : est le vecteur de position du quadrirotor.
m : la masse totale du quadrirotor.
Ω : la vitesse angulaire exprimée dans le repère fixe.
R : la matrice de rotation.
$\wedge$: le produit vectoriel.
J : la matrice d'inertie symétrique de dimension (3x3), elle est donnée par :

$$J = \begin{bmatrix} J_{xx} & 0 & 0 \\ 0 & J_{yy} & 0 \\ 0 & 0 & J_{zz} \end{bmatrix} \tag{II.19}$$

S(Ω) : est la matrice antisymétrique ; pour un vecteur de vélocité $\Omega = \begin{bmatrix}\Omega_1 & \Omega_2 & \Omega_3\end{bmatrix}^T$ elle est donnée par :

$$S(\Omega) = \begin{bmatrix} 0 & -\Omega_3 & \Omega_2 \\ \Omega_3 & 0 & -\Omega_1 \\ -\Omega_2 & \Omega_1 & 0 \end{bmatrix} \tag{II.20}$$

F_f : est la force totale générée par les quatre rotors, elle est donnée par :

$$F_f = R \times \begin{bmatrix}0 & 0 & \sum_{i=1}^{4} F_i\end{bmatrix}^T \tag{II.21}$$

$$F_i = b\omega^2 \tag{II.22}$$

F_g : est la force de pesanteur :

$$F_g = \begin{bmatrix} 0 \\ 0 \\ -mg \end{bmatrix} \tag{II.23}$$

M_f : est le moment provoqué par les forces de poussée et de traînée :

$$M_f = \begin{bmatrix} l(F_4 - F_2) \\ l(F_3 - F_1) \\ d(-\omega_1^2 + \omega_2^2 - \omega_3^2 + \omega_4^2) \end{bmatrix} \tag{II.24}$$

II.4.5-a Dynamique de translation

On a :

$$m\ddot{\xi} = F_f + F_g \tag{II.25}$$

En remplaçant chaque force par son expression on obtient :

$$m \begin{bmatrix} \ddot{x} \\ \ddot{y} \\ \ddot{z} \end{bmatrix} = \begin{bmatrix} c\phi s\theta c\psi + s\phi s\psi \\ c\phi s\theta s\psi - s\phi s\psi \\ c\phi c\theta \end{bmatrix} \sum_{i=1}^{4} F_i - \begin{bmatrix} 0 \\ 0 \\ mg \end{bmatrix} \tag{II.26}$$

Les équations dynamiques de mouvement de translation sont comme suit :

$$\begin{cases} \ddot{x} = \dfrac{1}{m}(c\phi s\theta c\psi + s\phi s\psi)U_1 \\ \ddot{y} = \dfrac{1}{m}(c\phi s\theta s\psi - s\phi s\psi)U_1 \\ \ddot{z} = \dfrac{1}{m}(c\phi c\theta)U_1 - g \end{cases} \tag{II.27}$$

Avec :
$U_1 = b(\omega_1^2 + \omega_2^2 + \omega_3^2 + \omega_4^2)$

II.4.5-b Dynamique de rotation

On a :

$$J\dot{\Omega} = -M_{gm} - M_{gh} + M_f \tag{II.28}$$

En remplaçant chaque force par son expression on obtient :

$$\begin{bmatrix} J_{xx} & 0 & 0 \\ 0 & J_{yy} & 0 \\ 0 & 0 & J_{zz} \end{bmatrix} \begin{bmatrix} \ddot{\phi} \\ \ddot{\theta} \\ \ddot{\psi} \end{bmatrix} = - \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} \wedge (\begin{bmatrix} J_{xx} & 0 & 0 \\ 0 & J_{yy} & 0 \\ 0 & 0 & J_{zz} \end{bmatrix} \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix}) - \begin{bmatrix} J_r g(u)\dot{\theta} \\ J_r g(u)\dot{\phi} \\ 0 \end{bmatrix} + \begin{bmatrix} lb(\omega_4^2 - \omega_2^2) \\ lb(\omega_3^2 - \omega_1^2) \\ d(-\omega_1^2 + \omega_2^2 - \omega_3^2 + \omega_4^2) \end{bmatrix} \tag{II.29}$$

Les équations dynamiques de mouvement de rotation sont comme suit :

$$\begin{cases} \ddot{\phi} = \dfrac{J_{yy} - J_{zz}}{J_{xx}} \dot{\theta}\dot{\psi} - \dfrac{J_r g(u)}{J_{xx}} \dot{\theta} + \dfrac{U_2}{J_{xx}} \\ \ddot{\theta} = \dfrac{J_{zz} - J_{xx}}{J_{yy}} \dot{\phi}\dot{\psi} + \dfrac{J_r g(u)}{J_{yy}} \dot{\phi} + \dfrac{U_3}{J_{yy}} \\ \ddot{\psi} = \dfrac{J_{xx} - J_{yy}}{J_{zz}} \dot{\phi}\dot{\theta} + \dfrac{U_4}{J_{zz}} \end{cases} \tag{II.30}$$

Avec :
$g(u) = (\omega_1 - \omega_2 + \omega_3 - \omega_4)$
$U_2 = lb(\omega_4^2 - \omega_2^2)$
$U_3 = lb(\omega_3^2 - \omega_1^2)$
$U_4 = d(-\omega_1^2 + \omega_2^2 - \omega_3^2 + \omega_4^2)$
En conséquence, le modèle dynamique complet qui régit le quadrirotor est le suivant [7]

$$\begin{cases} \ddot{x} = \dfrac{1}{m}(c\phi s\theta c\psi + s\phi s\psi)U_1 \\ \ddot{y} = \dfrac{1}{m}(c\phi s\theta s\psi - s\phi s\psi)U_1 \\ \ddot{z} = \dfrac{1}{m}(c\phi c\theta)U_1 - g \\ \ddot{\phi} = \dfrac{J_{yy} - J_{zz}}{J_{xx}} \dot{\theta}\dot{\psi} - \dfrac{J_r g(u)}{J_{xx}} \dot{\theta} + \dfrac{U_2}{J_{xx}} \\ \ddot{\theta} = \dfrac{J_{zz} - J_{xx}}{J_{yy}} \dot{\phi}\dot{\psi} + \dfrac{J_r g(u)}{J_{yy}} \dot{\phi} + \dfrac{U_3}{J_{yy}} \\ \ddot{\psi} = \dfrac{J_{xx} - J_{yy}}{J_{zz}} \dot{\phi}\dot{\theta} + \dfrac{U_4}{J_{zz}} \end{cases} \tag{II.31}$$

Avec :

$$\begin{bmatrix} U_1 \\ U_2 \\ U_3 \\ U_3 \end{bmatrix} = \begin{bmatrix} b & b & b & b \\ 0 & -lb & 0 & lb \\ -lb & 0 & lb & 0 \\ -d & d & -d & d \end{bmatrix} \begin{bmatrix} \omega_1^2 \\ \omega_2^2 \\ \omega_3^2 \\ \omega_4^2 \end{bmatrix} \tag{II.32}$$

II.4.6 Représentation d'état du système

Notre espace d'état non linéaire $\dot{x} = f(x, u)$ avec le vecteur d'état à 12 dimensions peut être représenté comme suite :
$x = (x_1, x_2, \cdots x_{12})^T = (\phi, \dot{\phi}, \theta, \dot{\theta}, \psi, \dot{\psi}, x, \dot{x}, y, \dot{y}, z, \dot{z})^T$
Donc la représentation d'état est la suivante :

$$\begin{cases} \dot{x_1} = x_2 \\ \dot{x_2} = \dfrac{J_{yy} - J_{zz}}{J_{xx}} x_4 x_6 - \dfrac{J_r g(u)}{J_{xx}} x_4 + \dfrac{U_2}{J_{xx}} \\ \dot{x_3} = x_4 \\ \dot{x_4} = \dfrac{J_{zz} - J_{xx}}{J_{yy}} x_2 x_6 + \dfrac{J_r g(u)}{J_{yy}} x_2 + \dfrac{U_3}{J_{yy}} \\ \dot{x_5} = x_6 \\ \dot{x_6} = \dfrac{J_{xx} - J_{yy}}{J_{zz}} x_2 x_4 + \dfrac{U_4}{J_{zz}} \\ \dot{x_7} = x_8 \\ \dot{x_8} = \dfrac{U_1}{m}(cos(x_1)sin(x_3)cos(x_5) + sin(x_1)sin(x_5)) \\ \dot{x_9} = x_{10} \\ \dot{x_{10}} = \dfrac{U_1}{m}(cos(x_1)sin(x_3)sin(x_5) - sin(x_1)cos(x_5)) \\ \dot{x_{11}} = x_{12} \\ \dot{x_{12}} = \dfrac{U_1}{m}(cos(x_1)cos(x_3)) - g \end{cases} \tag{II.33}$$

II.5 Conclusion

Dans ce chapitre nous avons déterminé le modèle dynamique du quadrirotor ainsi que sa représentation d'état non linéaire, après avoir donné une description générale de ses mouvement et les force qui influent sur sa dynamique. Nous avons utilisé le formalisme de Newton-Euler qui nous a permis d'obtenir un modèle simplifié.
La partie commande et simulation du quad-coptère sera abordée dans le prochain chapitre.

Identification des paramètres et synthèse de commande

Sommaire

Résumé

Dans ce chapitre nous allons faire une identification des paramètres du quad-copter, puis synthétiser une loi de commande par PID et l'appliquer sur le modèle symulink du système.

III.1 Introduction

Le travail fait précédemment était d'établir un modèle dynamique décrivant les mouvements du quadrirotor, maintenant nous devons trouver une commande robuste qui permet de stabiliser notre véhicule aérien et générer une trajectoire pour un vol autonome.
Notre but dans ce chapitre c'est de réaliser un système de régulation de type PID, puis faire une simulation des trajectoires du quad-coptère sous cette loi de commande.

III.2 Composants du drone

Ce travail est continuation du travail effectué dans le cadre d'un projet de fin d'étude précédent [7]. En s'inspirant de ce travail, notre drone sera composé des éléments suivants :

- Un châssis.
- Quatre moteurs brushless 800 KV.
- Quatre ESC 25 A.
- Quatre hélices $11X5$.
- Une batterie Lipo $3S$.
- Un contrôleur de vol (Arduino).

Le choix de ces éléments est justifié dans le travail [7], et la description de ces éléments sera détaillé dans le chapitre suivant.
Dans notre travail, nous avons utilisé d'autre méthodes d'identification, de plus la partie électronique intelligente est différente de [7].

III.3 Identification des paramètres

Pour pouvoir obtenir des résultats de simulation proches du comportement réel nous devons d'abord identifier les paramètre de notre UAV :

III.3.1 Masse totale

Après avoir fini le montage et le câblage de notre quad-copter, nous avons utilisé une balance électronique pour le peser, nous avons trouvé que notre drone pèse 1218g ce qui est idéale en cas où l'on veut ajouter une masse supplémentaire donc il y aura pas de surcharge.

FIGURE III.27 – Le poids du quadrirotor

III.3.2 Coefficient de portance "b"

Le coefficient de portance b est proportionnel au carré de la vitesse de rotation du moteur. D'après le datasheet du moteur qui est illustré sur la figure ci-dessous : [7]

SPECS for Turnigy Multistar 2216-800Kv 14Pole Multi-Rotor Outrunner V2

Battery	Prop Size	RPM	Thrust (g)	Current (A)	Power
11.1V (3Cell)	12" Slow Flier	5,500	907	17	185
	11" thin Style	7,200	908	11.7	128
	11" Slow Flier	6,000	908	16.2	171
	10" Slow Flier	7,440	771	10.8	120
	9" Slow Flier	8,000	544	8.9	105

FIGURE III.28 – Spécifications du moteur Brushless utilisé

Nous remarquons qu'avec des hélices de 11 pouces de taille, la vitesse de rotation est de $7200tr/min = 754rad/sec$ ce qui génère une force de poussée égale à $908g = 8.90N$.
Et comme : $F_i = b\omega^2$

$b = 15.65 \times 10^{-4}[N.sec^2]$

III.3.3 Coefficient de drag "d"

Selon la théorie aérodynamique, le coefficient de traînée dépend du coefficient de portance avec la relation suivante :

$$\frac{d}{b} = \tan\alpha \tag{III.34}$$

où α est l'angle d'incidence [1] de l'hélice [12] [3].
Comme le montrent la figure III.29, l'angle d'incidence d'une hélice est formée par son corps et le courant d'air.

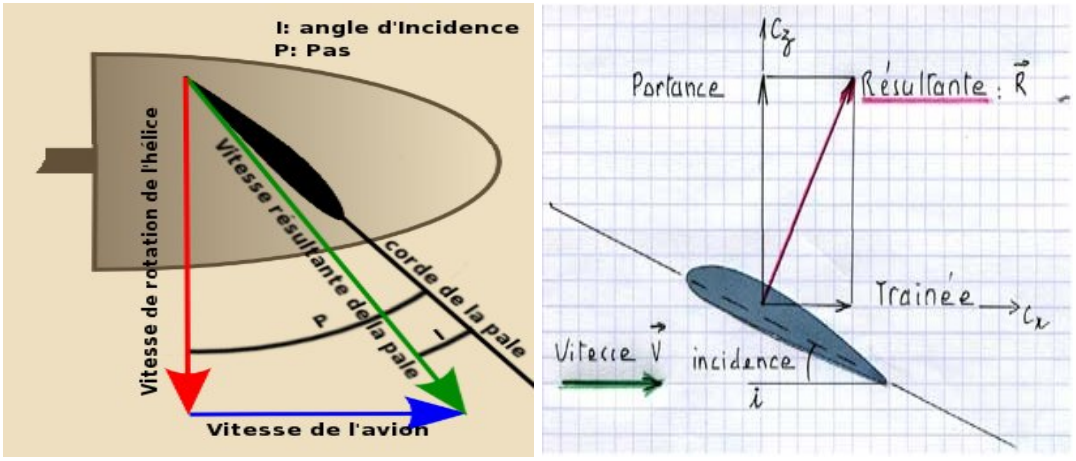

FIGURE III.29 – Angle d'attaque d'une hélice

Pour le calcul de cette angle nous pouvons utiliser la méthode du calcul de pente [2] comme illustré dans la figure suivante :

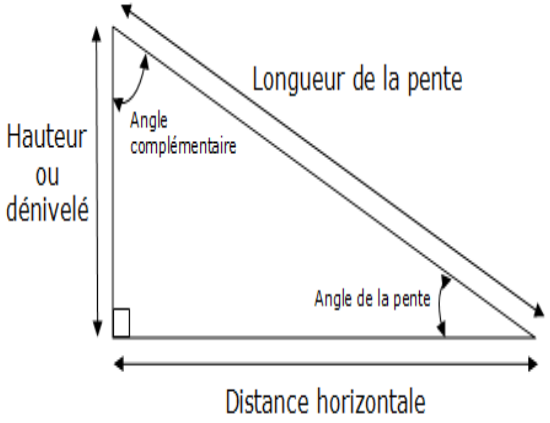

FIGURE III.30 – Calcul de pente

Nous mettons l'hélice à plat, cela nous permettra de mesurer la distance horizontale et la hauteur, puis les tracer sur papier, la longueur de pente est automatiquement définie. Maintenant nous pouvons mesurer l'angle d'inclinaison.

Cette angle vaut à peut prés dans notre cas : $\alpha = 21^{\circ}$.

1. En aérodynamique, l'angle d'incidence ou l'angle d'attaque, est l'angle que forme la corde d'un profil avec le courant de fluide.
2. a pente exprime l'inclinaison d'une surface par rapport à l'horizontale. Elle est le rapport entre le dénivelé (hauteur) et la distance horizontale

Donc : $d = 6 \times 10^{-4} N.m^2.s^2$

III.3.4 Matrice d'inertie "J"

Il y a deux méthodes pour déterminer les paramètres d'inertie soit par un calcul théorique de moment d'inertie ou par l'expérience du pendule bifilaire.
Dans notre cas nous avons utilisé la méthode expérimentale du pendule bifilaire [17].

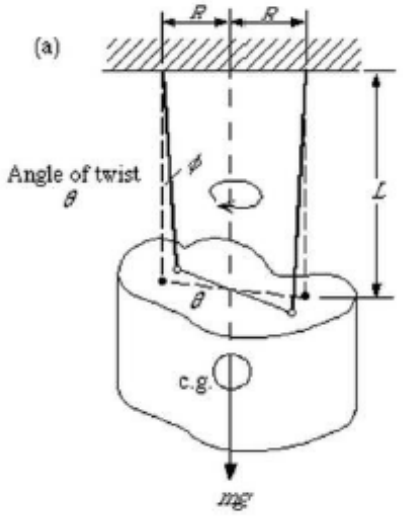

FIGURE III.31 – Expérience du pendule bifilaire

Dans cette expérience le corps de quad-coptère est maintenu par deux cordes . Le but de cette expérience est de mesurer la période d'oscillation, pour pouvoir calculer l'inertie du quadrirotor avec l'équation suivante :

$$I = \frac{mgT^2r}{4\pi^2L} \tag{III.35}$$

où :
m : la masse du système (Kg).
g : la force de pesanteur (m/s^2).
T : la période d'oscillation d'un cycle (s).
r : rayon entre les points des deux cordes (m).
L : longueur des deux cordes.

Les valeurs de ces paramètres sont indiquées dans le tableau suivant :

TABLE III.1 – Paramètres

Paramètre	Valeur
m (g)	1218
g (m/s^2)	9.81
r (m)	0.10
L (m)	0.57

Cette expérience est effectuée trois fois pour identifier les trois moments d'inertie J_{xx}, J_{yy}, J_{zz} respectivement.

Pour J_{xx}, l'expérience consiste à maintenir les deux cordes à la position d'équilibre dans le plan vertical figure III.32. Ensuite nous appliquant une rotation dans le plan horizontale avec un angle quelconque et nous relâchons.

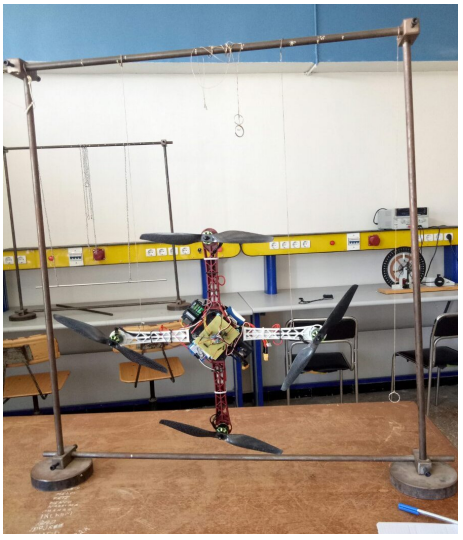

FIGURE III.32 – L'inertie selon X

Pour J_{yy}, l'expérience est répétée en effectuant au drone une rotation de $+/-90^{\circ}$ dans le plan vertical figure III.33. Le reste de l'expérience est identique.

FIGURE III.33 – L'inertie selon Y

Pour J_{zz}, le quadrirotor est maintenu en equilibre entre les deux fils à la position horizontale figure III.35. Ensuite pour la même procédure que précédent est appliquée.

FIGURE III.34 – L'inertie selon Z

Après calcul les moment d'inertie trouvés sont comme suit :

TABLE III.2 – Moments d'inertie

Paramètre	Valeur
Jxx (kg.m^2)	0.0149
Jyy (kg.m^2)	0.0149
Jzz (kg.m^2)	0.0294

III.3.5 Inertie du rotor (Jr)

Un rotor de moteur peut être modélisé par un empilement de cylindre, ensuite en utilisant la formule suivante [9] :

$I = \frac{1}{2}m \times r^2$.
Avec m est la masse du rotor et r le rayon.

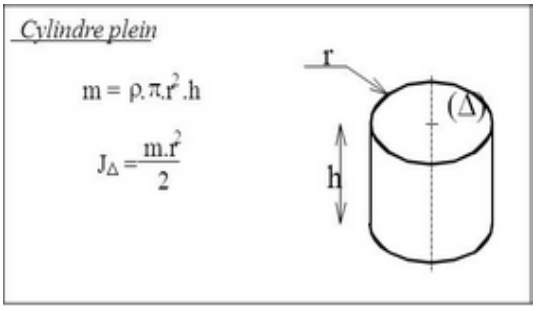

FIGURE III.35 – Inertie du rotor

Dans les spécifications du moteur nous avons :
m=0.130 Kg , r=0.14 m.

Donc : Jr = 1.27×10^{-5} Kg.m^2.

III.4 Synthèse du régulateur PID

Le régulateur PID, ou correcteur PID (proportionnel, intégral, dérivé) est un algorithme de contrôle qui permet d'améliorer les performances d'un asservissement, c'est-à-dire un système ou procédé en boucle fermée. C'est le régulateur le plus utilisé dans plusieurs domaines où ses qualités de correction s'appliquent à de multiples grandeurs physiques.
Parmi les avantages de ce régulateur nous citons [5] :
-Structure simple.
-Bonne performance dans plusieurs processus.
-Fiable, même sans un modèle spécifique du système de contrôle.

La correction se fait à base de l'erreur observée qui est la différence entre la consigne (valeur désirée) et la mesure (valeur réelle).

$$e = consigne - mesure$$

Le PID permet trois actions en fonction de cette erreur :
- Une action Proportionnelle : l'erreur est multipliée par un gain K_p pour améliorer la rapidité du système, plus K_p est grand plus le temps de réponse diminue.
- Une action Intégrale : l'erreur est intégrée sur un intervalle de temps s, puis divisée par

un gain K_i pour éliminer l'erreur résiduelle en régime permanent et améliore la précision mais, ceci provoque l'augmentation du déphasage.
- Une action Dérivée : l'erreur est dérivée suivant un temps t, puis multipliée par un gain K_d ce qui accélère la réponse du système et améliore la stabilité de la boucle.

La structure PID classique est montrée sur la figure suivante :

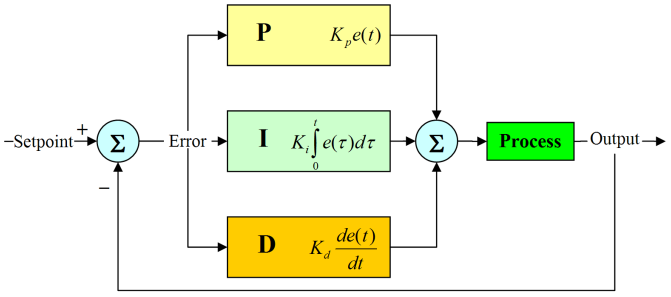

FIGURE III.36 – Structure parallèle d'un régulateur PID

L'expression générale du correcteur s'écrit sous la forme suivante :

$$U(t) = K_p e(t) + K_i \int_0^t e(\tau)d\tau + K_d \frac{de(t)}{dt} \tag{III.36}$$

Le quad-coptère est un système sous-actionné à 6 DDL, 4 entrées pour commander 6 sorties.
Le système est couplé entre deux boucles : une boucle interne qui contrôle les mouvement de rotation (roulis tangage et lacet), et qui doit être plus rapide [3] que la 2^{eme} boucle externe responsable à contrôler les mouvements de translation.

Donc ce système est composé de 6 PID, 3 régulateurs des mouvements de rotation dans la boucle interne et les 3 dans la boucle externe.

Pour notre cas nous voulons que les sorties du régulateur de positions soient des accélérations $(\ddot{x}, \ddot{y}, \ddot{z})$.
Si on pose $U = \ddot{x}$, et $e_x = x - x_d$ avec $x_d =$ constant, l'expression générale du PID devient comme suit :

$$\ddot{e_x}(t) = K_p e_x(t) + K_i \int_0^t e_x(t)(\tau)d\tau + K_d \frac{de_x(t)}{dt} \tag{III.37}$$

En dérivant cette equation nous obtenons l'équation suivante :

$$\dddot{e_x}(t) = K_p \dot{e_x}(t) + K_i e_x(t) + K_d \ddot{e_x}(t) \tag{III.38}$$

dont la solution en régime permanent est : $e = 0$ avec des gains bien choisis. Ainsi, $\ddot{x} = \ddot{x_d}$ ce qui assure le comportement souhaitable.

3. Il faut d'abord faire une rotation pour pouvoir faire une translation

Les accélérations angulaires sont régulées à partir de l'équation suivante :

$$U_i = K_p e_i(t) + K_i \int_0^t e_i(t)(\tau)d\tau + K_d \frac{de_i(t)}{dt} \tag{III.39}$$

Avec : $i = \phi, \theta, \psi$
Le modèle dynamique de notre quad-coptère contient deux effets gyroscopiques, l'influence de ces effets est dans notre cas moins importante que ceux du rotation des moteurs, ce qui va simplifier notre modèle. [13]
Donc les commandes U_2, U_3, U_4 sont exprimées comme suit :

$$\begin{cases} U_2 = \ddot{\phi} \times J_{xx} \\ U_3 = \ddot{\theta} \times J_{yy} \\ U_4 = \ddot{\psi} \times J_{zz} \end{cases} \tag{III.40}$$

L'expression de U_1 est déterminée depuis l'équation dynamique de translation $\ddot{z}$:

$$U_1 = \frac{m(\ddot{z} + g)}{cos\phi + \cos\theta} \tag{III.41}$$

Après avoir exprimé les commandes nous allons les utiliser pour déterminer les expressions des vitesses de rotation des moteurs :

A partir de l'équation (II.32) nous obtenons :

$$\begin{cases} \omega_1 = \sqrt{|\ \frac{U_1}{4b} - \frac{U_3}{2lb} - \frac{U_4}{4d}\ |} \\ \omega_2 = \sqrt{|\ \frac{U_1}{4b} - \frac{U_2}{2lb} + \frac{U_4}{4d}\ |} \\ \omega_3 = \sqrt{|\ \frac{U_1}{4b} + \frac{U_3}{2lb} - \frac{U_4}{4d}\ |} \\ \omega_4 = \sqrt{|\ \frac{U_1}{4b} + \frac{U_2}{2lb} + \frac{U_4}{4d}\ |} \end{cases} \tag{III.42}$$

Les commandes U_1, U_2, U_3, U_4 sont reconstruites à partir des expression des vitesses des moteurs par l'équation (II.32).

III.5 Contraintes de sous actionnement

Les contraintes de sous actionnement consistent à exprimer les angles ϕ_d et θ_d en fonction de ψ_d et le vecteur d'accélération $(\ddot{x}, \ddot{y}, \ddot{z})$ pour cela nous allons exploiter les équations dynamique de translation [7][18] :
Avec $U = (\ddot{x}, \ddot{y}, \ddot{z})$ l'équation II.25 devient :

$$U = -g_z + \frac{1}{m} R \times T \tag{III.43}$$

Avec : $T = \begin{bmatrix} 0 & 0 & \sum_{i=1}^{4} F_i \end{bmatrix}^T$ et $g_z = \begin{bmatrix} 0 & 0 & g \end{bmatrix}^T$

en déplaçant le vecteur de gravité g_z à gauche et en multipliant l'équation par R^T on obtient :

$$R^T(U + g_z) = \frac{1}{m}T \tag{III.44}$$

qui donne :

$$\begin{bmatrix} c\psi c\theta & s\phi s\theta c\psi - s\psi c\phi & c\phi s\theta c\psi + s\psi s\phi \\ s\psi c\theta & s\phi s\theta s\psi + c\psi c\phi & c\phi s\theta s\psi - s\phi c\psi \\ -s\theta & s\phi c\theta & c\phi c\theta \end{bmatrix}^T \begin{bmatrix} \ddot{x} \\ \ddot{y} \\ \ddot{z} + g \end{bmatrix} = \frac{1}{m} \begin{bmatrix} 0 \\ 0 \\ T \end{bmatrix} \tag{III.45}$$

$$\ddot{x}c\psi c\theta + \ddot{y}s\psi c\theta - (\ddot{z} + g)s\theta = 0 \tag{III.46}$$

$$\ddot{x}(s\phi s\theta c\psi - s\psi c\phi) + \ddot{y}(s\phi s\theta s\psi + c\psi c\phi) + (\ddot{z} + g)s\phi c\theta = 0 \tag{III.47}$$

$$\ddot{x}(c\phi s\theta c\psi + s\psi s\phi) + \ddot{y}(c\phi s\theta s\psi - s\phi c\psi) + (\ddot{z} + g)c\phi c\theta = \frac{1}{m}T \tag{III.48}$$

En considérant que $c\theta$ est différent de 0 on divise l'équation III.46 par $c\theta$ l'expression de θ_d est la suivante :

$$\theta_d = arctan(\frac{\ddot{x}c\psi + \ddot{y}s\psi}{\ddot{z} + g}) \tag{III.49}$$

On soustraite $III.47.cos\phi$ de $III.48.sin\phi$ pour obtenir :

$$\frac{1}{m}Tsin\phi = \ddot{x}sin\psi - \ddot{y}cos\psi \tag{III.50}$$

En manipulant l'équation III.44 nous obtenons la relation suivante :

$$(U + g_z)^T(U + g_z) = (\frac{1}{m}T)^T(\frac{1}{m}T) \tag{III.51}$$

Qui est équivalente à :

$$\ddot{x}^2 + \ddot{y}^2 + z\ddot{+}g^2 = (\frac{1}{m}T)^2 \tag{III.52}$$

L'expression de ϕ_d est obtenue en combinant les equations III.50 et III.52 :

$$\phi_d = arcsin(\frac{\ddot{x}sin\psi - \ddot{y}cos\psi}{\sqrt{\ddot{x}^2 + \ddot{y}^2 + (\ddot{z} + g)^2}}) \tag{III.53}$$

III.6 Simulation du modèle

Les paramètres que nous avons identifié sont cités dans le tableau suivant :

TABLE III.3 – Tableau des paramètres obtenus par l'identification

Paramètre	Valeur	Unité
m	1218	g
g	9.81	$m.s^{-2}$
l	0.25	m
J_{xx}	1.49×10^{-2}	$Kg.m^2$
J_{yy}	1.49×10^{-2}	$Kg.m^2$
J_{zz}	2.94×10^{-2}	$Kg.m^2$
J_r	1.27×10^{-5}	$Kg.m^2$
b	15.65×10^{-5}	$N.sec^2$
d	6×10^{-4}	$N.m.sec^2$

Le modèle Symulink de la commande du quadrirotor est le suivant :

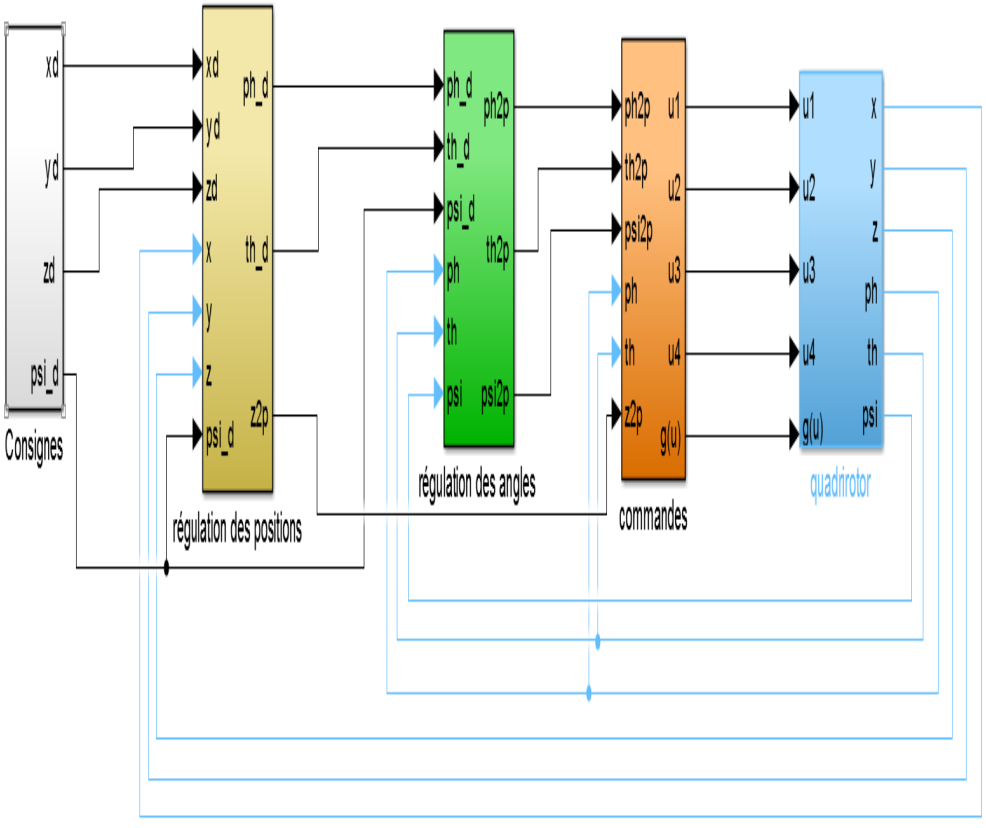

FIGURE III.37 – Modèle Simulink du quadrirotor

Le premier bloc est le bloc des entrées il contient les valeurs désirées de X, Y, Z et ψ.

Ensuite, les consigne vont être comparées avec les valeurs réelles, et les régulées par PID. Les sorties des PID $\ddot{x}, \ddot{y}, \ddot{z}$ et l'entrée ψ_d vont définir les expressions de ϕ_d et θ_d grâce au contraintes de sous-actionnement. Ce travail se fait dans le deuxième bloc.

Le troisième bloc est responsable de la régulation des angles ϕ, θ, ψ par d'autres PID.

Le quatrième bloc contient un couplage entre les commandes U_1, U_2, U_3, U_4 et les vitesses de rotation $\omega_1, \omega_2, \omega_3, \omega_4$, à partir des equations III.40, III.41, III.42 et II.32.

Les commandes U_1, U_2, U_3, U_4 sortantes vont agir sur notre système dans le dernier bloc.

Nous avons réussi à stabiliser les trajectoires du modèle du quad-coptère et cela grâce au paramètres PID ajustés par la méthode essai-erreur. Ces valeurs sont résumées dans le tableau suivant :

TABLE III.4 – Tableau des gains PID

Paramètre	ϕ	θ	ψ	X	Y	Z
K_p	2.1	1.3	4	18	16.2	5
K_i	0.01	0.01	-0.01	0	-0.01	0
K_d	12	18	1.5	9	9.05	4

III.7 Résultats et interprétations

III.7.1 Mouvements de translation

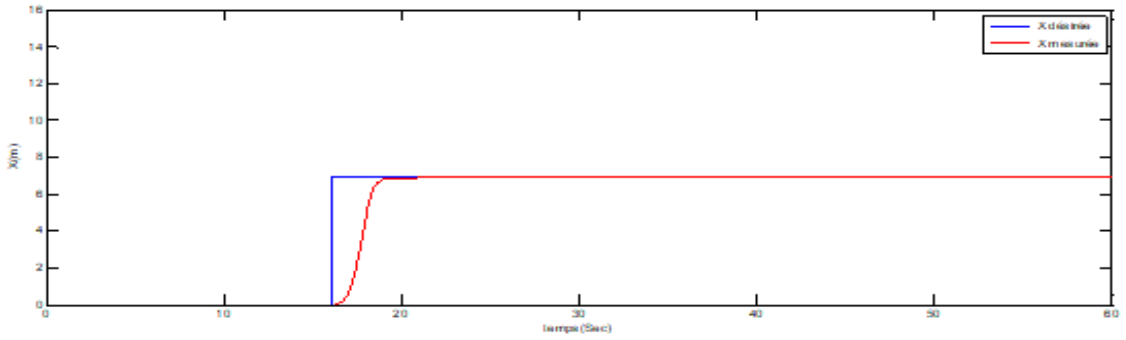

FIGURE III.38 – Tracés de x et x_d

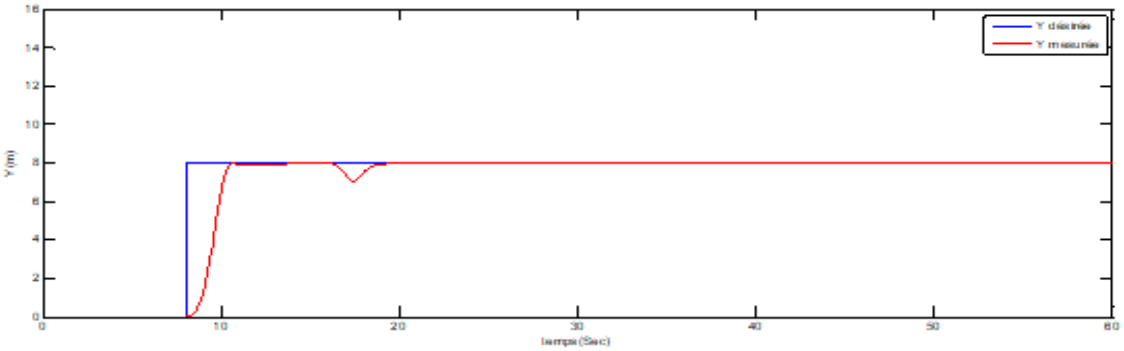

FIGURE III.39 – Tracés de y et y_d

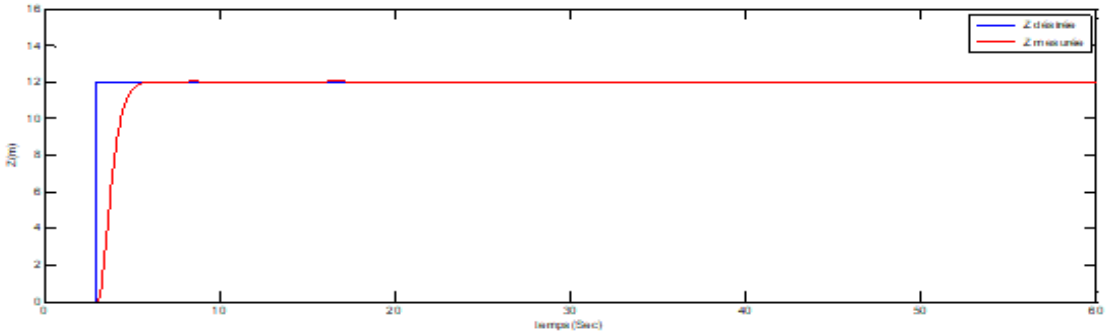

FIGURE III.40 – Tracés de z et z_d

Nous donnons en entrées des consignes de l'ordre de 7m pour X, 8m pour Y et 12m pour Z aux instants 16, 8 et 3 respectivement. Nous remarquons que les trajectoires réelles convergent vers les références en des temps acceptables. De plus l'erreur statique est nulle pour les trois tracés. La perturbation dans le tracé de Y est due au changement de la consigne de X à l'instant 16sec.

III.7.2 Mouvements de rotation

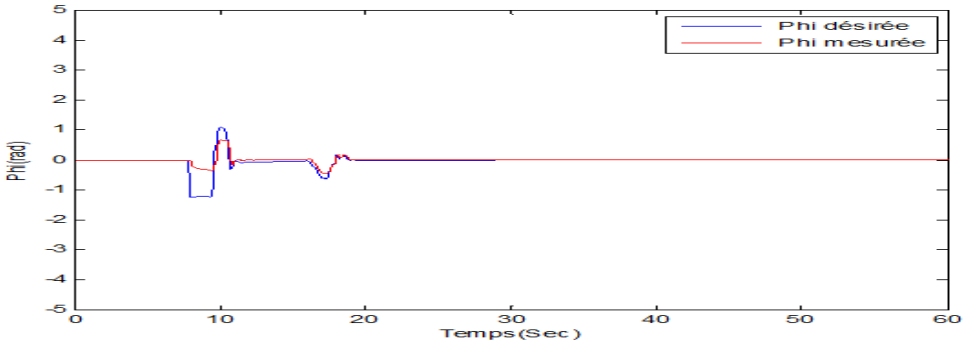

FIGURE III.41 – Tracés de ϕ et ϕ_d

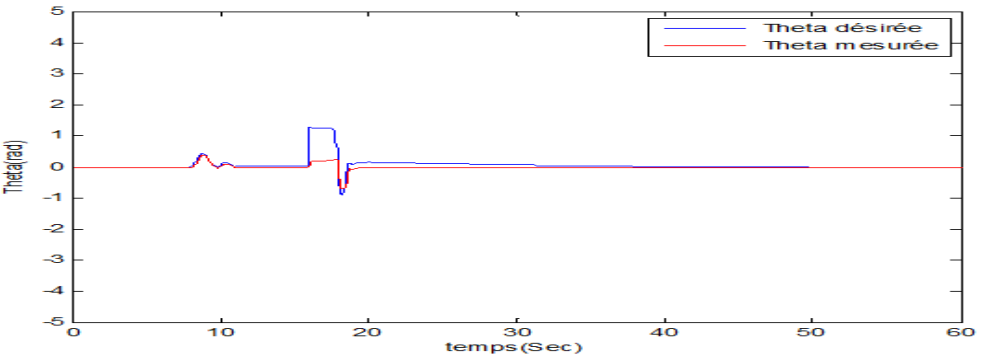

FIGURE III.42 – Tracés de θ et θ_d

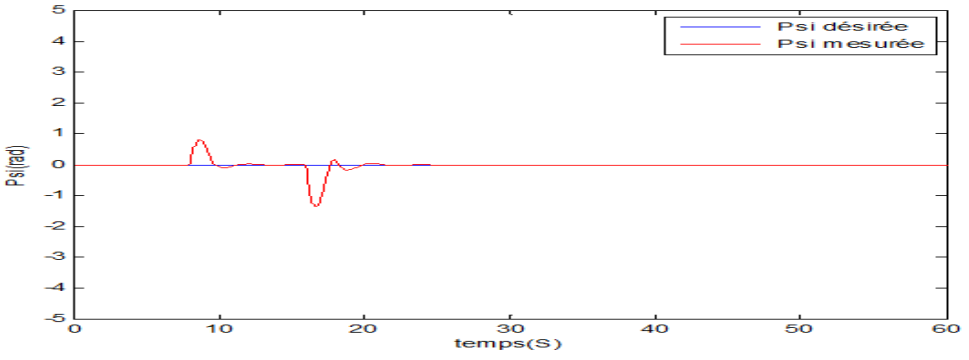

FIGURE III.43 – Tracés de ψ et ψ_d

Les angles ϕ_d, θ_d sont déduits à partir des contraintes de sous-actionnement, l'angle ψ_d est fixé à zéro.
Les valeurs mesurées suivent les valeurs désirés malgré les changements des consignes qui peuvent être considérés comme des perturbations. Les trajectoires se stabilisent autour de l'origine, avec une erreur statique nulle.

Les résultats obtenus sont satisfaisants et prouvent le bon choix des paramètres des PID et l'efficacité de ce régulateur en assurant la stabilité et en annulant l'erreur statique.

III.8 Conclusion

Dans la première partie de ce chapitre nous nous sommes basés sur l'identification des paramètres du quadrirotor, et cela afin d'obtenir un comportement en simulation le plus proche possible du comportement réel.
Ensuite, nous avons synthétisé des régulateurs PID pour stabiliser notre système et améliorer sa précision et rapidité. Les commandes synthétisées ont été appliquées sur le modèle de simulation et ont donné de bon résultats.
Maintenant il reste la phase pratique où nous allons construire notre quad-coptère, et lui appliquer cette commande PID. C'est ce que nous allons faire dans le quatrième chapitre.

Réalisation pratique

Sommaire

Résumé

Ce chapitre présente la conception et la réalisation pratique de notre véhicule aérien, nous allons tout d'abord donner une description du matériel utilisé, après nous allons expliquer le schéma générale et le montage du quad-coptère.

IV.1 Introduction

Nous présentons dans ce chapitre les éléments constituant notre drone. Nous avons choisi de construire un drone piloté par Arduino pour sa simplicité et son coût pas cher.
Comme son nom l'indique, un drone quadrirotor est réalisé à l'aide de 4 moteurs type BLDC, ces actionneurs sont commandés depuis un contrôleur de vol a base d'Arduino à travers 4 ESC, l'ensemble est alimenté par une batterie et piloté à l'aide d'une radio-commande.

Pour avoir une meilleure autonomie et temps de vol il faut faire un bon choix de ces composants.

IV.2 Matériel utilisé

IV.2.1 Châssis

Le châssis va déterminer beaucoup de composants, en effet si on prend un châssis de grande taille il faudra tout adapter à celui-ci. Il est par conséquent le premier élément à choisir.
Le quad-coptère est doté de quatre bras, en forme de « + » , « X » ,ou en « H » la seule différence entre ces deux modèles est le fait que celui en croix a une vue plus dégagé à l'avant et l'arrière. Ce qui le rend meilleur pour les prises de vue embarquées.
Les caractéristiques à prendre en compte pour le châssis sont le poids, qui sera lié aux matériaux utilisés et sa résistance au choc, plus le châssis est léger plus on conserve de la puissance et on gagne en temps de vol.

Le châssis que nous avons utilisé est un F450 de forme « X » et de 450mm d'envergure.

FIGURE IV.44 – Châssis F450

IV.2.2 Moteurs

Les drones utilisent actuellement des moteurs à courant continu sans balais (BLDC) : les bobines sont fixées au « boitier » du moteur et l'aimant tourne, contrairement aux moteurs CC avec balais. Les avantages des brushless sont multiples :

-Une régulation électronique de la vitesse plus performante.
-Couple et sens de rotation simple à gérer.
-Pas de frottement des balais qui peuvent engendrer des pertes de rendement.

La caractéristique la plus importante pour ce type de moteurs est le KV, c'est tout simplement sa constante de vélocité, elle correspond au nombre de tours par minute pour un volt. Le KV est la division du nombre de tours par minute par volt $KV = RPM/U$. Par exemple un moteur ayant un KV de 1000 tr/V fonctionnera à 12000 tr/min s'il est alimenté en 12 V.
Plus le KV sera élevé plus le moteur sera agressif, donc dur a contrôler. Pour un vol stationnaire il ne faut pas dépasser les 1000kV, et si on veut faire un vol acrobatique on pourra envisager un KV plus élevé, dans les 1500 KV.

Les moteurs que nous utilisons sont des Turnigy 800 KV, si notre batterie fournit un voltage de 12.6 V la vitesse de rotation peut aller jusqu'à 10000 tr/min. A noter que deux moteurs doivent être capable de soulever le poids de notre drone.

FIGURE IV.45 – Moteur Turnigy 800 KV

IV.2.3 Hélices

Les hélices sont de différents types (grand pas, petit pas), différentes tailles, et différentes matières (plastique, carbon, fibre de verre).
Plus l'hélice est grande plus il faudra de puissance pour la faire tourner. Mais plus elle est grande plus elle va générer de la portance et donc un vol plus stationnaire. Au contraire, pour des petites hélices il faudra moins de puissance et on aura donc moins de portance, mais un vol plus agressif.
Concernant le pas plus il est faible plus on aura de traction à faible vitesse, dans ce cas la vitesse maximale sera réduite, au contraire un grand pas implique moins de traction à faible vitesse, mais plus de poussée à vitesse élevée.
Les hélices ont un sens de rotation, soit dans le sens horaire (CW) soit dans le sens anti-horaire (CCW). Pour un quad-coptère on aura 2 hélices CW, et deux autres CCW.

les hélice que nous avons utilisé sont en fibre de carbone de taille 11 pouces et un pas de 5 (11×5) :

FIGURE IV.46 – Hélices $11x5$

IV.2.4 Contrôleurs de vitesse électroniques (ESC)

Les moteurs brushless sont commandés par des interfaces de puissance appelées « Contrôleurs » (ESC ou Electronique Speed Controller en anglais). Ce sont des circuits électroniques qui permettent de faire varier la vitesse de rotation des moteurs, à partir du courant délivré par la batterie. Ils sont caractérisés par une valeur en ampères (ex : $30A$) qui indique l'intensité maximale qu'ils peuvent encaisser pendant le vol et qui doit être supérieure où égale au courant supporté par le moteur .

FIGURE IV.47 – Esc $25A$

IV.2.5 Batterie

Les batteries utilisées sur un drone multirotors sont essentiellement des « Lithium Polymère ». Elles sont issues d'une technologie qui permet d'avoir un très bon rapport poids/puissance. Un élément LiPo (1S) fournit une tension de 3,7V. Sur un drone, on utilise en général des batteries à 3 ou 4 éléments (3S ou 4S).
Une batterie lipo a une capacité exprimée en mAh, par exemple une LIPO de 3300 mAh veut dire qu'elle peut fournir un courant de 3300 mA pendant une heure.

Le taux de décharge d'une batterie est exprimé en C : $1C$ représente un courant égal à 1 fois la capacité de la batterie.
Notre batterie est une Turnigy $3S$ ($3 \times 3.7 = 11.1V$), de $5Ah$ et de $20C$ qui peut délivrer un courant de $100A$ ($20 \times 5 = 100A$).

FIGURE IV.48 – LIPO $3S$ $5Ah$ $20C$

IV.2.6 Distributeur d'alimentation

La batterie doit fournir un tension au 4 ESC, pour cela un distributeur d'alimentation est recommandé. Cette carte divise les bornes positives et négatives de la batterie principale en quatre.

FIGURE IV.49 – Distributeur de tension

IV.2.7 La radio-commande

Pour piloter le drone, il faut un émetteur radio pour le pilote et un récepteur sur le drone. Il existe plusieurs technologie pour les radio-commandes, les radios FM en $41MHz$ (de moins en moins utilisées) et les radios en $2,4GHz$, souvent programmables pour s'adapter à chaque appareil radio-commandé.
Une radio-commande doit au minimum comporter 4 voies pour piloter un drone, ces voies sont généralement associés :
- au roulis (mouvement latéral à gauche et à droite).
- au tangage (qui se traduit par un mouvement d ?avant en arrière).
- à l'élévation (se rapprocher ou s'éloigner du sol).
- au lacet (rotation dans le sens normal ou inverse des aiguilles d'une montre).

Il existe deux modes de configuration des manettes, le Mode 1 dans lequel les gaz sont à droite et le mode 2 où les gaz sont à gauche. Certaines radios fonctionnent dans les deux sens, c'est-à-dire qu'elles peuvent envoyer des ordres à l'émetteur mais aussi recevoir des informations de celui-ci (tension de la batterie,...).

La radio-commande que nous avons utilisé pour notre quad-coptère est une turnigy tgy-i10 qui contient 10 canaux avec un émetteur et récepteur d'une fréquence de 2.4 GHz, un capteur de vitesse optique et magnétique, un capteur de température et un capteur de tension.

FIGURE IV.50 – Radio-commande Turnigy tgy-i10

IV.2.8 Contrôleur de vol

C'est une carte électronique, équipée de capteurs très précis, qui va traiter les consignes du pilote envoyées à l'émetteur ainsi que les informations envoyées par ses capteurs et va transmettre des impulsions électriques aux contrôleurs des moteurs pour faire varier leur vitesse.
Ces cartes sont équipées de gyroscopes et d'accéléromètres pour mesurer et compenser les déplacements.

Le controleur de vol utilisé dans notre projet est basé sur une carte Arduino, précisément nous utilisons la carte Arduino Uno qui est une carte à microcontrôleur open-source basée sur le microcontrôleur Microchip ATmega328P et développée par *Arduino.cc* . La carte est équipée de multiples broches d'entrée / sortie numériques et analogiques qui peuvent être interfacées avec diverses cartes d'extension et d'autres circuits.

L'Arduino Uno peut être alimentée via la connexion USB ou avec une alimentation externe. La source d'alimentation est sélectionnée automatiquement. L'alimentation externe peut provenir d'un adaptateur CA-CC ou d'une batterie. Les fils d ?une batterie peuvent être insérés dans les connecteurs à broches Gnd et Vin du connecteur POWER.
La carte peut fonctionner avec une alimentation externe de 6 à 20 volts. Avec une plage recommandée de 7 à 12 volts.
Les broches d'alimentation sont les suivantes :

VIN. La tension d'entrée sur la carte Arduino lorsqu'elle utilise une source d'alimentation externe .
5V. Cette broche génère un 5V régulé du régulateur sur la carte.
3.3V. Une alimentation de $3,3$ volts générée par le régulateur embarqué. Le courant maximal consommé est de 50 mA.
GND. Broches de terre.

Voici quelque spécifications de la carte :

Microcontrôleur : Microchip ATmega328P.
Tension de fonctionnement : 5 volts.
Tension d'entrée : 7 à 20 volts.
Broches d'E / S numériques : 14 (dont 6 fournissent une sortie PWM).
Broches d'entrée analogiques : 6.
Courant CC par broche d'E / S : 20 mA.
Courant CC pour broche 3,3 V : 50 mA.
Mémoire Flash : 32 Ko, dont 0,5 Ko utilisés par le chargeur de démarrage.
SRAM : 2 KB.
EEPROM : 1 KB.
Vitesse d'horloge : 16 MHz.
Longueur : 68,6 mm.
Largeur : 53,4 mm.
Poids : 25 g.

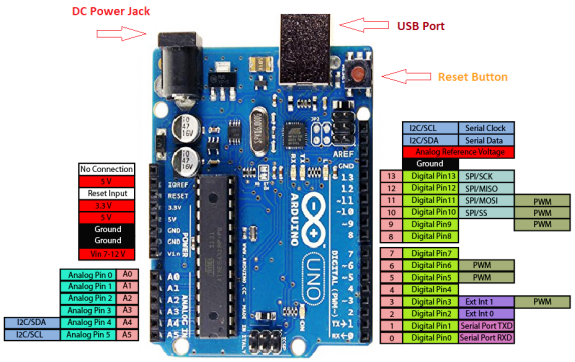

FIGURE IV.51 – Carte Arduino-Uno

IV.2.9 Capteurs

IV.2.9-a Accéléromètre

Un accéléromètre est un capteur qui permet de mesurer l'accélération linéaire d'un objet mobile. On parle d'accéléromètre même lorsqu'il s'agit de calculer les accélérations linéaires selon 3 axes orthogonaux.

IV.2.9-b Gyroscope

Le capteur gyroscopique mesure la vitesse angulaire autour de 3 axes. Les valeurs retournées sont analogiques. Ce capteur permet au quad-coptère de maintenir son équilibre, de calculer sa vitesse de rotation et de trouver son orientation angulaire.

Pour notre cas nous avons utilisé une IMU (Inertial measurement unit) $MPU6050$, cette centrale inertielle contient à la fois un accéléromètre à 3 axes et d'un gyroscope à 3 axes. Ce capteur est capable de mesurer la vitesse, l'accélération, et l'orientation angulaires.

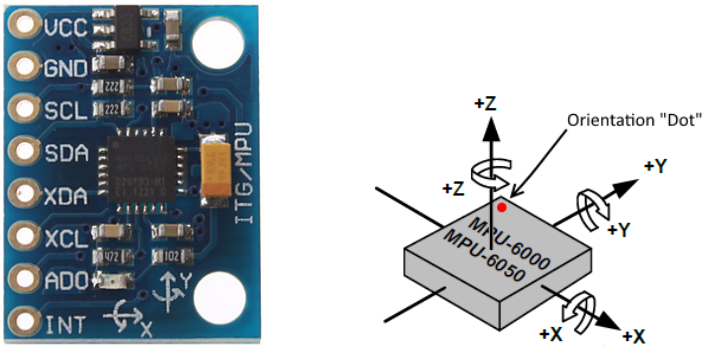

FIGURE IV.52 – Capteur MPU6050

Caractéristiques :
-Comprend un processeur de mouvement numérique (DMP), qui a la propriété de résoudre des calculs complexes.
-Constitué d'un convertisseur 16 bits analogique-numérique. Grâce à cette fonctionnalité, il capture les mouvements en trois dimensions en même temps.
-Utilise le module $I2C$ pour l'interfaçage avec Arduino.
-L'$MPU6050$ est moins cher, sa principale caractéristique est qu'il peut facilement être combiné avec un accéléromètre et un gyroscope.
-Ce module a quelques fonctionnalités célèbres qui sont facilement accessibles. En raison de sa disponibilité, il peut être utilisé avec un microcontrôleur célèbre comme Arduino.

IV.3 montage du quadrirotor

Les moteurs sont fixés au extrémités de châssis, la carte Arduino au centre, et le distributeur en dessous comme les figures suivantes le montrent.

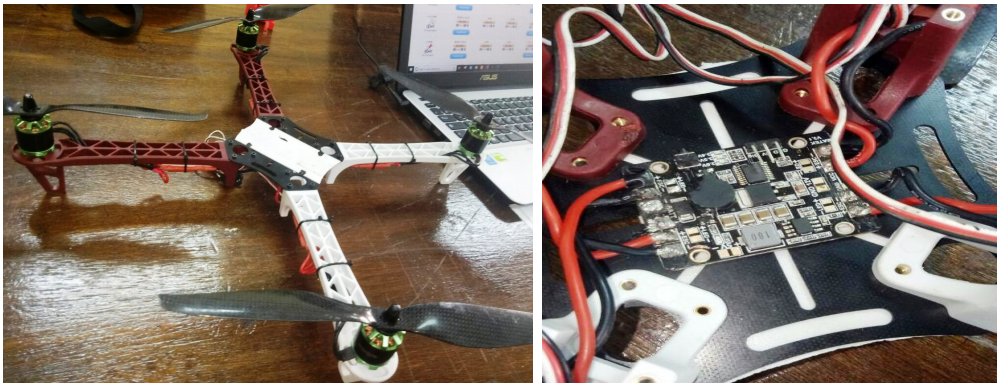

FIGURE IV.53 – Montage du quadrirotor

Le circuit général de notre projet est représenté dans la figure ci-dessous [2] :

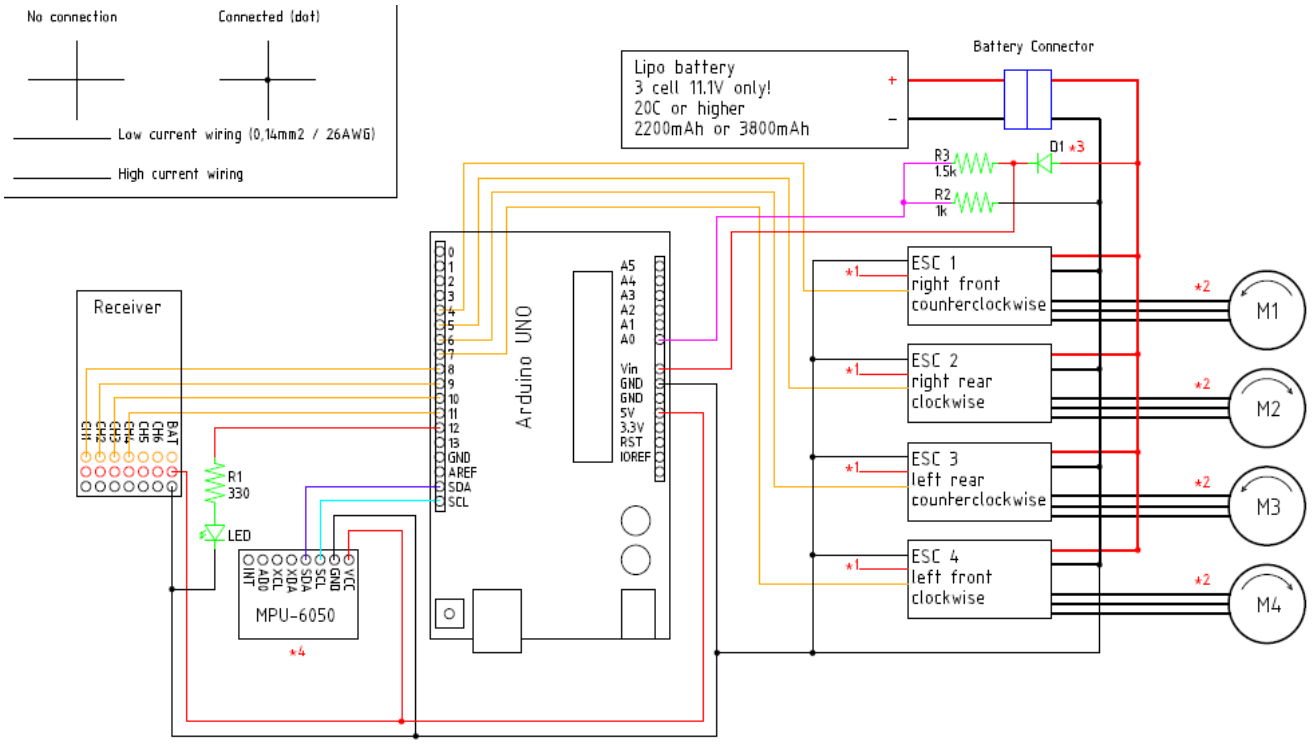

FIGURE IV.54 – Circuit général du quadrirotor

IV.3.1 Circuit de puissance

La connexion entre la batterie et le distributeur de puissance est faite à travers un connecteur $XT60$ Mâle/Femelle.

Les quatre ESC possèdent 2 fils d'alimentation $(+/-)$ qui sont soudées directement sur leurs emplacements au niveau de distributeur, de 3 phases qui sont connectées avec les fils des moteur.

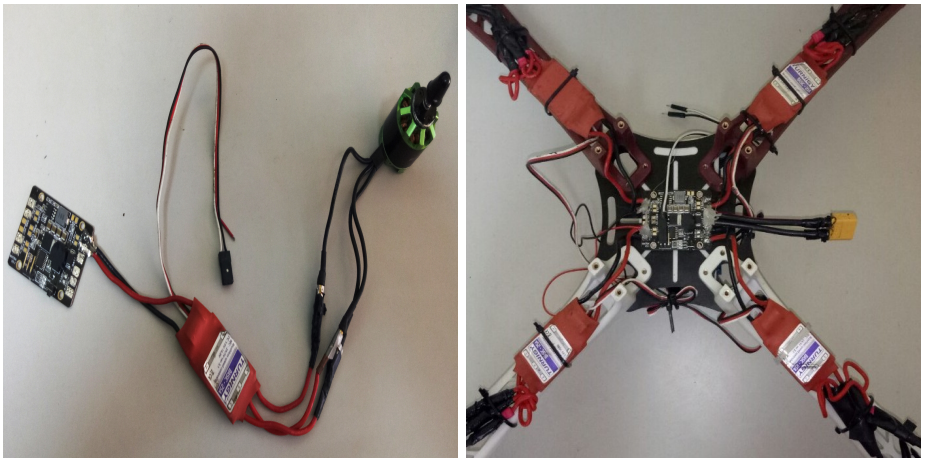

FIGURE IV.55 – Connexion des ESC avec le distributeur

Pour déterminer le sens de rotation des moteurs la structure utilisée est importante. Notre drone a une structure en (X), donc le sens de rotation est illustré dans la figure IV.56.

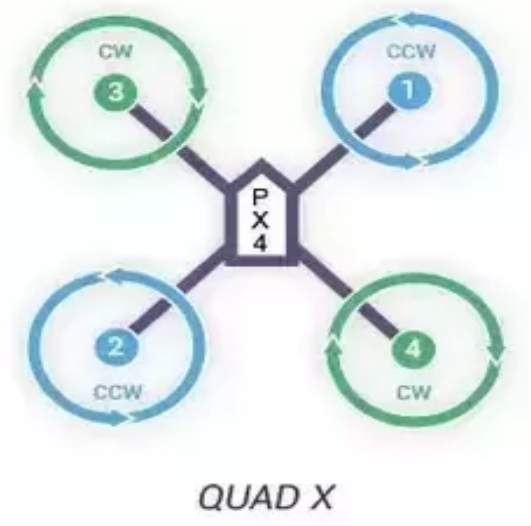

FIGURE IV.56 – Structure (X)

Les moteurs 1 et 2 tournent dans le sens anti-horaire, et les moteurs 3 et 4 dans le sens horaire. Pour changer le sens du rotation il faut juste inverser le branchement des phases.

L'alimentation de la carte Arduino est fourni directement avec le voltage de la batterie à travers le distributeur, cette alimentation est connectée à la broche Vin.

IV.3.2 Circuit de commande

Pour réaliser ce circuit nous avons utilisé un circuit imprimé compatible avec les broches de la carte Arduino pour faciliter les branchement.

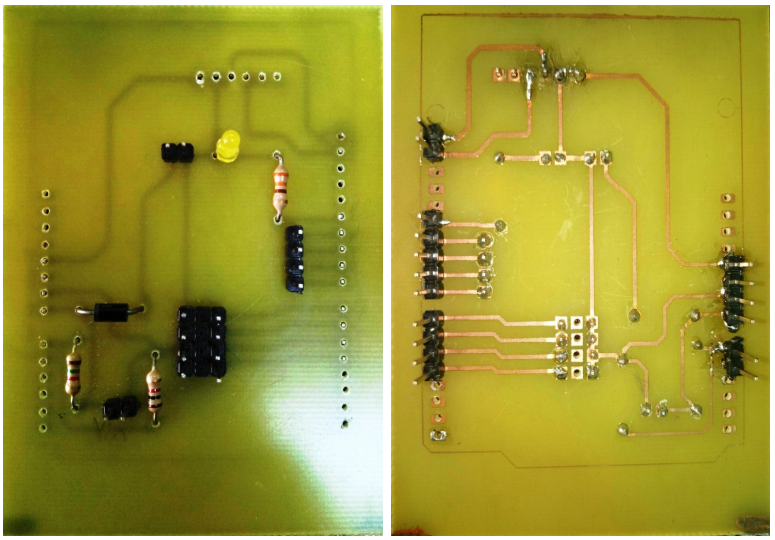

FIGURE IV.57 – Circuit imprimé pour le contrôleur de vol

- Le capteur $MPU6050$ possède 8 PINOUT, nous n'avons besoin que de 4 :
Vcc/GND : C'est l'alimentation du capteur. Branchées au 5V et GND de l'Arduino.
SCL : Cette broche est utilisée pour les impulsions d'horloge pour le fonctionnement en I2C.
SDA : Cette broche est utilisée pour transférer des données via une communication I2C.

Ce capteur doit de préférence être placé au centre du quadrirotor, avec l'axe z vertical (perpendiculaire à la surface), et les extrémités de l'IMU doivent être alignées avec les bords du quadcoptère. Pour cela nous l'avons fixé sur le circuit imprimé, figure IV.58.

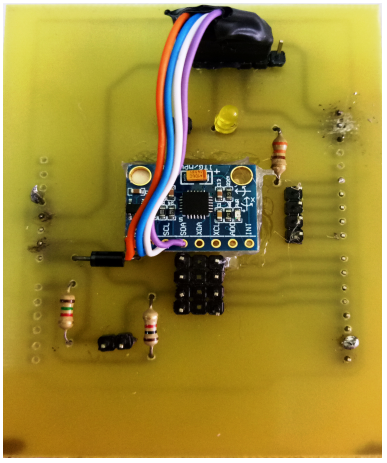

FIGURE IV.58 – Placement de l'IMU

- Le contrôleur de vol (Arduino + Circuit imprimé) est positionné au centre du châssis.

- Les fils des signaux des ESC (blancs) sont branchées au pins digitaux $4, 5, 6$ et 7, et les noires au GND respectivement comme montré dans la figure IV.54.
Le fil rouge : BEC (Battery Eliminator Circuit), il s'agit d'un régulateur de tension $5V$ que nous n'allons pas utiliser donc nous l'avons coupé.

- Pour pouvoir mesurer la tension de la batterie pendant le vol nous avons utiliser un diviseur de tension qui va fournir $5V$ à la broche $A0$ de l'Arduino quand la batterie est à $12.6V$.

- Le récepteur de radio-commande que nous utilisons contint 10 canaux, mais 4 canaux vont suffire pour piloter le drone.
Ces 4 sortie représentent le roulis (aileron), le tangage (elevator), le lacet (rudder), et l'accélérateur (throttle). Elles sont branchées respectivement au pins $8, 9, 10, 11$.
L'alimentation de ce récepteur est fourni depuis un régulateur de tension $5V$contenu dans le distributeur.
- Une LED témoin est branchée au PIN 13.

La figure IV.59 illustre ces branchements.

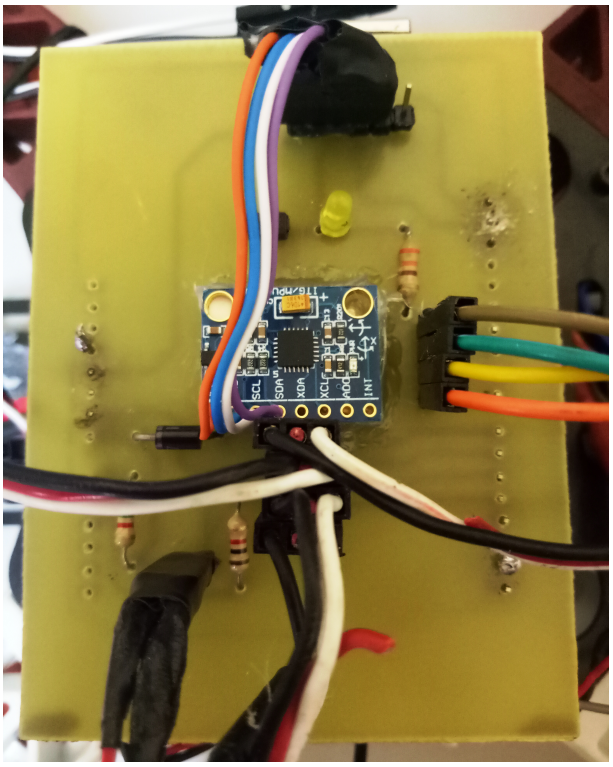

FIGURE IV.59 – Circuit de puissance

IV.4 Configuration et Calibration

Après avoir fini le montage il faut faire une calibration des capteurs et des ESC, et configurer le récepteur pour que notre contrôleur de vol puisse lire les signaux entrants.

Le projet YMFC-AL [4] offre deux programmes de calibrations, et un programme principal pour le contrôleur de vol.

Calibration de l'IMU

Pour la configuration du récepteur et la calibration du capteur il suffit juste de téléverser le programme de configuration(Setup), ouvrir le moniteur série à 57600 bauds et suivre les instruction demandées.

Une fois la procédure est terminée tous les paramètres sont stockés dans la mémoire EEPROM de l'Arduino, et le voyant est allumé pour indiquer que la calibration est terminée, donc il est temps de calibrer les ESC.

Calibration des ESC

Cette calibration est aussi simple, elle se fait avec les étapes suivantes :

- Téléverser le programme (ESC.Calibrate).
- Mettre l'accéléromètre à fond.
- Brancher la batterie.
- Après quelques signaux sonores on place la manette des gaz dans la position la plus basse.
- Déconnecter la batterie.

Le programme ESC.Calibrate offre la possibilité de vérifier les signaux envoyés par la radio-commande, ces signaux doivent avoir la valeur 1000 comme minimum et 2000 comme valeurs maximale.

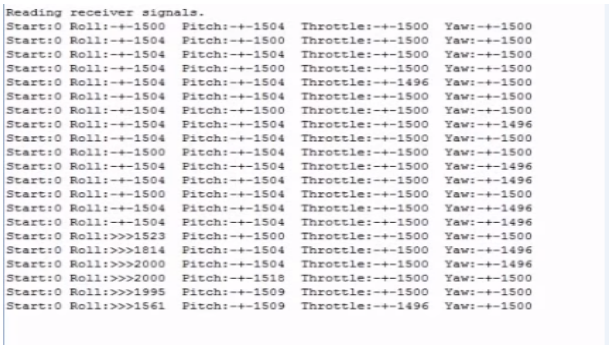

FIGURE IV.60 – Signaux envoyés par la radio-commande

4. Le YMFC-AL est un quadricoptère à nivellement automatique basé sur Arduino Uno. La mise à niveau automatique signifie que, lorsqu'on relâche les manettes, le quadricoptère se met à niveau.[2]

Il offre aussi la possibilité d'afficher les valeurs des angles calculés par le contrôleur de vol, et les vérifier avec l'orientation du quadrirotor.

```
Print the quadcopter angles.
Gyro calibration starts in 2 seconds (don't move the quadcopter).
Calibrating the gyro.....................
Pitch: 0 Roll: 0 Yaw: -0
Pitch: -0 Roll: 0 Yaw: 0
Pitch: -0 Roll: 0 Yaw: 0
Pitch: -0 Roll: 0 Yaw: -0
Pitch: -0 Roll: 0 Yaw: -0
Pitch: -0 Roll: 0 Yaw: 0
Pitch: -0 Roll: 0 Yaw: 0
Pitch: -0 Roll: 0 Yaw: -0
Pitch: -0 Roll: 0 Yaw: 0
Pitch: -0 Roll: 0 Yaw: -0
Pitch: -0 Roll: 0 Yaw: -0
Pitch: -0 Roll: 0 Yaw: 0
Pitch: -0 Roll: 0 Yaw: 0
Pitch: -0 Roll: 0 Yaw: 0
Pitch: -0 Roll: 0 Yaw: 0
Pitch: -0 Roll: 0 Yaw: 0
Pitch: -0 Roll: 0 Yaw: 0
Pitch: -0 Roll: 0 Yaw: -0
Pitch: -0 Roll: 0 Yaw: 0
Pitch: -0 Roll: 0 Yaw: -0
Pitch: -0 Roll: 0 Yaw: -0
Pitch: -0 Roll: 0 Yaw: -0
Pitch: -0 Roll: 0 Yaw: 0
```

FIGURE IV.61 – Angles calculés par le contrôleur de vol

La phase des calibrations est donc terminée, il faut maintenant téléverser le programme du controleur de vol, ajuster les régulateurs PID, et faire un premier test de vol.

IV.5 Régulation des PID

Dans ce projet nous utilisons un seul capteur (MPU6050) qui s'agit d'un accéléromètre/gyroscope, ce capteur est capable de calculer les angles de rotation selon les trois axes x, y, z.
Donc nous utilisons 3 régulateurs PID pour la boucle interne(ϕ, θ, ψ).
Les position X, Y, Z doivent être régulées manuellement à l'aide de la radio-commande.
Après avoir téléverser le programme principale de contrôleur de vol, les gains des PID sont déclarés comme des flottants au début du code comme le montre la figure IV.62.

```
////////////////////////////////////////////////////////////////////////////////////

#include <Wire.h>                 //Include the Wire.h library so we can communicate with the gyro.
#include <EEPROM.h>               //Include the EEPROM.h library so we can store information onto the EEPROM

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
//PID gain and limit settings
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
float pid_p_gain_roll = 1.3;              //Gain setting for the roll P-controller
float pid_i_gain_roll = 0.04;             //Gain setting for the roll I-controller
float pid_d_gain_roll = 18.0;             //Gain setting for the roll D-controller
int pid_max_roll = 400;                   //Maximum output of the PID-controller (+/-)

float pid_p_gain_pitch = 1.3;  //Gain setting for the pitch P-controller.
float pid_i_gain_pitch = 0.04;  //Gain setting for the pitch I-controller.
float pid_d_gain_pitch = 18;  //Gain setting for the pitch D-controller.
int pid_max_pitch = 400;                  //Maximum output of the PID-controller (+/-)

float pid_p_gain_yaw = 4.0;               //Gain setting for the pitch P-controller. //4.0
float pid_i_gain_yaw = 0.02;              //Gain setting for the pitch I-controller. //0.02
float pid_d_gain_yaw = 0.0;               //Gain setting for the pitch D-controller.
int pid_max_yaw = 400;                    //Maximum output of the PID-controller (+/-)
```

FIGURE IV.62 – Programme de contrôleur de vol

IV.6 Test de vol

Nous avons réussi à ajuster les paramètres des trois PID pour stabiliser le drone et rejeter les perturbations après plusieurs essais en utilisant la méthode d'essai-erreur, le tableau suivant contient les gains finaux des PID :

TABLE IV.5 – Tableau des gains PID

Paramètre	ϕ	θ	ψ
K_p	2.1	1.3	4
K_i	0.01	0.01	0.02
K_d	12	18	0

Nous remarquons que les paramètres des PID réels sont preque identiques au paramétrés de la simulation ce qui prouve que la partie identification a donné de bon résultat qui nous ont permis d'obtenir un comportement plus proche du modèle réel.

Au final, nous avons réussi à faire décoller notre drone, à effectuer un vol quasi-stationnaire et à contrôler sa position à partir de la radiocommande.

FIGURE IV.63 – Test du vol

IV.7 Acquisition des donnés

Pour analyser le comportement du quad-coptère pendent le vol, une récupération des données est nécessaire. Pour cela nous avons réalisé une télémétrie à l'aide d'un module XBee (voir l'annexe-2).

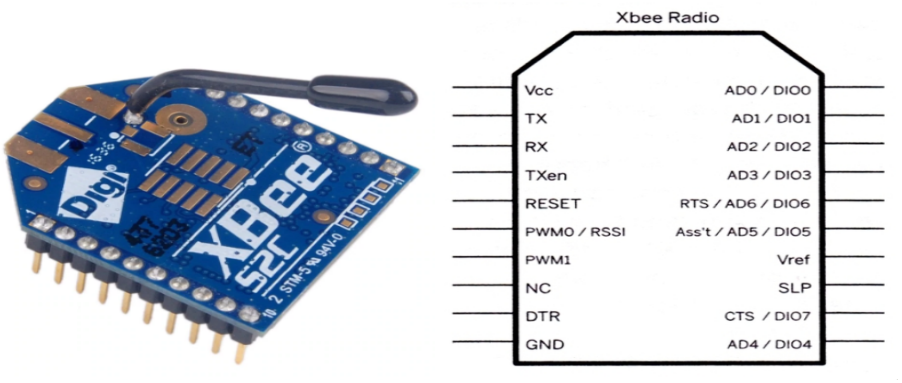

FIGURE IV.64 – Module XBee

Dans notre cas nous avons récupéré, puis tracé, les valeurs réelles et désirées des angles de roulis, tangage. Les courbes obtenues sont illustrées dans les figures suivante :

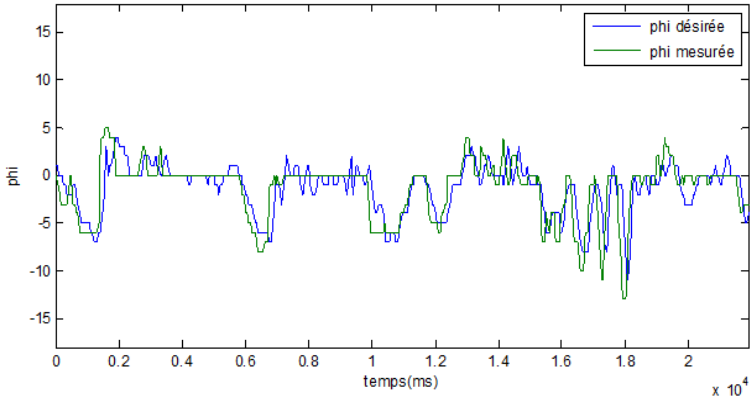

FIGURE IV.65 – Tracé de ϕ

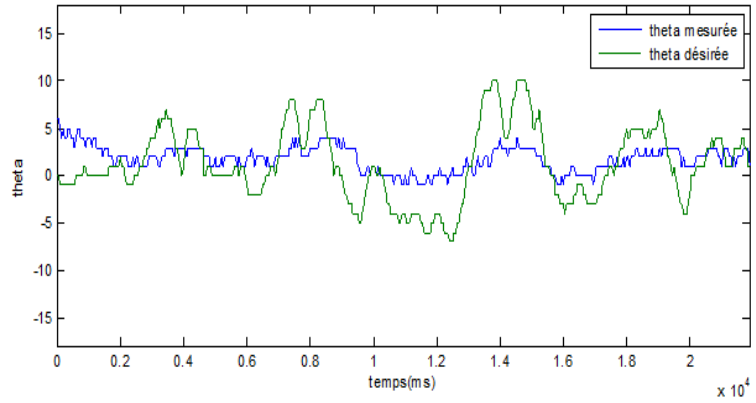

FIGURE IV.66 – Tracé de θ

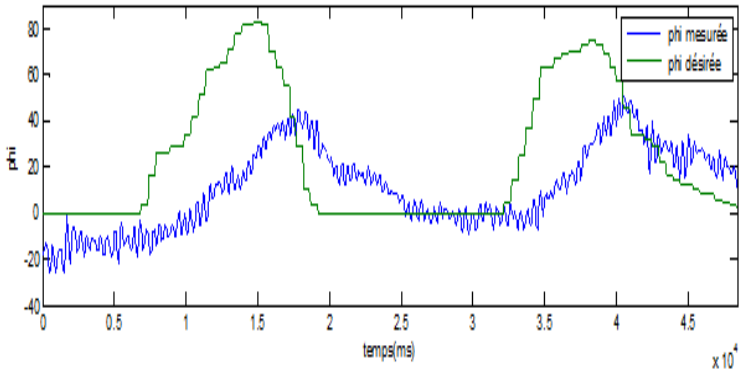

FIGURE IV.67 – Tracé de ψ

nous remarquons que l'angle ϕ suit correctement sa référence, avec un faible écart. Pour l'angle θ l'écart est élevé, cela est du à cause de la sensibilité du capteur $MPU6050$.
Autour de l'axe z il n'est pas possible de calculer un angle en degrés avec l'IMU que nous avons utilisé, donc nous avons tracé la vitesse angulaire autour de cet axe. On remarque le suivi de consigne avec un retard et les valeurs sont bruitées malgré l'utilisation d'un filtre complémentaire.

IV.8 Conclusion

Dans ce chapitre nous avons détaillé les différents choix des composants de notre drone quadrirotor. Ensuite, il était question de réaliser le montage avec ces composants en respectant les conventions dans les branchements. Une fois le montage final est réalisé, il a fallu configurer et calibrer le contrôleur de vol basé sur une carte Arduino Uno, et ajuster les paramètres de la commande pour garantir la stabilité du quad-coptère.
Finalement, et après plusieurs tentatives nous avons réussi à stabiliser notre drone grâce à une loi de commande robuste.

Conclusion générale

Avec la croissance de l'intérêt pour les UAV, celui des avions à atterrissage et à décollage verticaux (VTOL) et le besoin d'une instrumentation discrète et surtout légère, le quadrotor a connu une grande popularité ces dernières années. Cela en fait, la principale motivation pour ce travail de recherche. L'objectif principal du travail était d'adopter un modèle mathématique de complexité minimal et d'avoir des lois de commande de vol qui permettent d'assurer la stabilité d'un quadcopter, pour pouvoir ensuite appliquer cette loi de commande sur un modèle pratique.

Le but de la première partie était de présenter un état de l'art sur les drones et leurs différents types, parmi ces types nous nous sommes basés précisément sur les quadcopters, leurs avantages et leurs domaines d'applications. Ensuite dans le deuxième chapitre nous avons entamé la modélisation du quadrirotor qui est une étape indispensable à la bonne compréhension des lois de la physique qui régissent notre système. L'exploitaion du modèle mathématique qui définit la dynamique du quad-copter, nous permet de déduire le modèle d'état.

Ensuite, nous nous sommes intéressés à l'identification des paramètres du quadrirotor pour pouvoir contrôler les différents mouvements en utilisant la technique PID. De bon résultats de simulation sous Simulink/Matlab ont été obtenus en adoptant la méthode essai-erreur pour le réglage des paramètres des correcteurs PID. Dans la dernière partie, nous avons choisi les composants nécessaires pour la construction de notre quadrirotor. L'implementation de la commande PID sur le drone est faite par la carte élémentaire Arduino uno. Cette carte réduit considérablemment le coût de l'application. Elle n'est pas aussi performante qu'une carte de vol professionnelle mais a permis malgré tout d'obtenir des résultats satisfaisants.

Le tracé des trajectoires de vols a été réalisé en récuperant les informations de vol par transmission XBee.
Des tests en indoor et outdoor ont été effectués pour valider la procédure.

Ce travail est une continuation du travail effectué dans le cadre d'un projet de fin d'étude précédent. Dans lequel, les auteurs avaient utilisé la carte de vol auto

pilote Pixhawk. Dans ce travail la partie identification a été améliorée, les paramètres des PID utilisés en simulation sont pratiquement identiques à ceux des tests pratiques. La carte Arduino a permis de réduire les coûts, de simplifier l'implémentation et de préparer l'implémentation de d'autres lois de commande ce qui est plus facile par rapport à la carte Pixhawk.

Cette carte n'a permis pour le moment la commande que de la boucle interne à défaut de l'utilisation d'un GPS qui pourrait aider à commander la boucle externe et d'une meilleure centrale inertiele. Ceci pourrait faire l'objet de travaux ultérieurs.

Annexe : Description du modèle Simulink

Dans cette annexe nous allons détailler le modèle simulink et décrire le contenue de chaque bloc.

Bloc des entrées

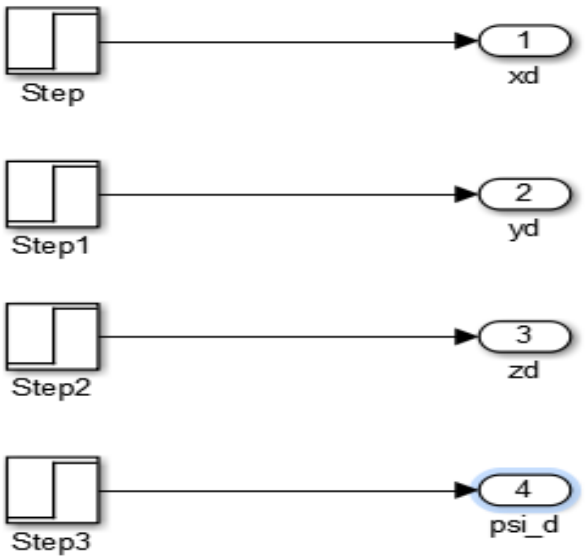

FIGURE IV.68 – Trajectoires désirées

Les quatre consignes sont des échelons unitaires pour les trajectoires désirées de X, Y, Z et ψ.

Bloc de régulation des positions

Ce bloc contient deux opérations comme le montre la figure suivante :

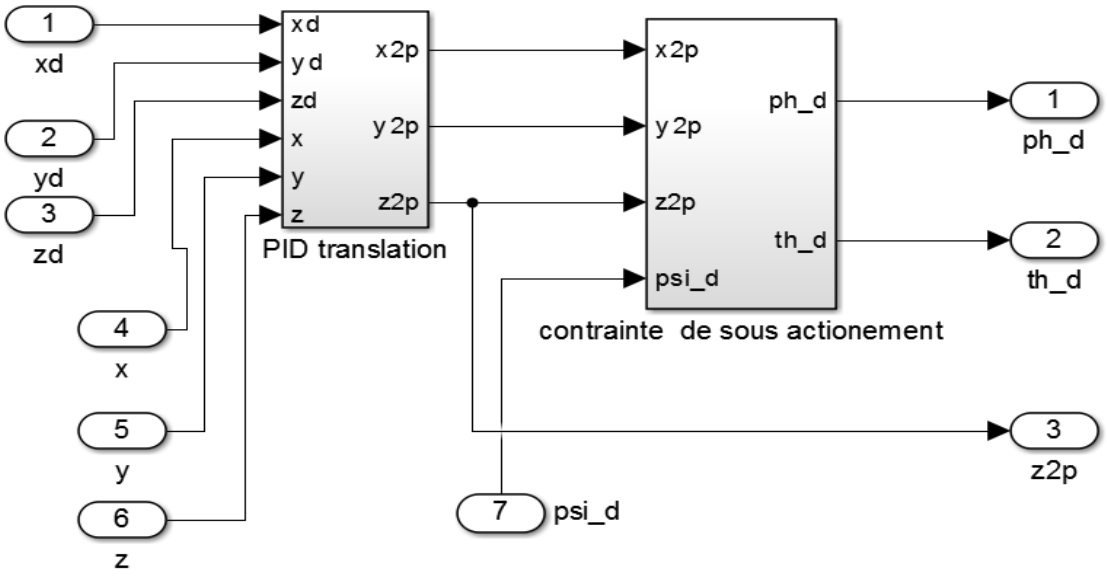

FIGURE IV.69 – Régulation des positions

Premièrement les positions désirées sont comparées avec les valeur réelles pour être réguler par PID.

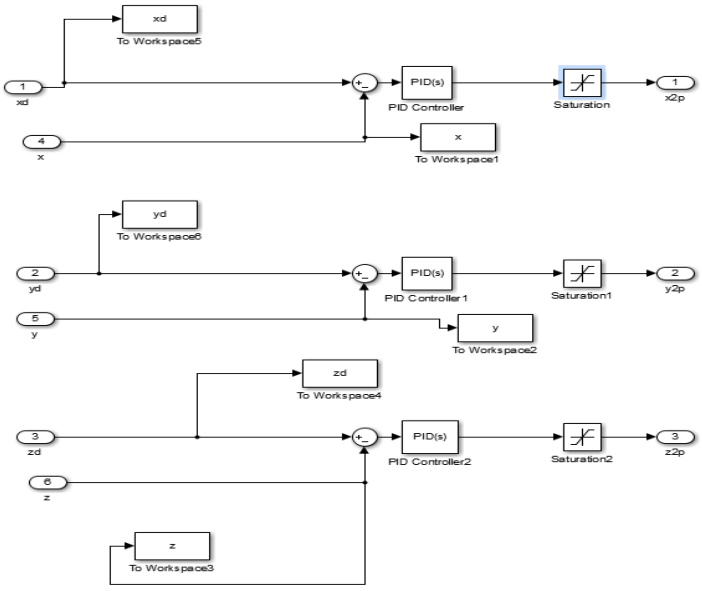

FIGURE IV.70 – PID de translations

La saturation est utilisée pour protéger les moteurs, elle est limitée à $-28 < \ddot{x} < 28$.

Ensuite les sortie des PID sont exploitées avec l'angle ψ_d pour déterminer les angles ϕ_d et θ_d par leur expressions.

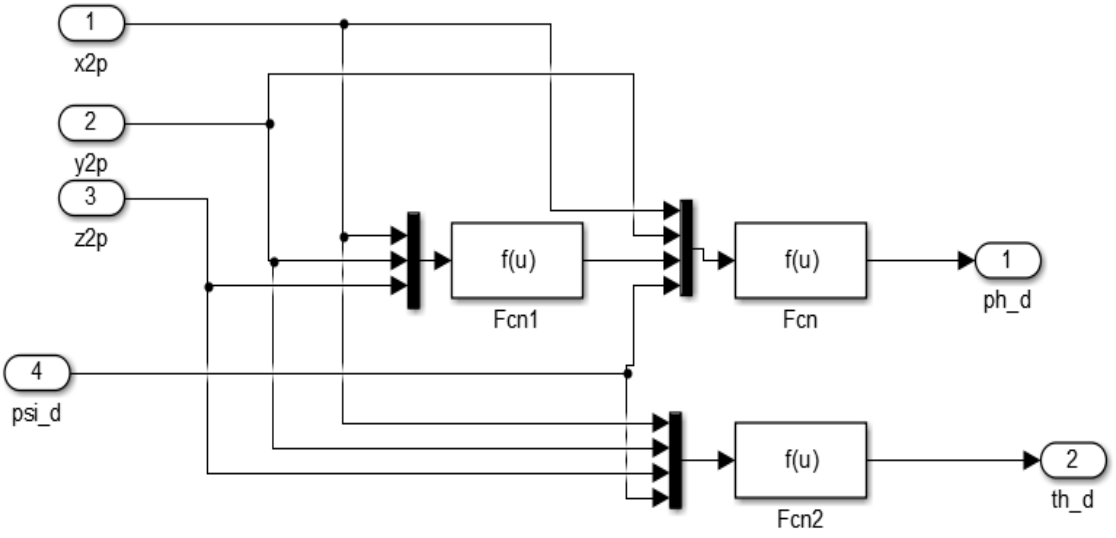

FIGURE IV.71 – Contraintes de sous-actionnement

Les fonction Fcn et Fcn2 contiennent les équations III.53 et III.49 respectivement.

Bloc de régulation des angles

Les angles ϕ_d, θ_d, ψ_d sortants du bloc précédent sont régulés par des régulateurs PID, figure IV.72.

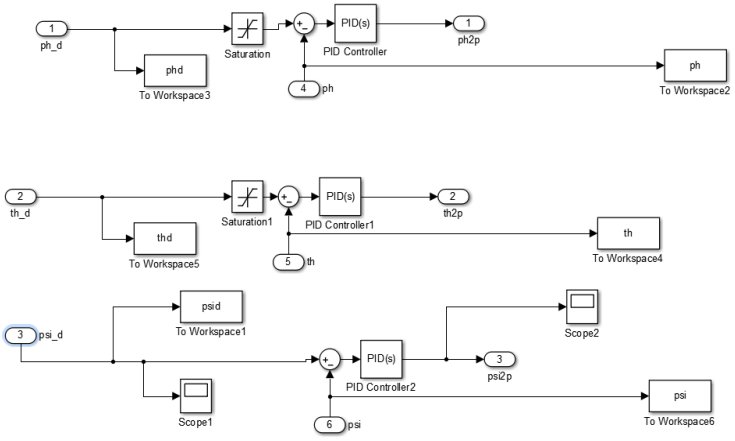

FIGURE IV.72 – PID de rotation

Les angles θ_d et ϕ_d sont limités entre $\frac{-\pi}{6}$ et $\frac{\pi}{6}$.

Bloc des commandes

Ce bloc est une combinaison entre les commandes U_1, U_2, U_3, U_4 et les vitesses de rotation $\omega 1, \omega 2, \omega 3, \omega 4$.

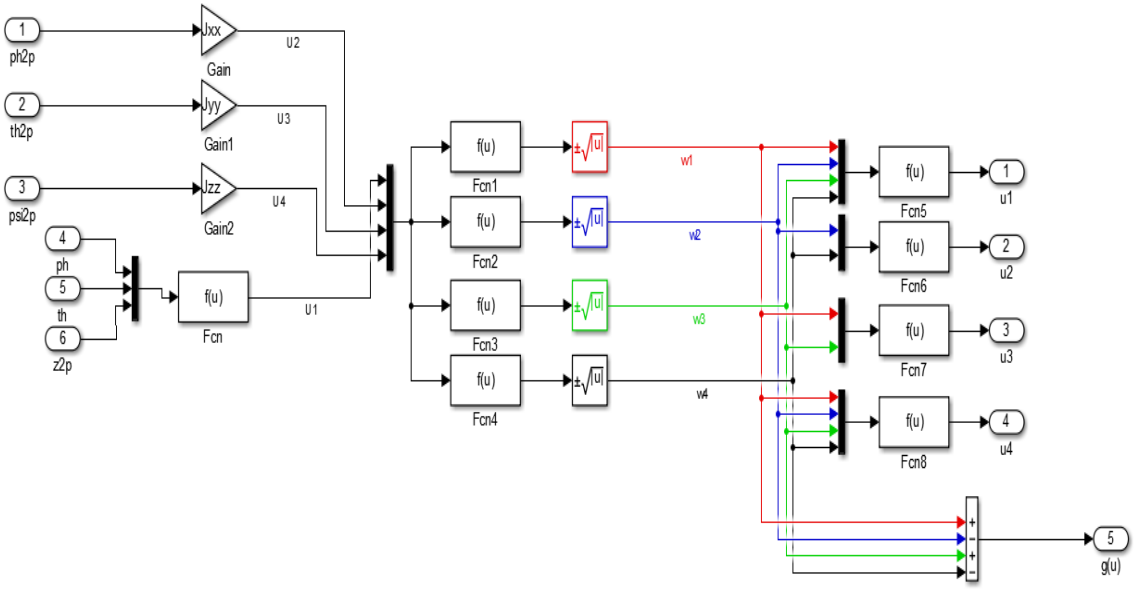

FIGURE IV.73 – Bloc des commandes

Les quatre commandes sont d'abord exprimées à partir des equations III.40 et III.41.
Ensuite en utilisant l'équation III.42 nous déterminons les vitesses de rotations des moteurs.
Enfin les commandes sont reconstruites à partir de ces vitesses par l'équation II.32.

Bloc de quadrirotor

Ce bloc représente le modèle dynamique du quad-coptère, il est composé de deux sous-système : le premier pour les mouvements de rotation et le deuxième pour les mouvements de translation.

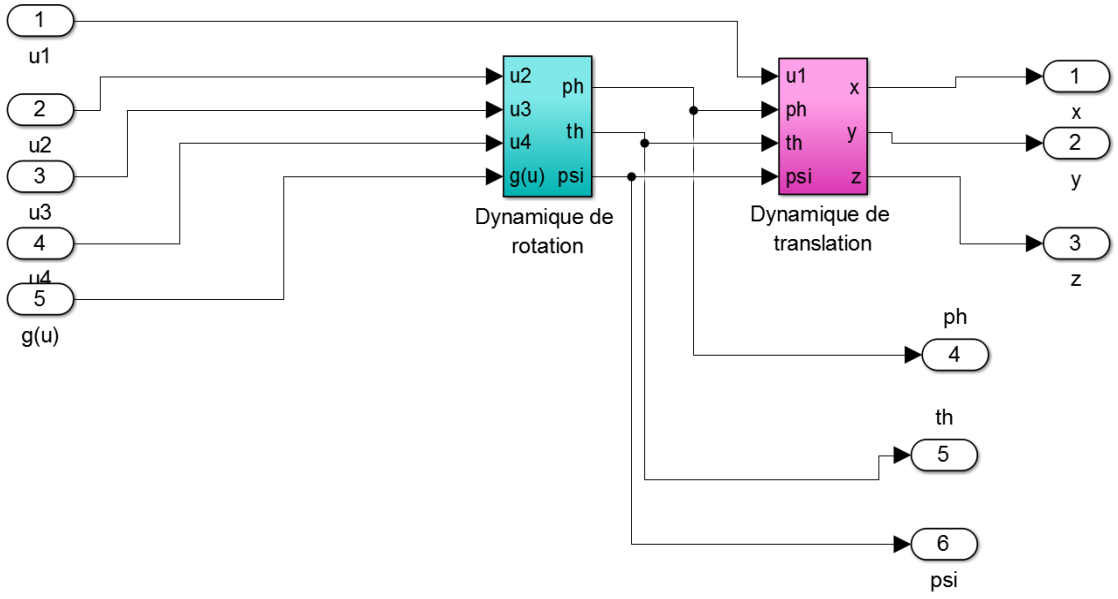

FIGURE IV.74 – Bloc de la dynamique du quadrirotor

Dynamique de rotation

Le modèle dynamique de rotation est traduit en schéma simulink suivant :

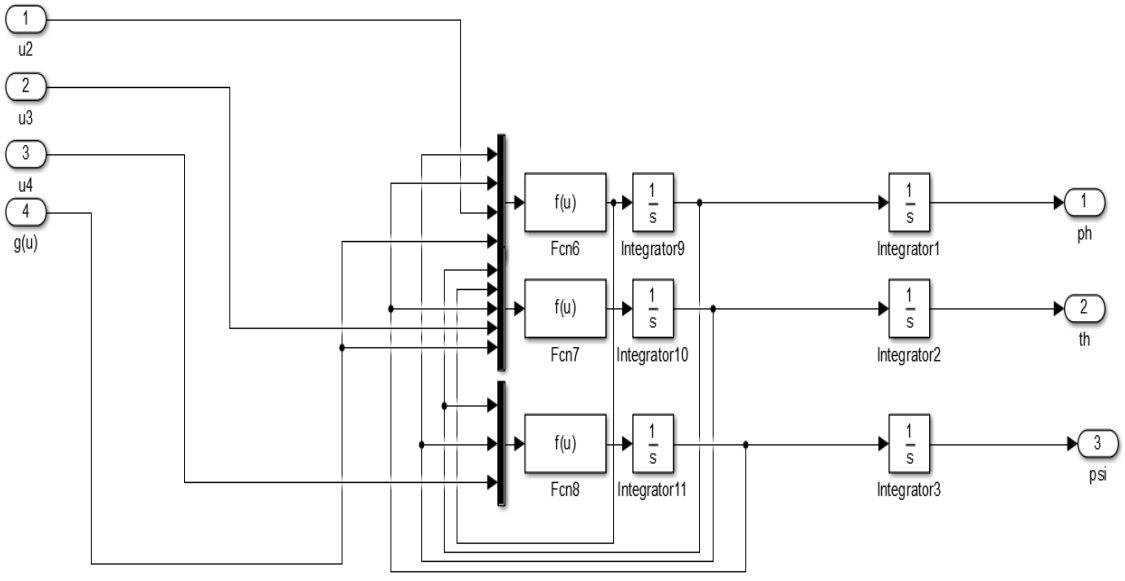

FIGURE IV.75 – Dynamique de rotation

Les fonctions $Fcn6$, $Fcn7$, $Fcn8$ contiennent les equations dynamique de $\ddot{\phi}$, $\ddot{\theta}$, et $\ddot{\psi}$ respectivement.

Dynamique de translation

Le modèle dynamique de translation est traduit en schéma simulink suivant :

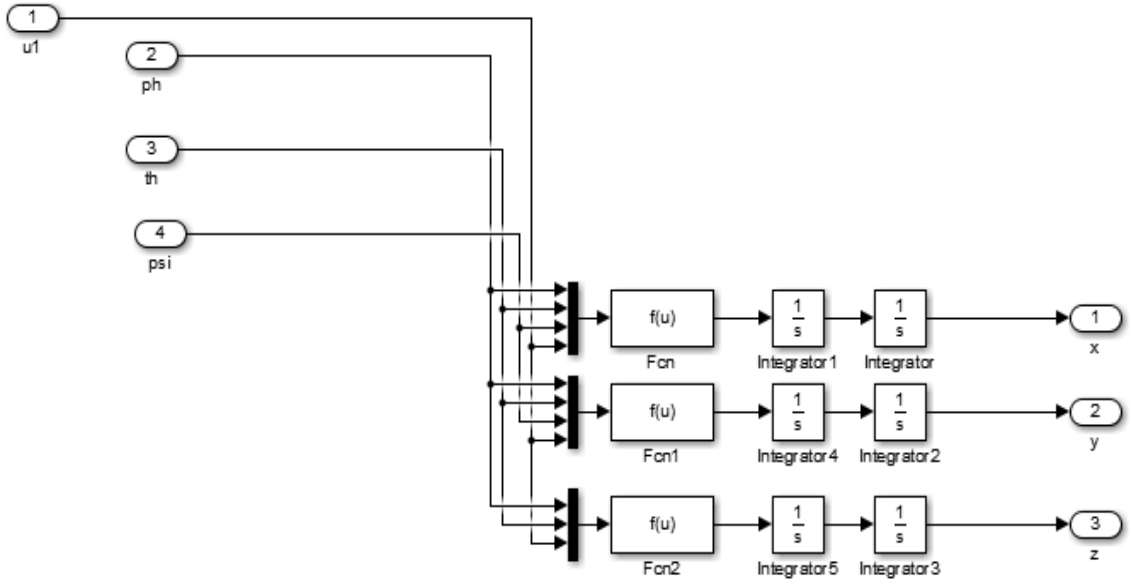

FIGURE IV.76 – Dynamique de translation

Les fonctions Fcn, $Fcn1$, $Fcn2$ contiennent les equations dynamique de $\ddot{x}$, $\ddot{y}$, et $\ddot{z}$ respectivement.

Annexe : Description et utilisation du module Xbee

Le XBee est un microcontrôleur sans fil fabriqué par Digi qui utilise un émetteur-récepteur sans fil 2.4 GHz pour communiquer avec un autre module.
Les modules XBee permettent d'envoyer et de recevoir des données, sans fil, performants et accessibles. Ils sont caractérisés par une portée tr$\hat{e}$s confortable d'une centaine de mètres en environnement d'intérieur, et jusqu'à plus d'un kilomètre en zone dégagée pour les modules XBee-PRO équipés d'une antenne adaptée.
Ils peuvent être utilisés couplés à un microcontrôleur ou de façon indépendante.
Ils sont très pratiques pour la réalisation de nombreux montages électroniques qui doivent pouvoir communiquer entre eux.

Spécifications

Alimentation : 3,3 Vcc (50 mA)
XBee série 1
Puissance : 1 mW
Débit HF : 250 kbps
Débit série : jusqu'à 115,2 kbps
Portée moyenne :
- 30 m en intérieur
- 100 m en extérieur
Sensibilité en réception : -92 dBm
Fréquence : 2,4 GHz
Dimensions : $28 \times 25 \times 10$ mm
T° de service : -40 à +85 C°
Poids : 3 g

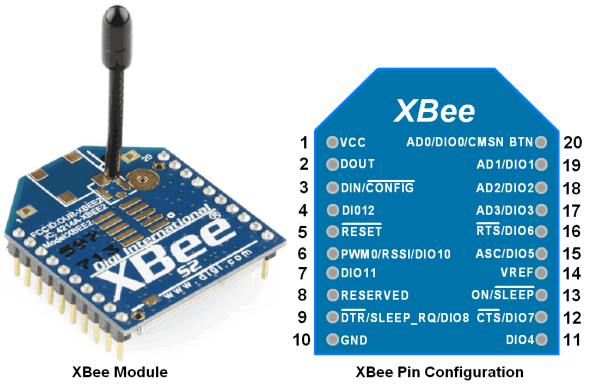

FIGURE IV.77 – Module XBee

Configuration et communication

La configuration des deux modules XBee (émetteur et récepteur) exige l'utilisation d'un support USB XBee Explorer figure IV.78 et l'installation du logiciel XCTU [5].

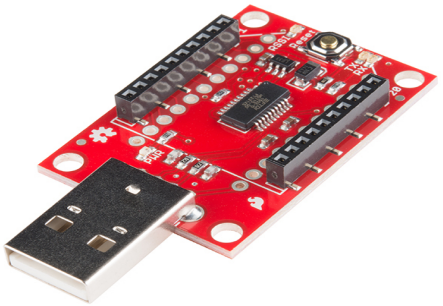

FIGURE IV.78 – Support USB XBee Explorer

La procédure de configuration est la suivante :
- Exécuter le programme et connecter le support USB XBee Explorer à l'ordinateur.
- Cliquer sur l'icône "Discover devices" pour ajouter notre XBee dans le logiciel XCTU.
- Cliquer sur le module qui s'affiche à gauche et régler le champ CH sur "C", et le champ ID sur "1001", par exemple. Ces valeurs doivent être identiques pour tous les modules XBee puissent communiquer entre eux.
- Définir le champ CE comme "Coordonator". Le débit en bauds doit être défini sur 9600bps.
- Cliquer sur le bouton "Write" pour enregistrer les modifications dans notre émetteur XBee.

5. XCTU est une application multiplate-forme gratuite conçue pour permettre aux développeurs d'interagir avec les modules Digi RF via une interface graphique simple Ã utiliser. https://www.digi.com/products/iot-platform/xctu

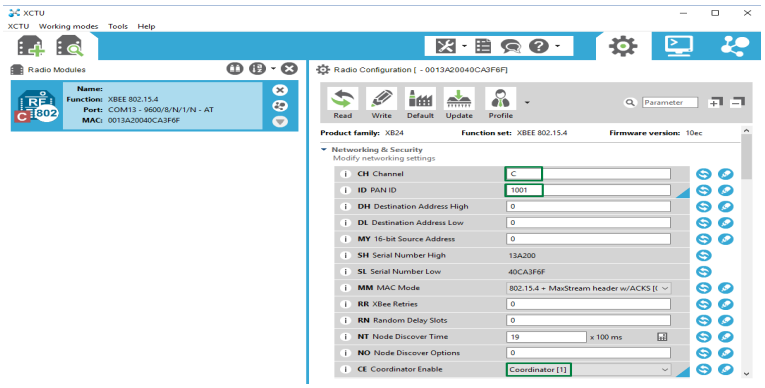

FIGURE IV.79 – Configuration de l'émetteur

- Déconnecter la carte XBee Explorer, et connectez l'autre module dessus.
- Connecter à nouveau la carte de l'explorateur à l'ordinateur et suivre la même procédure, sauf que cette fois il faut définir le champ CE comme "End device".

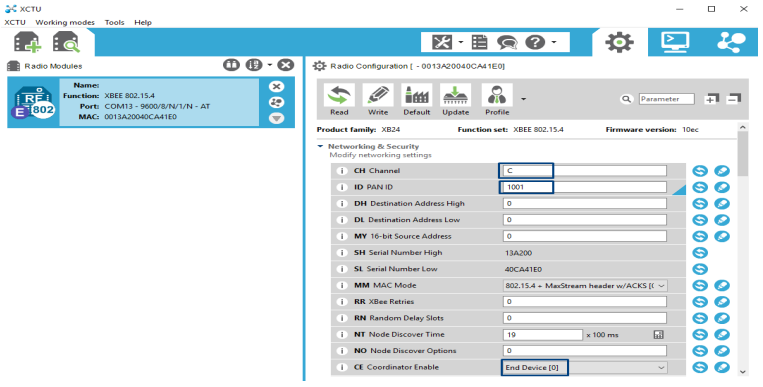

FIGURE IV.80 – Configuration du récepteur

L'émetteur est connecté à la carte Arduino comme la figure suivante le montre :

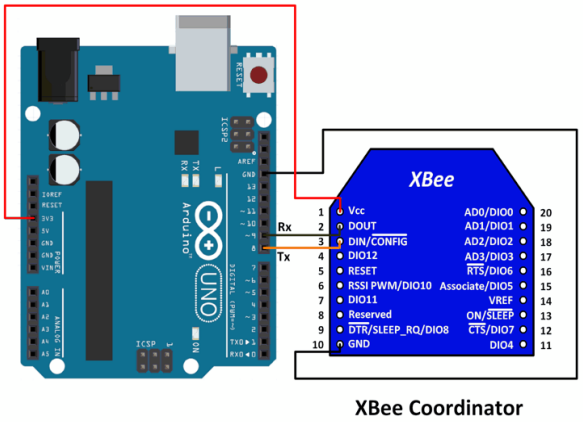

FIGURE IV.81 – Connexion de l'XBee avec Arduino

Annexe : Explication du programme pour le contrôleur de vol

Avant d'expliquer les différentes étapes du programme principal il faut d'abord comprendre comment notre contrôleur de vol va fonctionner.

Principe de fonctionnement du contrôleur de vol

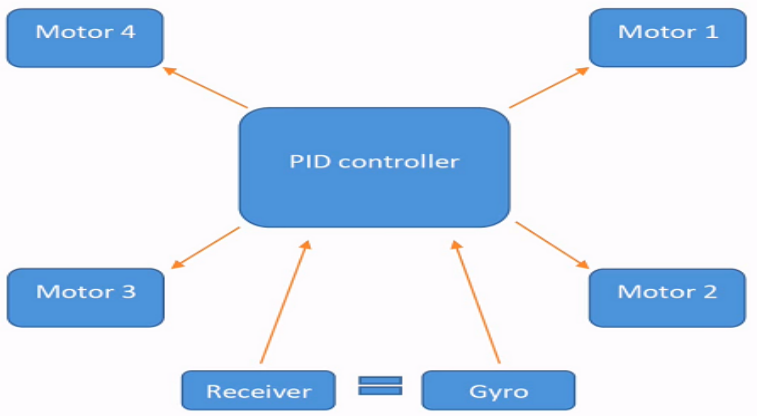

FIGURE IV.82 – Schéma représentant le principe de fonctionnement du contrôleur de vol

Tout d'abord, l'erreur s'agit d'une différence entre les valeurs réelles envoyées par le capteur, et les consignes envoyées par la radio-commande.
Cette erreur est utilisée ensuite pour le calcul des PID.
Les résultats du calcul vont construire les impulsions pour les ESC, qui vont à leur tour commander les moteurs.

Lecture et conversion des données

Données envoyées par le capteur

La lecture des données se fait par la partie du code illustrée dans la figure IV.83.

```
void gyro_signalen(){
  //Read the MPU-6050
  if(eeprom_data[31] == 1){
    Wire.beginTransmission(gyro_address);
    Wire.write(0x3B);
    Wire.endTransmission();
    Wire.requestFrom(gyro_address,14);

    receiver_input_channel_1 = convert_receiver_channel(1);
    receiver_input_channel_2 = convert_receiver_channel(2);
    receiver_input_channel_3 = convert_receiver_channel(3);
    receiver_input_channel_4 = convert_receiver_channel(4);

    while(Wire.available() < 14);
    acc_axis[1] = Wire.read()<<8|Wire.read();
    acc_axis[2] = Wire.read()<<8|Wire.read();
    acc_axis[3] = Wire.read()<<8|Wire.read();
    temperature = Wire.read()<<8|Wire.read();
    gyro_axis[1] = Wire.read()<<8|Wire.read();
    gyro_axis[2] = Wire.read()<<8|Wire.read();
    gyro_axis[3] = Wire.read()<<8|Wire.read();
  }
```

FIGURE IV.83 – Lecture des données envoyées par l'IMU

Les valeurs envoyées par le capteur $MPU6050$ sont des valeurs brutes, pour cela il est nécessaire de les convertir avant de les utiliser, les trois ligne du programme dans la figure ci-dessous vont déterminer les entrréelles en degrés/secondes [6] pour le calcul de l'erreur.

```
//65.5 = 1 deg/sec (check the datasheet of the MPU-6050 for more information).
gyro_roll_input = (gyro_roll_input * 0.7) + ((gyro_roll / 65.5) * 0.3);   //Gyro pid input is deg/sec.
gyro_pitch_input = (gyro_pitch_input * 0.7) + ((gyro_pitch / 65.5) * 0.3);//Gyro pid input is deg/sec.
gyro_yaw_input = (gyro_yaw_input * 0.7) + ((gyro_yaw / 65.5) * 0.3);      //Gyro pid input is deg/sec.
```

FIGURE IV.84 – Conversion en degrés/secondes

Les figures IV.85, IV.86 illustrent la partie responsable à la conversion en degrés, tout en exploitant les données fournis par le gyroscope et l'accéléromètre.

```
//Gyro angle calculations
//0.0000611 = 1 / (250Hz / 65.5)
angle_pitch += gyro_pitch * 0.0000611;                                   //Calculate the traveled pitch angle and add this to the angle_pit
angle_roll += gyro_roll * 0.0000611;                                     //Calculate the traveled roll angle and add this to the angle_roll

//0.000001066 = 0.0000611 * (3.142(PI) / 180degr) The Arduino sin function is in radians
angle_pitch -= angle_roll * sin(gyro_yaw * 0.000001066);                 //If the IMU has yawed transfer the roll angle to the pitch angel.
angle_roll += angle_pitch * sin(gyro_yaw * 0.000001066);                 //If the IMU has yawed transfer the pitch angle to the roll angel.

//Accelerometer angle calculations
acc_total_vector = sqrt((acc_x*acc_x)+(acc_y*acc_y)+(acc_z*acc_z));      //Calculate the total accelerometer vector.

if(abs(acc_y) < acc_total_vector){                                       //Prevent the asin function to produce a NaN
  angle_pitch_acc = asin((float)acc_y/acc_total_vector)* 57.296;         //Calculate the pitch angle.
}
if(abs(acc_x) < acc_total_vector){                                       //Prevent the asin function to produce a NaN
  angle_roll_acc = asin((float)acc_x/acc_total_vector)* -57.296;         //Calculate the roll angle.
}

//Place the MPU-6050 spirit level and note the values in the following two lines for calibration.
angle_pitch_acc -= 0.0;                                                  //Accelerometer calibration value for pitch.
angle_roll_acc -= 0.0;                                                   //Accelerometer calibration value for roll.
```

FIGURE IV.85 – Conversion en degrés

6. D'après le datasheet de l'MPU6050 : 65.5 de valeurs brutes correspond à $1d/s$

```
angle_pitch = angle_pitch * 0.9996 + angle_pitch_acc * 0.0004;
angle_roll = angle_roll * 0.9996 + angle_roll_acc * 0.0004;

pitch_level_adjust = angle_pitch * 15;
roll_level_adjust = angle_roll * 15;

if(!auto_level){
  pitch_level_adjust = 0;
  roll_level_adjust = 0;
}
```

FIGURE IV.86 – Conversion en degrés

Données envoyées par la radio-commande

Les contrôleur de vitesse sont commandés par des impulsion de $1000us$ comme valeur minimale, et $2000us$ comme valeur maximale. Ces impulsions vont déterminer les vitesses des moteurs brushless qui sont proportionnelles au nombre de KV fois la tension fournie par la batterie ($\omega = KV \times U$).
Ce qui fait que les signaux envoyés par la raio-commande diffèrent entre 1000 et 2000 us.
De plus les quatre entrés du récepteur doivent être déclarer comme des interruptions qui vont interrompre le programme à chaque changement d'état.

Les figures ci-dessous représentent la configuration et la lecture des quatre signaux d'entré.

```
PCICR |= (1 << PCIE0);
PCMSK0 |= (1 << PCINT0);
PCMSK0 |= (1 << PCINT1);
PCMSK0 |= (1 << PCINT2);
PCMSK0 |= (1 << PCINT3);
```

FIGURE IV.87 – Configuration des quatre entrées de récepteur comme des interruptions

```
ISR(PCINT0_vect){
  current_time = micros();
  //Channel 1=========================================
  if(PINB & B00000001){
    if(last_channel_1 == 0){
      last_channel_1 = 1;
      timer_1 = current_time;
    }
  }
  else if(last_channel_1 == 1){
    last_channel_1 = 0;
    receiver_input[1] = current_time - timer_1;
  }
  //Channel 2=========================================
  if(PINB & B00000010 ){
    if(last_channel_2 == 0){
      last_channel_2 = 1;
      timer_2 = current_time;
```

FIGURE IV.88 – Détection de chaque changement d'état des quatre canaux

```
int convert_receiver_channel(byte function){
  byte channel, reverse;
  int low, center, high, actual;
  int difference;

  channel = eeprom_data[function + 23] & 0b00000111;
  if(eeprom_data[function + 23] & 0b10000000)reverse = 1;
  else reverse = 0;

  actual = receiver_input[channel];
  low = (eeprom_data[channel * 2 + 15] << 8) | eeprom_data[channel * 2 + 14];
  center = (eeprom_data[channel * 2 - 1] << 8) | eeprom_data[channel * 2 - 2];
  high = (eeprom_data[channel * 2 + 7] << 8) | eeprom_data[channel * 2 + 6];

  if(actual < center){
    if(actual < low)actual = low;
    difference = ((long)(center - actual) * (long)500) / (center - low);
    if(reverse == 1)return 1500 + difference;
    else return 1500 - difference;
  }
```

FIGURE IV.89 – Conversion des signaux

```
  else if(actual > center){
    if(actual > high)actual = high;
    difference = ((long)(actual - center) * (long)500) / (high - center);
    if(reverse == 1)return 1500 - difference;
    else return 1500 + difference;
  }
  else return 1500;
}
```

FIGURE IV.90 – Conversion des signaux

Maintenant que nous avons les signaux envoyés par la radio-commande, nous pouvons définir les entrés désirées pour le calcul de l'erreur. Pour qu'elles soient homogènes avec les valeurs réelles, ces entrées sont aussi convertis en degrés/secondes, cela se fait au niveau du code illustré dans les figures IV.91 IV.92.

```
//The PID set point in degrees per second is determined by the roll receiver input.
//In the case of deviding by 3 the max roll rate is aprox 164 degrees per second ( (500-8)/3 = 164d/s ).
pid_roll_setpoint = 0;
//We need a little dead band of 16us for better results.
if(receiver_input_channel_1 > 1508)pid_roll_setpoint = receiver_input_channel_1 - 1508;
else if(receiver_input_channel_1 < 1492)pid_roll_setpoint = receiver_input_channel_1 - 1492;

pid_roll_setpoint -= roll_level_adjust;                    //Subtract the angle correction from the standardized receiver roll
pid_roll_setpoint /= 3.0;                                  //Divide the setpoint for the PID roll controller by 3 to get angle


//The PID set point in degrees per second is determined by the pitch receiver input.
//In the case of deviding by 3 the max pitch rate is aprox 164 degrees per second ( (500-8)/3 = 164d/s ).
pid_pitch_setpoint = 0;
//We need a little dead band of 16us for better results.
if(receiver_input_channel_2 > 1508)pid_pitch_setpoint = receiver_input_channel_2 - 1508;
else if(receiver_input_channel_2 < 1492)pid_pitch_setpoint = receiver_input_channel_2 - 1492;

pid_pitch_setpoint -= pitch_level_adjust;                  //Subtract the angle correction from the standardized receiver pit
pid_pitch_setpoint /= 3.0;                                 //Divide the setpoint for the PID pitch controller by 3 to get ang
```

FIGURE IV.91 – Définition des consignes pour le calcul de l'erreur

```
//The PID set point in degrees per second is determined by the yaw receiver input.
//In the case of deviding by 3 the max yaw rate is aprox 164 degrees per second ( (500-8)/3 = 164d/s ).
pid_yaw_setpoint = 0;
//We need a little dead band of 16us for better results.
if(receiver_input_channel_3 > 1050){ //Do not yaw when turning off the motors.
  if(receiver_input_channel_4 > 1508)pid_yaw_setpoint = (receiver_input_channel_4 - 1508)/3.0;
  else if(receiver_input_channel_4 < 1492)pid_yaw_setpoint = (receiver_input_channel_4 - 1492)/3.0;
}
```

FIGURE IV.92 – Définition des consignes pour le calcul de l'erreur

Calcul des PID

Les sortie des PID s'agit d'une somme des trois actions proportionnelle, intégrale et dérivée.

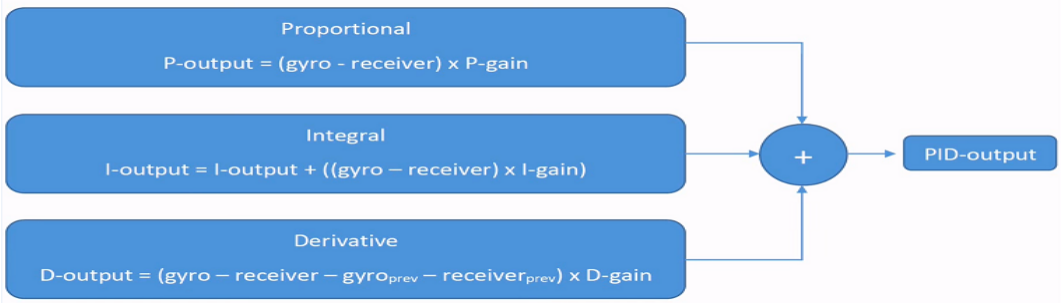

FIGURE IV.93 – Calcul des sorties des PID

- L'action P est déterminée par le produit de l'erreur fois le gain K_p.
- Pour l'action I, l'erreur est multipliée par un gain K_i, puis ajoutée à la sortie calculée précédemment.
- Finalement, l'action D est déterminée par la différence entre l'erreur actuelle et l'erreur précédente, fois un gain K_d.

La figure ci-dessous représente l'implémentation de ces trois calculs dans le programme pour le mouvement du roulis.

```
void calculate_pid(){
  //Roll calculations
  pid_error_temp = gyro_roll_input - pid_roll_setpoint;
  pid_i_mem_roll += pid_i_gain_roll * pid_error_temp;
  if(pid_i_mem_roll > pid_max_roll)pid_i_mem_roll = pid_max_roll;
  else if(pid_i_mem_roll < pid_max_roll * -1)pid_i_mem_roll = pid_max_roll * -1;

  pid_output_roll = pid_p_gain_roll * pid_error_temp + pid_i_mem_roll + pid_d_gain_roll * (pid_error_temp - pid_last_roll_d_error);
  if(pid_output_roll > pid_max_roll)pid_output_roll = pid_max_roll;
  else if(pid_output_roll < pid_max_roll * -1)pid_output_roll = pid_max_roll * -1;

  pid_last_roll_d_error = pid_error_temp;
```

FIGURE IV.94 – Implémentation du calcul des PID

Le même calcul est fait pour les deux autres mouvements.

Calcul d'impulsions pour les ESC

Les sorties des trois PID sont utilisées pour calculer les impulsion nécessaires au contrôleurs de vitesse pour commander les moteurs, chaque impulsion est définie par le signal envoyé par la manette du gaz (trottle), et une addition, ou soustraction [7] de de chaque sortie des PID calculées.
Le calcul des quatre impulsions est représenté dans la figure suivante :

```
if (start == 2){
  if (throttle > 1800) throttle = 1800;
  esc_1 = throttle - pid_output_pitch + pid_output_roll - pid_output_yaw;
  esc_2 = throttle + pid_output_pitch + pid_output_roll + pid_output_yaw;
  esc_3 = throttle + pid_output_pitch - pid_output_roll - pid_output_yaw;
  esc_4 = throttle - pid_output_pitch - pid_output_roll + pid_output_yaw;
```

FIGURE IV.95 – Calcul des impulsions pour les ESC

Calcul et compensation du voltage

Pendent le vol le drone consomme de la puissance, ce qui fait que le voltage fourni par la batterie va diminué, ce qui provoque la perte en altitude du véhicule. Pour éviter que cela se passe, le contrôleur de vol a besoin de savoir l'état de la batterie durant le vol. Les figures ci-dessous représentent le calcul et la compensation de la chute du voltage.

```
//Load the battery voltage to the battery_voltage variable.
//65 is the voltage compensation for the diode.
//12.6V equals ~5V @ Analog 0.
//12.6V equals 1023 analogRead(0).
//1260 / 1023 = 1.2317.
//The variable battery_voltage holds 1050 if the battery voltage is 10.5V.
battery_voltage = (analogRead(0) + 65) * 1.2317;
```

FIGURE IV.96 – Calcul de voltage

Le calcul se fait en une seule ligne avec une conversion analogique/numérique, et cela grâce au diviseur du tension branché à la broche A0.

7. Cela dépend de la commande désirée pour chaque moteur : signe + veut dire plus du temps d'impulsion, donc plus de vitesse. Et le contraire en cas de signe −

```
if (battery_voltage < 1240 && battery_voltage > 800){
  esc_1 += esc_1 * ((1240 - battery_voltage)/(float)3500);
  esc_2 += esc_2 * ((1240 - battery_voltage)/(float)3500);
  esc_3 += esc_3 * ((1240 - battery_voltage)/(float)3500);
  esc_4 += esc_4 * ((1240 - battery_voltage)/(float)3500);
}
```

FIGURE IV.97 – Compensation du voltage

Envoi des impulsions au ESC

Il existe plusieurs méthodes pour envoyer des impulsion au ESC, la plus simple c'est de créer des temporisation égaux au temps d'impulsion calculé précédemment.

```
//The refresh rate is 250Hz. That means the esc's need there pulse every 4ms.
while(micros() - loop_timer < 4000);                                   //We wait until 4000us are passed.
loop_timer = micros();                                                 //Set the timer for the next loop.

PORTD |= B11110000;                                                    //Set digital outputs 4,5,6 and 7 high.
timer_channel_1 = esc_1 + loop_timer;                                  //Calculate the time of the faling edge of the esc-1 pulse.
timer_channel_2 = esc_2 + loop_timer;                                  //Calculate the time of the faling edge of the esc-2 pulse.
timer_channel_3 = esc_3 + loop_timer;                                  //Calculate the time of the faling edge of the esc-3 pulse.
timer_channel_4 = esc_4 + loop_timer;                                  //Calculate the time of the faling edge of the esc-4 pulse.

//There is always 1000us of spare time. So let's do something usefull that is very time consuming.
//Get the current gyro and receiver data and scale it to degrees per second for the pid calculations.
gyro_signalen();

while(PORTD >= 16){                                                    //Stay in this loop until output 4,5,6 and 7 are low.
  esc_loop_timer = micros();                                           //Read the current time.
  if(timer_channel_1 <= esc_loop_timer)PORTD &= B11101111;             //Set digital output 4 to low if the time is expired.
  if(timer_channel_2 <= esc_loop_timer)PORTD &= B11011111;             //Set digital output 5 to low if the time is expired.
  if(timer_channel_3 <= esc_loop_timer)PORTD &= B10111111;             //Set digital output 6 to low if the time is expired.
  if(timer_channel_4 <= esc_loop_timer)PORTD &= B01111111;             //Set digital output 7 to low if the time is expired.
```

FIGURE IV.98 – Envoi des impulsions

Bibliographie

[1] A.CHRIETTE. *DRMMC : Drones autonomes : Définition et Classification.* Ecole Centrale de Nantes â Département AutoâRobot, 2012.

[2] Joop Brokking. Project YMFC-AL The Arduino auto-level quadcopter. http ://www.brokking.net/ymfc-al$_d$*ownloads.html*, *janvier*2016.

[3] Forum Drone. Calculer l'angle de l'hélice. http ://forumdrone.fr/topic/6667-calculer-langle-de-lhélice/, 2015.

[4] Z.ChEKAKTA et A.ZOUBIRI. *Conception, Modélisation et Commande d'un UAV de type Quadrirotor.* Projet de Fin d'étude Pour lâObtention du Diplôme dâIngénieur dâEtat. Ecole Nationale Polytechnique dâOran, 21 juin 2016.

[5] A.BALLIT et M.BADRAN. *Modélisation et controle dún quadrirotor.* Université Libanaise-Faculté de génie-Al Hadath-Département Electrique, 2012.

[6] F.MOHAMEDI et N.SACI. *Simulation dún drone sous MATLAB.* En vue de lâobtention du diplôme master professionnel informatique. Université de Béjaia, faculté des Sciences Exactes, 2016.

[7] I. MESLOULI et R.M MESLI. *Réalisation et pilotage dún drone à quatre rotors.* Mémoire présenté pour l'obtention du diplôme de Master en Automatique, Univesité Abou-Bakr, Tlemcen, 2017.

[8] F.MORBIDI. *Initiation a la robotique. Institution :Laboratoire MIS. Equipe Perception et Robotique. Université de Picardie Jules Verne.* ME 1.1, licence professionnelle automatisme et robotique. Laboratoire MIS, équipe perception et robotique. Université de Picardie Jules Verne, 2007.

[9] Forums.futura-sciences. -calcul-inertie-rotor-dun-moteur, 2006.

[10] M.TADJINE H.BOUADI, M.BOUCHOUCHA. *Sliding Mode Control based on Backstepping Approach for an UAV Type-Quadrotor.* International Journal of Mechanical and Mechatronics Engineering, 1(2). World Academy of Science, Engineering and Technology, 2007.

[11] H.KHEBBACHE. *Tolérance aux défauts via la méthode backstepping des systemes non linéaires Application : Systeme UAV de type Quadrirotor.* Mémoire présenté pour lâObtention du Diplôme de Magister, département dâelectrotechnique, faculté de Technologie, université FERHAT ABBAS de Setif, 06/06/2012.

[12] Mecaflux. Incidence angle d'attaque d'un profil dáile : la portance et trainée, 2019.

[13] S.BOUABDALLAH. *Design and control of quadrotors with application to autonomous flying. Mémoire présenté pour l'obtention du grade docteur en sciences, Ecole polytechnique fédérale LAUSANNE.* Thèse présentée pour l'obtention du grade de docteur. Faculté des sciences et techniques de l'ingénieur, école polytechnique fédérale de Lausanne, 2007.

[14] Wikipédia. https ://en.wikipedia.org/wiki/quadcopterapplications, 13 avril 2019.

[15] Wikipédia. https ://fr.wikipedia.org/wiki/dronehistoire, 26 avril 2019.

[16] Wikipédia. https ://fr.wikipedia.org/wiki/quadrirotorhistoire, 8 féevrier 2019.

[17] A.M. DARSONO et H.H. YUSOF Z.MUSTAPA, S.SAAT. *EXPERIMENTAL VALIDATION OF AN ALTITUDE CONTROL FOR QUADCOPTER.* ARPN journal of engineering and applied sciences 11(6). Faculty of Electronic and Computer Engineering, Universiti Teknikal Malaysia Melaka, 2016.

[18] Z.ZOU. *Trajectory tracking control design with command-filtered compensation for quadrotor.* IEEE Xplore : IET Control Theory and Applications. Beihang University (BUAA), 2010.

Conception et commande d'un quadrotor UAV à base d'Arduino

Résumé

Ce projet de Master est porté sur la modélisation, la simulation et la mise en ?uvre pratique d'un Drone quadrirotor.

D'abord, un modèle mathématique représentant la dynamique du drone a été développé. Ce modèle servira à la synthèse des lois de commande et de régulation dans le but d'assurer un asservissement total du drone tout en préservant le maximum de stabilité.

Afin d'illustrer les résultats obtenus, différentes méthodes d'identification ont été utilisées permettant de déterminer les valeurs des paramètres pour validation en simulation.

Une mise en oeuvre pratique du quadrotor est abordée par la suite, expliquant les stratégies de choix des composants élémentaires d'un quadrotor pour une optimisation des performances du drone ainsi que pour sa mobilité.

L'application de la commande est transmise au drone à travers la carte Arduino qui représente la partie électronique intelligente. Des vols en indoor et outdoor ont été effectué pour valider l'efficacité de la commande à base de PID.

Des trajectoires de vol ont été tracées grâce à la transmission des données de vol via XBee.

Mots clés

Quadrirotor, modèle mathématique, Arduino, identification,stabilité, XBee.

Design and control of a quadrotor UAV based on Arduino

Abstract

This master project focuses on the modeling, simulation and practical implementation of a quadrotor drone.

First, a mathematical model representing the dynamics of the drone has been developed. This model will is used for the regulation laws synthesis in order to ensure total control of the drone while preserving the maximum stability.

In order to illustrate the obtained results, different identification methods were used to determine the values of the parameters used in simulation.

A practical implementation of the quadrotor is discussed later, explaining the choice strategies of the basic quadrotor components for an optimization of the drone performances as well as for its mobility.

The implementation of the control to the drone is made through the Arduino board which represents the intelligent electronic part. Indoor and outdoor flights were conducted to validate the effectiveness of the PID-based control.

Flight trajectories have been ploted using the transmission of flight data via XBee.

Keywords

Quadrotor, mathematical model, Arduino, identification, stability, XBee.