



République Tunisienne
Ministère de l'Enseignement Supérieur,
de la Recherche Scientifique
Université de Carthage
Ecole Nationale d'Ingénieurs de Carthage

Projet de fin d'année

Spécialité : mécatronique

Drone Autonome

Réalisé par : BEN SAID Alaeddine

Encadrant : M. MAGHRAOUI Mohamed Nabil

Année universitaire : 2019/2020

Remerciements

A ma mère Raja ,
A mon père Seifeddine ,
A tous ceux et celles qui ont contribué de près ou de loin, par leurs conseils, leurs suggestions et par leurs encouragements, à la réalisation de ce travail.

Table des matières

Introduction générale.....	8
Chapitre I : Etude des applications des drones civile	9
INTRODUCTION	9
I/Application des drones civiles.....	9
II / Problématique et solutions.....	10
III/ Conclusion.....	10
Chapitre 2 : Etude et conception d'une drone autonome.....	11
INTRODUCTION.....	11
A/Etude mécanique.....	11
1/Le poids	11
2/ La Portance	12
3/La Trainée.....	13
4/La poussée	13
Les mouvements possible	13
B/Etude électrique :.....	15
I-Partie Radiocommande :.....	15
1/Le Transmetteur	15
2/Le récepteur	16
II/Partie Puissance :.....	17
1/ Moteurs sans balais (brushless motor).....	17
2/Electronic Speed Controller (variateur de vitesse électronique).....	18
3/ Batteries (LiPo batterie).....	18
4/ Carte d'alimentation.....	19
III/Partie commande :.....	19
1/Le microcontrôleur (Arduino Uno).....	19
2/MPU 9250.....	19
3/Le baromètre (BMP280).....	20
4/Module GPS (NEO-6 u-blox 6)	21
C/Fonctionnement d'une quadraopter (mode normale).....	22
Système d'asservissement pour un quadraopter.....	22
D/Fonctionnement d'une quadraopter (mode autonome).....	23
Chapitre 3 : Réalisation d'une drone.....	25
I/Réalisation de la partie Hard.....	25
1/Radiocommande.....	25
2/Châssis du drone	26
3/Carte de distribution d'alimentation	30
4/Connection de Batteries	30
5/Moteur sans balais(Brushless motor).....	31
6/Variateur de vitesse du moteur (ESC).....	31

7/MPU 9025 (Gyroscope + accéléromètre + magnétomètre).....	31
8/Equilibrage des Hélices.....	31
II/Réalisation de la partie Soft.....	35
1/ La partie SETUP	35
2/ Le code principale (void loop).....	35
3/ Routine d'interruption	36
III/ Mise en œuvre du projet.....	36
Conclusion	36
 Conclusion générale	 37
 Bibliographies.....	 39
 Netographie.....	 40
 ANNEXE A (code du transmetteur + code de récepteur).....	 41
 ANNEXE B Code du contrôleur de vol (pour le mode normale).....	 47
 ANNEXE C (carte inclinaison du nord de champ magnétique terrestre par rapport au nord géographique).....	 54

Liste des figures :

Fig2-1 : sens de rotation des moteurs d'une quadrocoptère.....	11
Fig2-2 : composants de la force aérodynamique.....	12
Fig2-3 : Les axes de rotation d'une drone.....	14
Fig 2-4 : mouvements possibles d'une quadrocoptère.....	14
Fig2-5 :Schéma synoptique	15
Fig2-6 :Schéma électrique du transmetteur.....	16
Fig2-7 :exemple d'un signal PPM.....	17
Fig2-8 :schéma électrique du récepteur.....	17
Fig2-9 :Système d'asservissement.....	22
Fig2-10 : équations PID.....	23
Fig2-11 :Angle de relèvement entre A et B.....	24
Fig 3-1 : conception d'un transmetteur basé sur l'Atmge8 + NRF24L01.....	25
Fig 3-2 : conception d'un récepteur basé sur l'Atmge8 + NRF24L01.....	26
Fig3-3 : Le design du drone est dessiné sur le contreplaqué avant le couper.....	27
Fig 3-4 : partie supérieure du châssis.....	27
Fig3-5 : partie inférieur du châssis (fixation du moteur et des ESC).....	28
Fig3-6 : châssis finale.....	29
Fig 3-7 : Carte de distribution d'alimentation.....	30
Fig3-8 :Câblage des batterie en parallèle.....	31
Fig3-9 :Moteur du quadrocoptère(A2212/6T 2200KV).....	31
Fig3-10 :ESC 30A BLDC ESC.....	32
Fig3-11 : MPU9025.....	32
Fig 3-12 : une hélice déséquilibrer.....	33
Fig3-13 : ajoutant du scotch au pale droite de faible masse.....	33
Fig3-14 : Résultat après l'équilibrage.....	34
Fig3-15 : conception finale du drone.....	35

Introduction générale :

Pourquoi une drone autonome ?

Cet idée qui tourne dans ma tête d'après longtemps consiste à créer un réseau des drones autonome qui peut couvrir une pays et offre des plusieurs services comme la livraison , intervention d'urgence et d'autre .

Notre siècle est marquée par l'autonomie on voit partout des robots autonomes , des voitures ...et maintenant c'est le tour pour les drones .

les tests et les prototypes de ce type de drone sont dans la phase initiale et même les grande entreprises de livraison investissent dans ce domaine .

L'application de ce type de drone doit satisfait tous les normes de sécurité et doit contenir des plusieurs capteurs et des algorithmes avancée pour bien réaliser sa mission en toute sécurité

Et tous ça on va le discuter dans les chapitres suivants :

Chapitre I : étude globale des applications des drones civile

Chapitre II : étude et conception d'une drone autonome

Chapitre III : réalisation d'un prototype d'une drone autonome

Chapitre I : étude globale des applications des drones civile

INTRODUCTION

Le marché des drones va connaître un véritable âge d'or, notamment sur le segment professionnel. Le besoin d'investisseurs est réel, particulièrement lors des premiers tours de table, si l'on veut voir émerger de futurs géants du secteur.

I/Application des drones civiles

Sauvés par un drone : à la fin du mois de juin 2019 , plus d'une centaine d'enfants vivant dans une région reculée du Ghana, en Afrique, ont été secourus par l'arrivée d'un drone de la société Zipline. Atteints de diarrhées aiguës, les écoliers, qui ne bénéficiaient ni d'un centre de santé local ni de moyens de transport adéquats, ont tous reçu une dose de sels de réhydratation, un traitement convoyé en un temps record par l'appareil volant. La preuve, s'il en fallait encore, que le meilleur de la technologie, en s'affranchissant des contraintes logistiques ou énergétiques, peut rendre des services très concrets, voire sauver des vies.

Les applications de drones proviennent d'une grande variété d'industries qui déploient déjà des drones à leurs fins respectives. En fait, on a trouvé des applications de drones dans différent secteur d'industrie :

- Agriculture
- Arts, spectacles et loisirs
- Construction
- Services éducatifs
- Énergie
- Santé et services sociaux
- Information
- Assurance
- Extraction minière, exploitation en carrière, et extraction de pétrole et de gaz
- Administration publique
- Services professionnels, scientifiques et techniques
- Immobilier, location et crédit-bail et usines industrielles
- Réparation et entretien
- Sûreté et sécurité
- Transport et entreposage

II / problématique et solutions

Avec des applications aussi diverses que l'agriculture, l'énergie, la construction, la santé, les assurances, le cinéma et, bien entendu, la sécurité, ne risque-t-on pas cependant de poser les conditions d'une véritable « guerre du ciel » au-dessus de nos villes et de nos campagnes ? Ici encore, les progrès technologiques nous permettent d'envisager l'avenir avec sérénité : de plus en plus systématiquement équipés de dispositifs de geofencing et anticollision, les drones nouvelle génération ont tout pour rassurer les autorités et régulateurs du ciel. L'OACI (Organisation de l'Aviation Civile Internationale) se penche depuis 2008 sur les cas des drones, et devraient accoucher, bientôt, d'une réglementation au moins aussi sécurisée que celle de l'aviation.

III/ conclusion

L'utilisation des drones devient de plus en plus très importante , on peut résumer l'importance des drones dans ces 5 points :

- SECURITÉ
- GAIN DE TEMPS
- NOUVELLES PERSPECTIVES
- ÉCOLOGIQUE
- INNOVANT

Chapitre 2 Etude et conception d'une drone autonome

INTRODUCTION

Comment une Drone vole ?

L'hélice est un dispositif formé de plusieurs pales disposées régulièrement autour d'un axe du moteur . Lorsque son axe entre en rotation , ce système grâce aux pales orientées suivant un certain angle d'attaque , prend appui sur le fluide, ce qui forme la force de portance (la drone peut voler facilement a condition que la valeur de la force de portance et deux fois plus grand que la valeur de la force de gravité) .

Pour une drone de type quadraopter il faut que les deux moteurs opposés et leur hélices tournent en sens horaire et les deux autres tournent en sens anti-horaires

La drone nécessite un ensemble des cateurs pour maintenir sa stabilité et suivre la consigne de pilote.

A/Etude mécanique

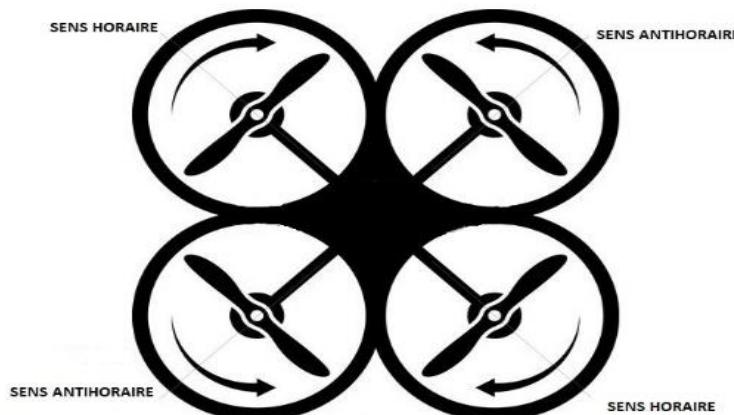


Fig2-1 : sens de rotation des moteurs d'une quadraoptére

Les forces appliquées :

En étudiant le système on trouve qu'il est soumis à 4 forces :

- Le poids
- La portance
- La poussée
- La Trainée

1/Le poids :

Le poids est la force de pesanteur exercée par la Terre sur le système ,appliquée au centre de gravité et d'intensité :

$$P = m \cdot g \quad (2.1)$$

La force aérodynamique (portance + trainée) :

La rotation des quatre moteurs fournit une résultante aérodynamique . cette force résulte de l'écoulement de l'air sur les surfaces de pales .C'est une force inclinée dirigée du bas vers le haut , appliquée au centre de gravité. Cette force est généralement décomposée en deux forces : **la portance et la trainée** .

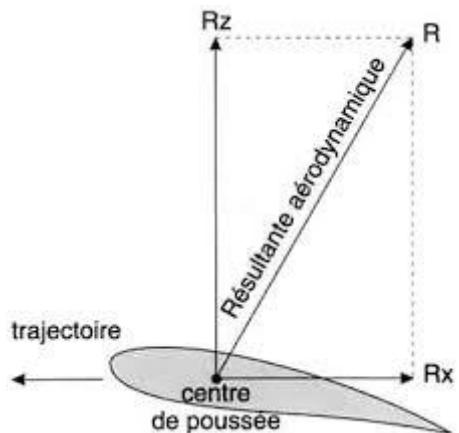


Fig2-2 : composants de la force aérodynamique

2/ La Portance :

La portance est une force verticale qui ,lorsqu'elle compense au moins le poids, permet au drone de s'élever , sa équation est linéaire et elle est sous la forme suivante :

$$F_z = \frac{1}{2} * \rho * S * C_z * V^2 \quad (2.2)$$

V : vitesse entre l'hélice et l'aire en m/s $V = \Omega * r$ (2.3)

Ω : fréquence de rotation en rad/s.

r : la distance entre le centre et le point de force sur l'hélice en m

ρ : masse volumique d'air en kg/m³ 1.3

S : surface à l'aire de l'aile en m² $S = L * R$

R : rayon de l'hélice

C_z : coefficient de portance sans unité

Portance totale du quadrorcopter :

D'après la formule de vitesse de l'hélice (2.3) , la vitesse varie le long de l'hélice c-à-d la vitesse sur l'axe de rotation est nulle , mais elle est maximum à l'extrémité .

Alors pour obtenir la portance totale de l'hélice il suffit d'intégrer les forces infinitésimales sur des surfaces infiniment petites le long du rayon de l'hélice .

Donc : $F_{zth} = 1/6 * \rho * S * C_z * \Omega^2 * R^3 \quad (2.4)$

Sachant qu'on a 4 hélices et chaque hélice contient 2 pales donc la force portance totale du quadrorcopter : $F_{zt} = 4/3 * \rho * S * C_z * \Omega^2 * R^3 \quad (2.5)$

3/La Trainée :

La trainée est la force qui s'oppose au mouvement d'hélice. C'est la résultante des frottements de l'air sur l'hélice , elle est opposée à la trajectoire .

C'est la composante latérale de la force produite par le mouvement de l'hélice.

L'équation de la trainée est linéaire sous la forme suivant :

$$F_x = \frac{1}{2} * \rho * S * C_x * V^2 \quad (2.6)$$

V : vitesse entre l'hélice et l'aire en m/s

ρ : masse volumique d'air en kg/m³

S : la surface de référence projetée de l'aile suivant OX . surface a l'aire de l'hélice en m²

S = E * R (avec E hauteur ou largeur sur le plan OX OZ)

Cx : coefficient de trainée sans unité

Alors pour obtenir la trainée totale de l'hélice il suffit d'intégrer les forces infinitésimales sur des surfaces infiniment petites le long du rayon de l'hélice (même méthode avec la portance)

$$F_{xth} = \frac{1}{6} * \rho * S * C_x * \Omega^2 * R^3 \quad (1.7)$$

Sachant qu'on a 4 hélices et chaque hélice contient 2 pales donc la force portance totale du quadraopter : $F_{xt} = \frac{4}{3} * \rho * S * C_x * \Omega^2 * R^3 \quad (1.8)$

4/La poussée :

La poussée est la force qui conduit le quadraopter pour se déplacer dans la direction voulue . la poussée ou plus exactement la force de poussée est le résultat d'une petite augmentation de rotation au niveau de l'un de 4 moteurs ou même de deux moteurs .cette augmentation nous donne un angle d'inclinaison qui se produit lorsqu'on veut augmenter la vitesse de déplacement latéralement tout en maintenant l'avion en vol

Comment augmenter l'efficacité du quadraopter selon les formules obtenues des ces forces ?

1/ selon la formule de gravité (2.1) on doit diminuer le maximum de poids du système

2/ selon la formule de portance (2.2) pour l'augmenter on peut jouer sur les facteurs suivant : Augmentation du taille d'hélice , augmentation du vitesse de rotation , augmentation le nombre de pale d'une hélice.

Les mouvements possibles :

1/ les gaz (Throttle) : elle correspond tout simplement à la montée / descente (soit en augmentant ou réduction de la vitesse des 4 moteurs)

2/ Le Lacet (Yaw) : cet mouvement sert à faire tourner le quadraopter sur lui-même . il est obtenu soit en augmentant la vitesse des hélices du sens horaire et en diminuant proportionnellement la vitesse des hélices du sens antihoraire, ou inversement

3&4/ Le roulis et le tangage (Roll and Pitch) : ces mouvements sont assez similaires visant à pencher la drone sur un axe ou sur un autre . ce mouvement est obtenu en augmentant la vitesse de deux hélice de même coté et en abaissant proportionnellement la vitesse des deux hélice opposée.

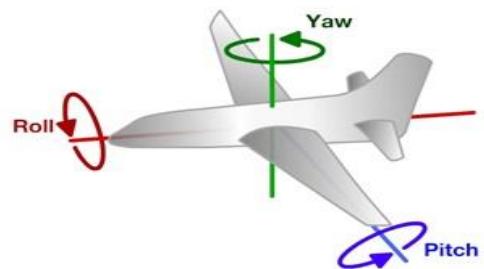


fig2-3 : Les axes de rotation d'une drone

<u>Throttle control</u>		<u>Pitch control</u>	
Move down		Move forward	Move backward
<u>Roll control</u>		<u>Yaw control</u>	
Bend left	Bend Right	Rotate left	Rotate right

- Normal Speed
- High Speed

Fig 2-4 : mouvements possibles d'une drone

B/ Etude électrique :

Schéma synoptique du drone autonome

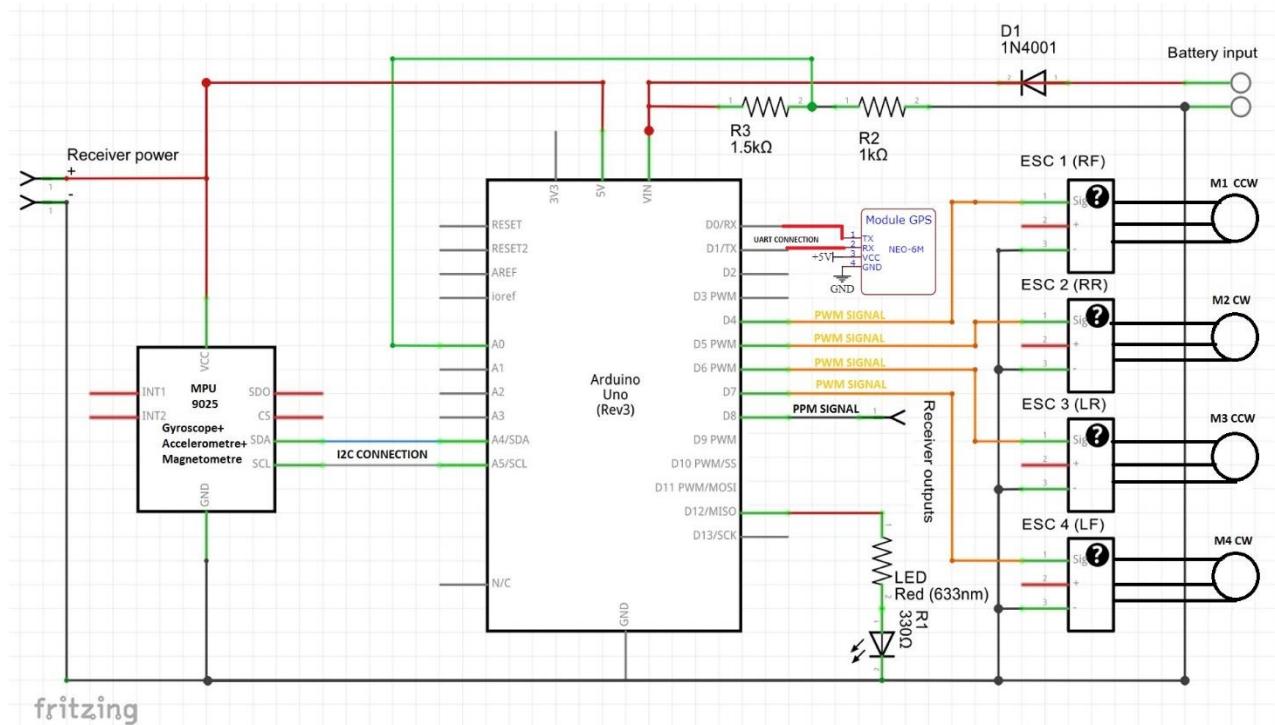


Fig2-5 Schéma synoptique

I-Partie Radiocommande :

Une radiocommande d'aéromodélisme est un instrument permettant de commander un système aérodynamique à distance.

Une radiocommande se compose de deux éléments :

1/Le Transmetteur :

Notre transmetteur doit être capable d'envoyer 6 chaines au récepteur pour une longue distance et en haut vitesse pour bien contrôler la stabilité du drone ,les mouvements possibles et d'autre fonction pour la drone.

notre transmetteur contient 4 potentiomètres avec 2 interrupteurs , un microcontrôleur et un module radio de 2.4GHz

*Microcontrôleur :

on choisit le Atmega8 ; c'est un microcontrôleur de 8 bits il contient 6 pin analogique et les restes sont numérique , une seule interface qui supporte le protocole de communication SPI,I2C,USART ,une horloge de 16Mhz,8KO de mémoire de programme,1KOctet de SRAM,512 Octet EEPROM et il est alimenté avec 5v

*Module radio :

On choisit le NRF24L01 avec une antenne qui a une bande de 2.4GHz ,sa vitesse est entre 250Kbps jusqu'à 2Mbps il contient 4 pin pour la protocole de communication SPI et il est alimenté avec une tension de 3.3v

***4 potentiomètres** pour simuler les mouvements (Throttle, Yaw , Roll , Pitch)

***2 boutons interrupteurs** pour simuler d'autre fonctions

Algorithmes du transmetteur :

1/Setup du module radio (en choisissant la baud correcte , définir le mode de connexion entre le module radio et le microcontrôleur , définir le clé spéciale entre le transmetteur et le récepteur)

2/ lire les valeurs des différent potentiomètres et interrupteurs et les mettre entre 0 et 255

3/envoyer les données en utilisant les registres du module radio

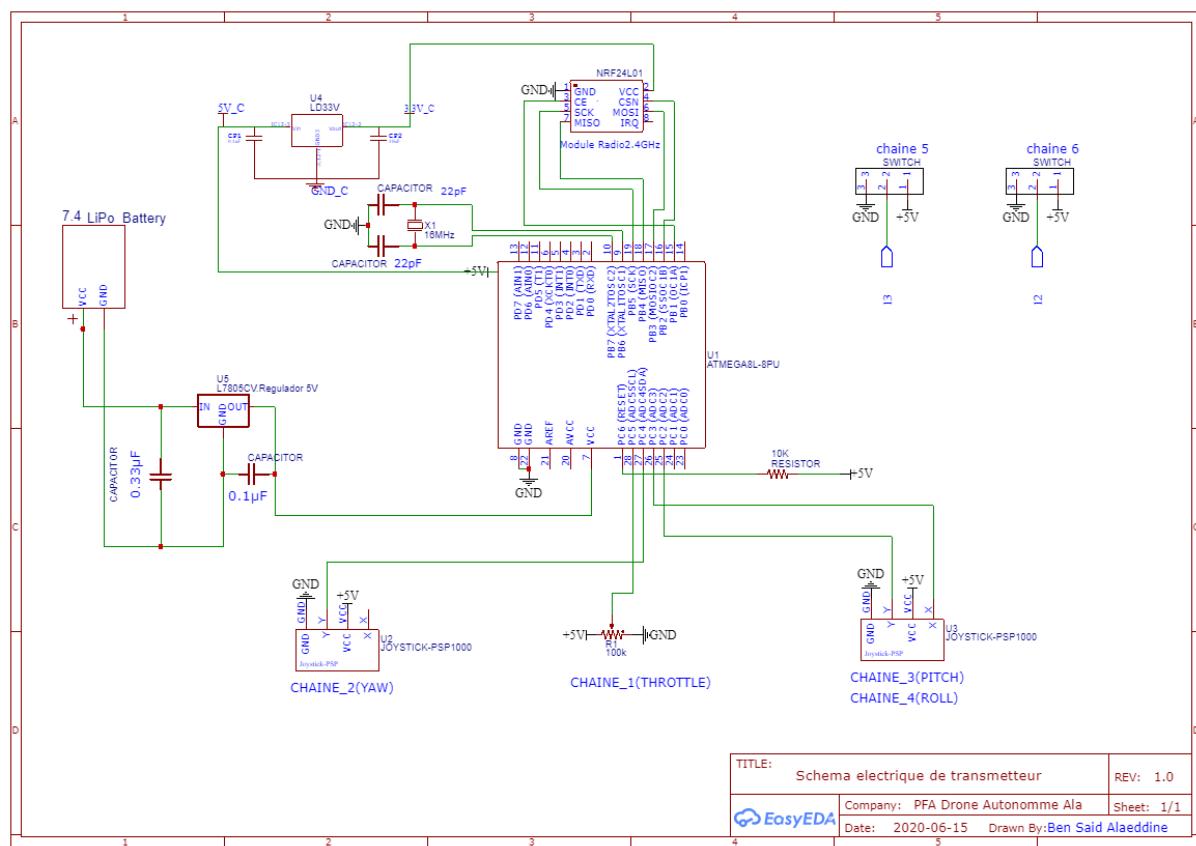


Fig2-6 Schéma électrique du transmetteur

2/Le récepteur :

Il contient presque les mêmes composant que le transmetteur (Atmega8 et le module radio nrf24L01)

Le rôle du récepteur et d'envoyer les données envoyé par le transmetteur au contrôleur de vol en utilisant une interruption de type TIMER le récepteur va générer un signal de type PPM sortant sur un seul pin du récepteur vers le contrôleur de vol

Ce signal PPM contient les valeurs de toutes les chaînes du radio commande comme indique la photo suivante

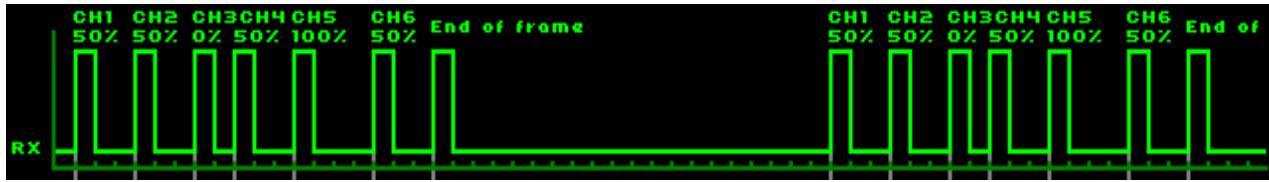


Fig2-7 exemple d'un signal PPM

Algorithmes du récepteur :

1/Setup du module radio (en choisissant la baud correcte , définir le mode de connexion entre le module radio et le microcontrôleur , définir le clé spéciale entre le transmetteur et le récepteur)

2/recevoir de données

3/en utilisant l'interruption TIMER on va générer un signal PPM qui contient tous les valeurs des 6 chaines l'une après l'autre

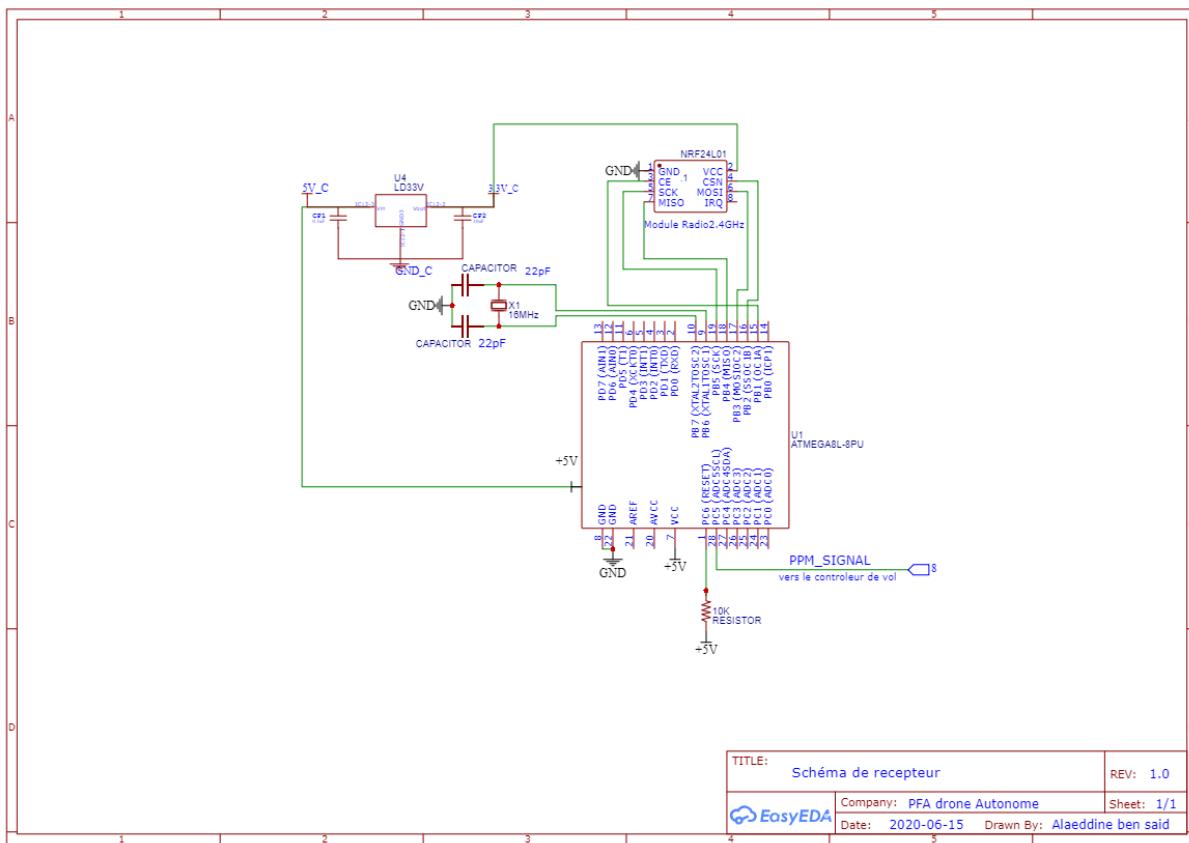


Fig2-8 schéma électrique du récepteur

II/Partie Puissance :

1/ moteurs sans balais (brushless motor)

Les moteurs sans balais sont les moteurs les plus populaires pour les drones.

Le moteur sans balais ou bien le moteur brushless est une machine électrique de la catégories du machines synchrones dont le rotor constitué d'aimants permanent et le stator constitué des bobine,

Il se base simplement sur l'interaction créée entre les aimants et le champ électromagnétique créé . Vu de l'extérieur, il fonctionne en courant continu. Par contre un système électronique de commande doit assurer la commutation du courant dans les enroulements statoriques.

L'utilisation de ce type de moteur est justifié par sa fiabilité ,sa longue durée d'usage ,moins de bruit et de maintenance ,son prix bas et surtout le contrôle de son vitesse

- Le nombre KV d'un moteur brushless

Le nombre KV indiqué sur le moteur traduit la rotation par minute selon la formule suivante :

$$\text{Rotation Par Minute} = \text{KV} * \text{Tension en entrée}$$

Dans ce projet on choisit un moteur de 2200 KV avec une batterie de 12 volts

Donc le RPM max peut fournir ce moteur est $2200 * 12 = 26400$ rotation/min

Caractéristique du moteur choisi lors de ce projet

- Référence A2212/13T
- KV 2200
- 2-3 cellules LiPO
- Courant max 13 A pour 60 secondes
- Dimension 2.8*2.8*3.2 mm
- Propulsion générer 300gr-800gr

2/Electronic Speed Controller (variateur de vitesse électronique)

Un variateur de vitesse électronique ou ESC est un circuit électronique qui contrôle et régule la vitesse d'un moteur électrique .il est constituer d'un ensemble de transistor et un microcontrôleur pour commuter entre les 3 différents entrées du moteur d'une façon bien synchroniser et précis .

Le régulateur de vitesse électronique reçoit un signal PWM à partir du contrôleur de vol entre 1000 μ s et 2000 μ s (1000 μ s signifie que le moteur est en repos et 2000 μ s signifier que le moteur est en rotation maximale)

Le régulateur de vitesse électronique contient un capteur d'effet de Hall pour bien synchroniser la commutation entre les trois phase du moteur

Caractéristique du ESC choisi lors de ce projet :

- Référence 30A BLDC ESC
- Supporte une sortie de 30A continue , 40 A pour 10 secondes
- Dimensions 55mm x 26mm x 13mm
- Programmable en différent modes

3/ Batteries (LiPo batterie)

Une batterie Lipo est une batterie de type Lithium-polymère.

Ce type est le plus utilisé pour les drones , voitures RC et autres véhicules , elle offre un meilleur rapport puissance/poids que toutes les autres batteries.

La batterie se caractérise par :

- Nombre de cellule
- Capacité
- Taux de décharge (C-Rate)

Chaque cellule a un voltage nominale de 3.7 v (4.2 pour une cellule pleinement chargé).

Pour ce projet on fait la liaison parallèle entre deux batteries(3s) 11.1v pour augmenter l'intensité (c'est-à-dire la durée de vol)

Les deux batteries sont identiques de type Lipo ,de capacité 3000mAh et avec C-rate C=20 .

→Le taux de décharge maximale d'une seule batterie = capacité * C -rate

$$=3000*20 = 60.A \text{ (c'est le courant maximale peut être débiter par la batterie)}$$

→La capacité d'une batterie c'est le débit du courant qu'une batterie peut le générer d'une façon continue pendant une heure (3000mAh c-à-d intensité de 3A continue dans une heure jusqu'à la batterie est épuisée)

4/ Carte d'alimentation

Cette carte permet la distribution de l'alimentation pour les 4 moteurs brushless et leur ESCs Elle alimente aussi le contrôleur de vol, un diviseur de tension (deux résistances 1kΩ et 1.5 kΩ) pour mesurer la tension de la batterie et une diode de Schottky pour protéger la circuit de puissance lors du téléchargement du code

III/Partie commande :

contrôleur de vol

Le contrôleur de vol est le cerveau de notre système. Il gère de nombreux paramètres pour rendre possible son pilotage. Il est généralement accompagné de nombreux capteurs externes et leurs données sont traitées par le microcontrôleur

1/Le microcontrôleur (Arduino Uno)

Pour ce projet on choisit l'Arduino Uno basé sur l'Atmega328P qu'est un microcontrôleur de 8bits avec une horloge de 16MHz, il contient 6 chaines PWM ,32K Byte de mémoire de programme ,1KBytes EEPROM,2KBytes SRAM ,un seul interface pour la communication SPI et un seul interface pour la communication I2C.

On va utiliser l'Arduino IDE pour programmer ce microcontrôleur

Le microcontrôleur va recevoir les différents données des capteurs installés et puis il va assurer le calcul nécessaire pour maintenir la stabilité du drone et les différents modes de vol

Les capteurs

2/MPU 9250

C'est un capteur MEMS(systèmes micro électromécaniques) inertiel de 9 axes ,alimenté avec 5v .Il est constitué d'un gyroscope de 3 axes ,un accéléromètre de 3 axes et un magnétomètre de 3 axes ,le MPU9250 communique avec le microcontrôleur en utilisant le protocole de communication I2C

Le gyroscope :

C'est un gyroscope à structure vibrante , qui utilise une structure vibrante pour déterminer la vitesse de rotation

Le gyroscope permet la détermination de la vitesse de rotation suivant 3 axes (roll , pitch,yaw).

Le gyroscope consomme environ 3.2mA .

La conversion analogique numérique du signal est sur 16 bits .

Selon la datasheet la sortie du gyroscope est exprimée en degré par seconde.

On doit configurer quelque registre dans le MPU9250 afin d'activer le gyroscope et de sélectionner le Full-Scale Range (gamme complète) pour un valeur de +/-500 deg/seconde

L'accéléromètre :

Un accéléromètre est un capteur qui, fixé à un mobile ou tout autre objet, permet de mesurer l'accélération linéaire de ce dernier. On parle d'accéléromètre même lorsqu'il s'agit en fait de 3 accéléromètres qui calculent les accélérations linéaires selon 3 axes orthogonaux.

L'accéléromètre consomme environ 450µA.

La conversion analogique numérique du signal est sur 16 bits .

Les données d'accéléromètre sont exprimer en G (c'est la constante de gravitation environ 9,81 m/s²)

On doit configurer quelque registre dans le MPU9250 afin d'activer l'accéléromètre et de sélectionner le Full-Scale Range (gamme complète) pour un valeur de +/-4 g

On peut utiliser l'accéléromètre pour vérifier si les axes(Roll, Pitch , Yaw) si ils sont inversé ou non lors du vol et d'éliminer le bruit

Le magnétomètre :

Un magnétomètre est un appareil qui sert à mesurer selon les cas l'intensité ou la direction d'un champ magnétique (dans notre cas c'est le champ magnétique terrestre).

Le magnétomètre consomme environ 280µA.

La conversion analogique numérique du signal est sur 14 bits .

Il existe une seul full scale range +/- 4800 µT.

Le rôle principale d'un magnétomètre pour une drone et de se comporter comme une boussole donc on doit convertir le nord magnétique indiquée par le magnétomètre au nord géographie et cette marge de variation entre ces deux nord varie selon la position géographie comme indiquer sur la carte dans l'annexe (en Tunisie la différence entre le nord géographie et le nord magnétique est 3°)

3/Le baromètre (BMP280) :

Le baromètre BMP280 utilise la technologie MEMS, ce qui le rend capable de mesurer la pression atmosphérique dans une petite structure flexible.

Comment ça fonctionne ?

IL contient un diaphragme formé à travers une plaque capacitive en contact avec l'atmosphère, la pression atmosphérique est détectée à travers la déformation du diaphragme due à la pression résultante.

Plus la pression est élevée, plus le diaphragme bouge, ce qui se traduit par une lecture du baromètre plus élevée.

Le rôle de ce capteur est de mesurer l'altitude pour aider la drone a garder l'altitude ou bien d'autre fonctionnalités pour une drone autonome .

On connaît bien que l'altitude est proportionnel inversement à la pression mesuré

Le calcul d'altitude se fait simplement avec la formule du nivelllement barométrique

$$P = P_0 e^{\frac{-\mu g h}{RT}},$$

où

μ - Masse molaire de l'air terrestre, 0,0289644 kg/mol

g - Accélération gravitationnelle, 9,80665 m/(s*s)

h - Différence d'altitude, mètre

R - Constante universel des gaz pour l'air, 8,31432 N·m /(mol·K)

T - Température de l'air, K

P_0 correspond également à la pression atmosphérique standard 0 mètre au-dessus de la mer

Pour le BMP280 on utilise le protocole de communication SPI .

La précision de ce capteur est ± 0.12 hPa (équivalent à ± 1 m).

Le BMP280 consomme environ 2.7 μ A pour un taux d'échantillonnage de 1Hz

4/Module GPS (NEO-6 u-blox 6) :

Global Positioning System (Système mondial de positionnement) est un module essentiel pour la partie autonomie de la quadraopter , le module GPS nous donne plusieurs données comme la longitude, latitude, altitude, temps réelle, nombre de satellite connecté au module ,pour ce projet on va utiliser juste la longitude et latitude pour bien repérer la position du quadraopter

Comment ça fonctionne ?

Pour définir une position dans l'espace, il faut trois coordonnées (x, y et z). Les données GPS intègrent également une quatrième variable : le temps. Il faut donc quatre satellites munis d'horloges atomiques pour obtenir une position, ainsi qu'un récepteur GPS qui va décoder et calculer les signaux reçus.

Chaque satellite émet une onde électromagnétique de vitesse connue. Cette onde, porteuse d'un code « pseudo-aléatoire », est émise à un temps bien déterminé. Le récepteur calcule ensuite le temps de transmission, c'est à dire le temps nécessaire pour que son signal (qui est porteur du même code pseudo-aléatoire) soit en phase avec le signal émis par le satellite. En multipliant ce temps par la vitesse, il obtient donc la distance qui le sépare du satellite.

A l'issue de ce calcul, le récepteur dispose d'une première information : il se trouve sur un cercle centré sur le satellite. En répétant cette procédure avec un deuxième satellite, il peut à nouveau se situer sur un second cercle centré sur le deuxième satellite. En réitérant l'opération une troisième fois et en cherchant la zone d'intersection entre ces trois cercles, on obtient la position sur la Terre. Le quatrième satellite permet de déterminer le décalage entre l'heure du récepteur gps et l'heure exacte fournie par les satellites, pour affiner la position. Plus le nombre de satellites captés sera important, meilleure sera la précision.

Le taux de mise à jour (update rate) pour ce module GPS est 1Hz (chaque seconde).

Le protocole de communication utilisé pour ce module est UART .

C /Fonctionnement d'une quadraopter (mode normale) :

A l'aide d'une interruption sur un front montant le contrôleur de vol lire tous les valeur des chaines envoyés par le récepteur sous forme d'un signal PPM ,ces valeurs sont la consigne

ou bien " Set point " , le microcontrôleur convertit ensuite ces valeurs entre l'intervalle de $1000\mu s$ et $2000\mu s$ ($1000\mu s$ c-à-d moteur au repos et $2000\mu s$ c-à-d moteur à sa vitesse maximale) et puis il envoie pour chaque ESC la pulse bien déterminé sous forme d'un signal PWM pour gérer les mouvements désirés du pilote

On envoie pour tous les ESCs la même valeur de chaîne1 (Throttle) et puis pour réaliser les autres mouvements (Roll ,Pitch ,Yaw) on augmente les pulses selon la configuration des moteurs

Système d'asservissement pour un quadraopter

Le quadraopter ne peut pas maintenir sa stabilité facilement à cause des vents, centre de gravité n'est pas au centre du système , les 4 moteurs et leur ESC ne sont pas parfaitement identiques ,d'où le risque d'être écrasé .

Pour cela on utilise un gyroscope pour déterminer la vitesse angulaire suivant les 3 axes du système, la différence entre les valeurs du radiocommande (consigne) et la position du quadraopter mesuré par le gyroscope représente l'erreur de ce système

Donc on va implémenter 3 régulateur PID (un régulateur pour chaque axe) pour maintenir la stabilité et corriger les pulses envoyés vers les ESCs

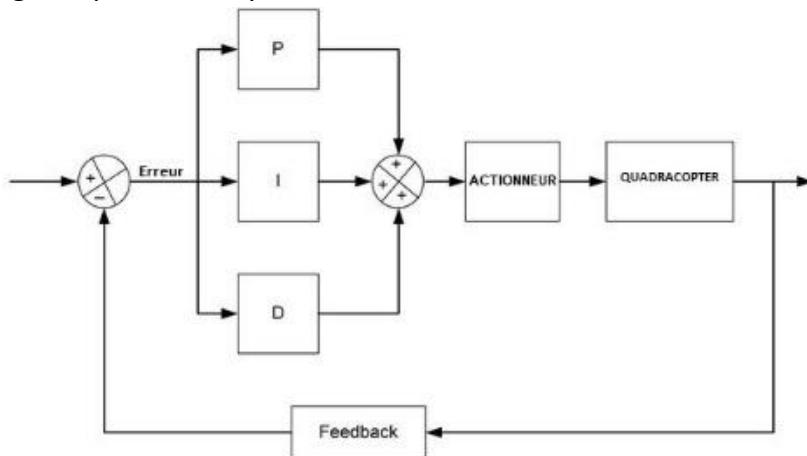


Fig2-9 Système d'asservissement

Le gain P détermine à quel point le contrôleur de vol travaille dur pour corriger l'erreur et atteindre la trajectoire de vol souhaitée (c'est-à-dire où le pilote veut que le drone se rende en déplaçant les manches de l'émetteur).

Le gain I détermine à quel point le contrôleur de vol travaille dur pour maintenir l'attitude du drone contre les forces externes , telles que le vent et le CG décentré.

Le gain D fonctionne comme un amortisseur et réduit la sur-correction et les dépassemens causés par le terme P. Comme un amortisseur empêche la suspension de rebondir, l'ajout d'un gain D peut diminuer les oscillations causées par un gain P excessif.

Les coefficients P,I et D dépendent de la conception mécanique du quadraopter le "Hardware" utilisé et d'autre terme , la détermination des ces valeurs nécessite plusieurs tests pour trouver les correspondantes pour une vol stable .

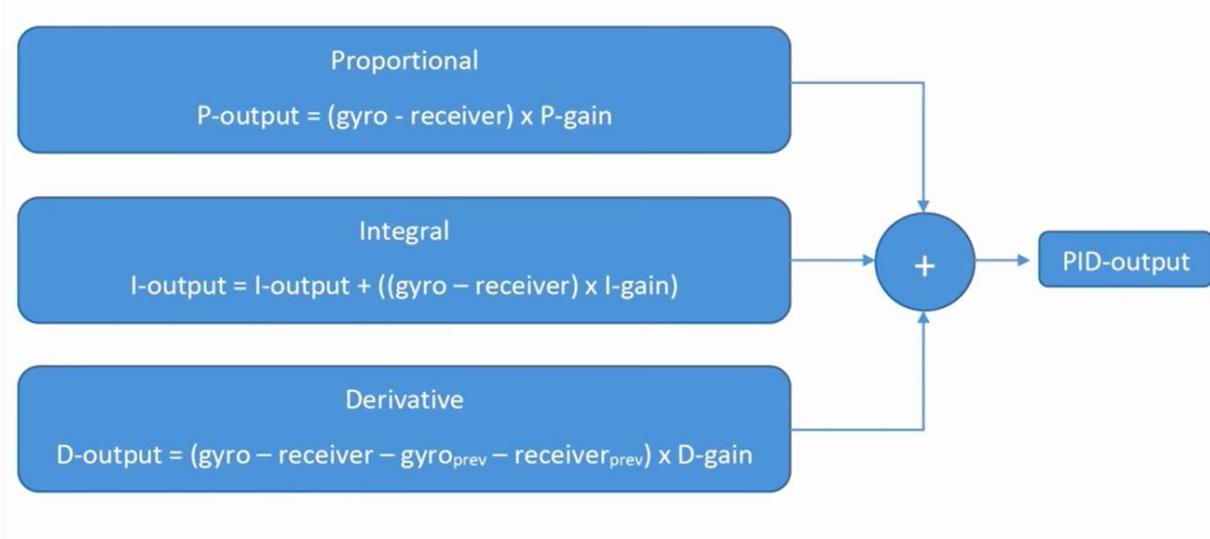


Fig2-10 équations PID

On envoie pour tous les ESCs la même valeur de chaine1 (Throttle) et puis pour réaliser les autres mouvements (Roll ,Pitch ,Yaw) on augmente les pulses pour les ESCs correspond (comme indiqué dans les équation suivantes)

Pulse pour ESC 1 (face-droit - CCW)

$$\text{esc_1} = \text{throttle} - \text{pid_output_pitch} + \text{pid_output_roll} - \text{pid_output_yaw}$$

Pulse pour ESC 2 (arrière-droit - CW)

$$\text{esc_2} = \text{throttle} + \text{pid_output_pitch} + \text{pid_output_roll} + \text{pid_output_yaw}$$

Pulse pour ESC 3 (arrière-gauche – CCW)

$$\text{esc_3} = \text{throttle} + \text{pid_output_pitch} - \text{pid_output_roll} - \text{pid_output_yaw}$$

Pulse pour ESC 4 (face-gauche - CW)

$$\text{esc_4} = \text{throttle} - \text{pid_output_pitch} - \text{pid_output_roll} + \text{pid_output_yaw}$$

D/Fonctionnement d'une quadraopter (mode autonome) :

Pour que la drone peut se voler complètement autonome elle doit connaitre sa destination , sa position actuel , sa direction et l'altitude

Il existe des plusieurs fonctions autonome (garder la position , garder l'altitude , retourne de sécurité , mission autonome de point A à point B)

Avant tout , la radiocommande doit contenir un bouton interrupteur qui permet le passage entre le mode normale et le mode autonome ou l'inverse pour assurer la sécurité

1/Sélectionner de l'altitude pour voler

La drone autonome doit voler sur une altitude bien choisi affin d'éviter les collisions avec les bâtiments ou bien d'autre obstacle selon la géographie du lieu

Donc la première étape s'agit de l'élévation du drone jusqu'à que le baromètre indique l'acquisition de l'altitude nécessaire (ainsi pour la procédure de décollage)

2/Orientation vers la destination

On doit choisir le plus court chemin pour atteindre le point B (c à d la plus courte droite qui relie le point A et B)

Pour le choix de cette direction on doit calculer l'angle de relèvement entre ces deux points.

Le relèvement : est la détermination de l'angle que fait, dans le plan horizontal, la ligne d'un observateur vers un objet avec celle d'une direction de référence fixe. En général, le relèvement se fait par rapport au Nord géographique : positif dans le sens horaire.

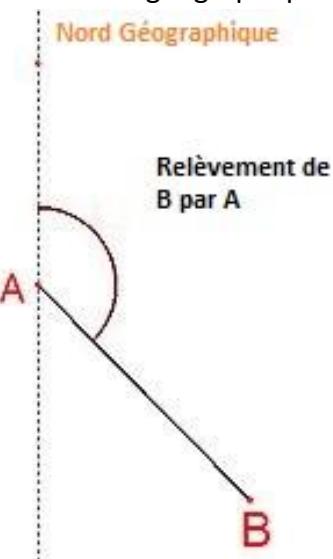


Fig2-11 Angle de relèvement entre A et B

L'angle de relèvement peut se calculer à l'aide des logiciels, On stocke cet angle dans la mémoire de contrôleur de vol, à l'aide du magnétomètre la drone s'orienter afin de maintenir cet orientation par rapport au nord magnétique

3/ Trajectoire à suivre

On fait stocker aussi dans la mémoire du contrôleur de vol les coordonnées(Longitude , Latitude) de la point B (point destination) .

Lors du vol la drone traite les données du module GPS qui indique sa position actuelle , On fait la comparaison entre la destination et la position actuel , La drone prend la décision d'avancer ou non

Chapitre 3

Réalisation d'une drone

Introduction

Dans ce chapitre nous avons présenté les étapes de construction et la réalisation du quadrocoptère .L'objectif de ce projet est de construire une première version d'une drone autonome avec le cout le plus bas pour tester le code et son efficacité

I/Réalisation de la partie Hard

1/Radiocommande

Transmetteur

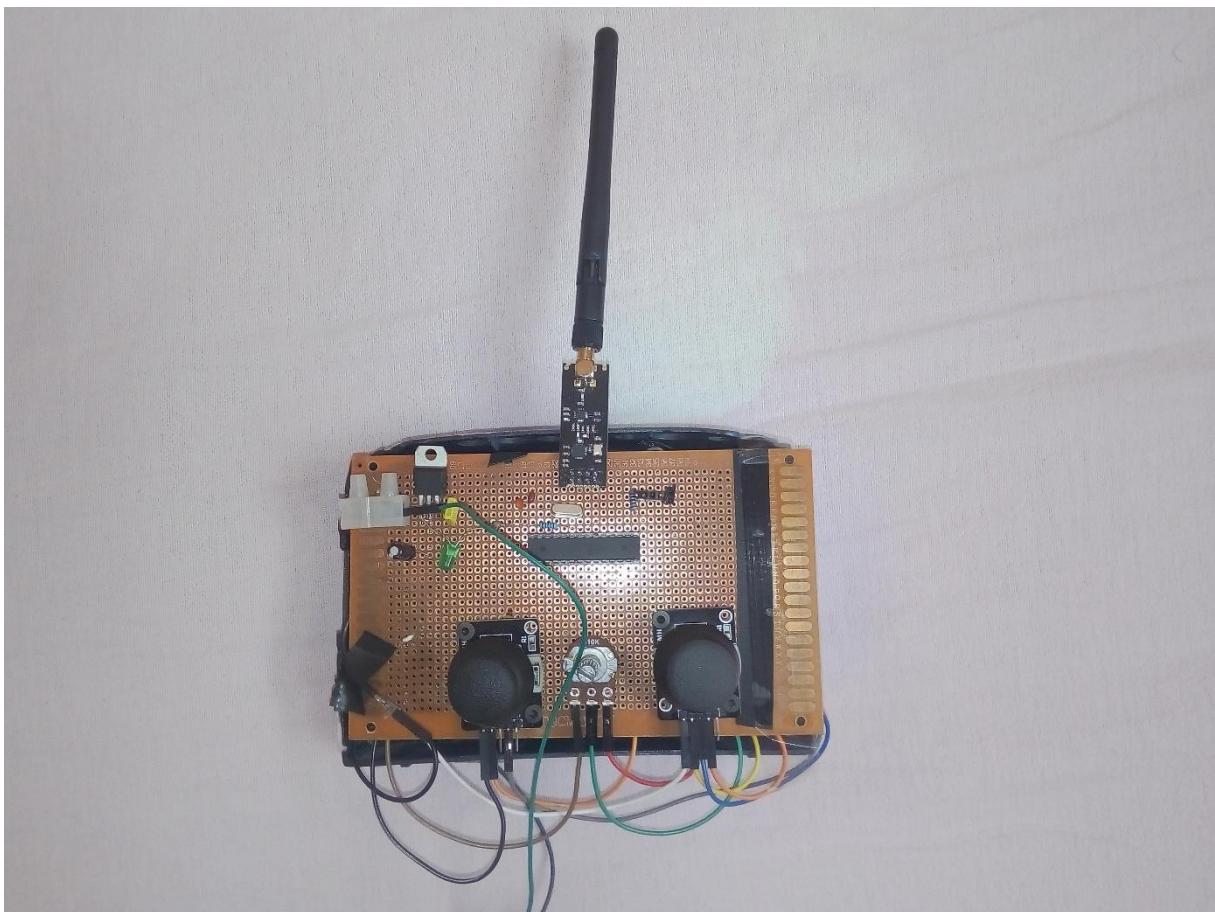


Fig 3-1 : conception d'un transmetteur basé sur l'Atmge8 + NRF24L01

Récepteur

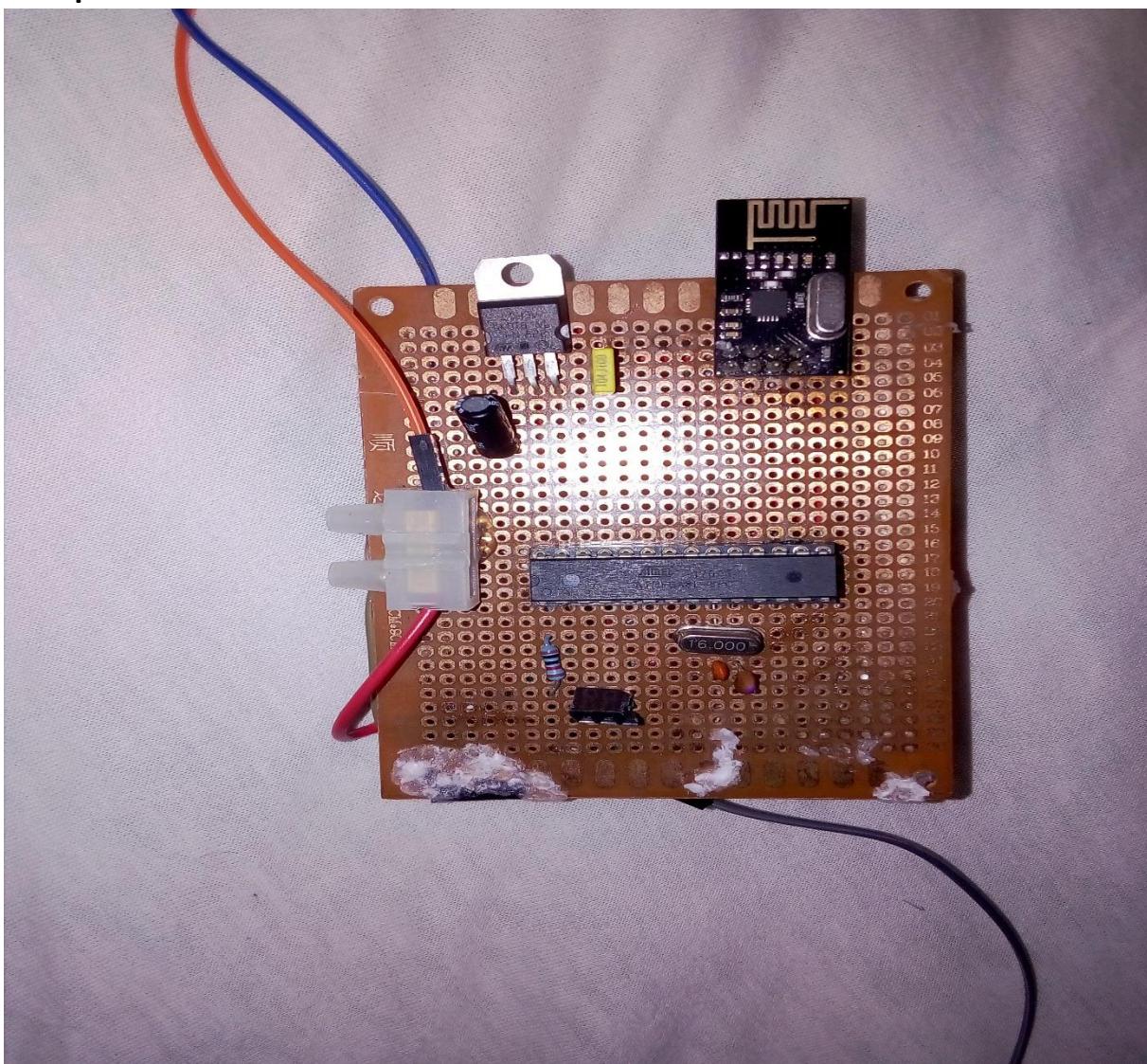


Fig 3-2 : conception d'un récepteur basé sur l'Atmge8 + NRF24L01

2/Châssis du drone :

On choisit un contreplaqué comme un matériau puisqu'il est moins cher et plus résistant que d'autres matériaux comme le plastique



Fig3-3 Le design du drone est dessiné sur le contreplaqué avant le couper



Fig 3-4 partie supérieure du châssis



Fig3-5 partie inférieur du châssis (fixation du moteur et des variateurs de vitesse ESC)

On insère entre la partie supérieure et inférieure un matériau léger comme la ponce puis on relie les deux parties avec un scotch (ruban adhésif).
cette structure est une protection pour que l'hélice ne représente aucune danger pour le pilote et aussi de la protéger pendant l'écrasement
Cette forme qui semble à un Tuyère Kort permet de générer une force de portance plus grande que les autres hélices sans un Tuyère Kort



Fig3-6 châssis finale

3/Carte de distribution d'alimentation :

Cette carte permet la distribution d'alimentation pour les variateurs de vitesse ESCs et leur moteur brushless , ainsi que le contrôleur de vol ,elle contient aussi un diviseur de tension pour déterminer le niveau de batterie et un diode de Schottky pour la protection de la carte Arduino

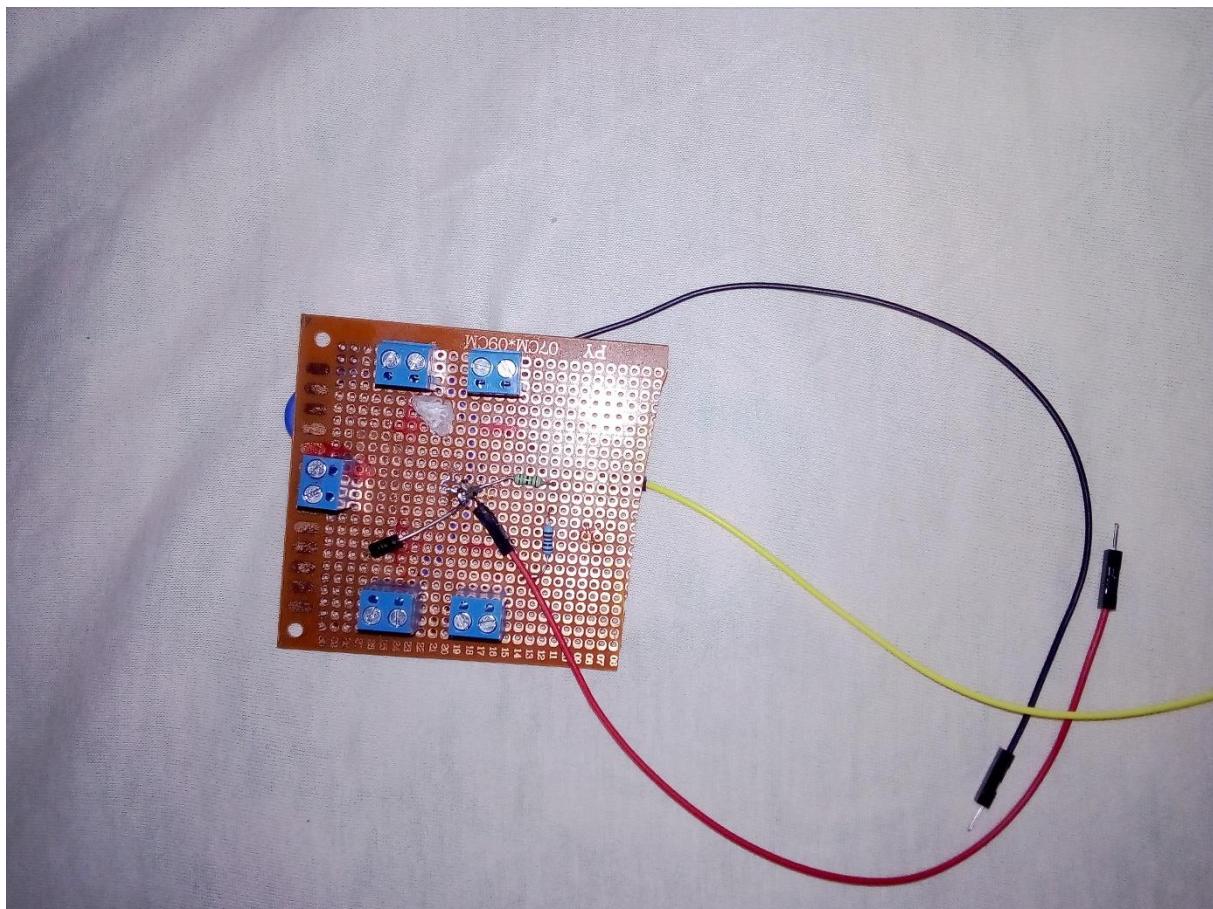


Fig 3-7 Carte de distribution d'alimentation

4/Connection de Batteries :

On connecte deux batterie LiPO en parallèle dans le but d'augmenter l'ampérage qui conduit à l'augmentation de la durée du vol du drone



Fig3-8 Câblage des batterie en parallèle

5/Moteur sans balais(Brushless motor)



Fig3-9 Moteur du quadraoptére(A2212/6T 2200KV)

6/Variateur de vitesse du moteur (ESC)

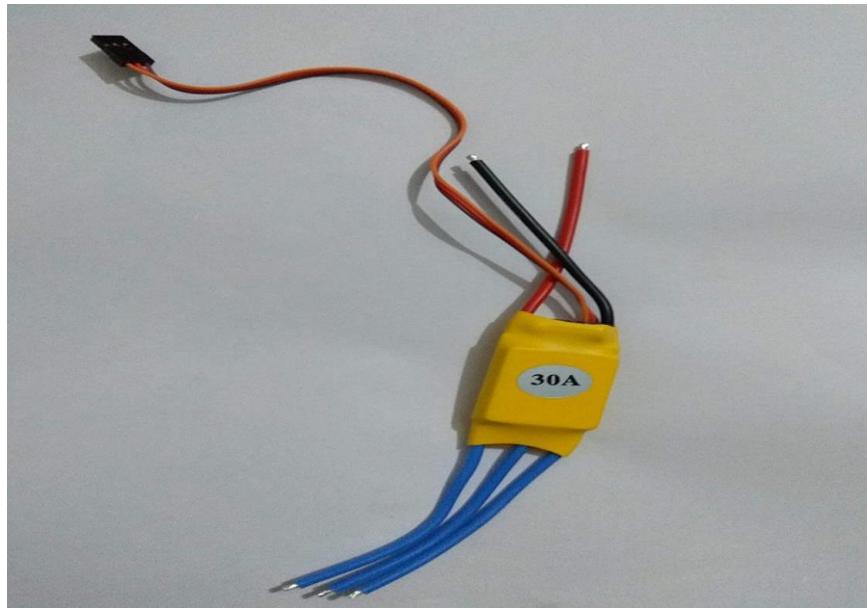


Fig3-10 ESC 30A BLDC ESC

7/MPU 9025 (Gyroscope + accelerometer + magnetometer)

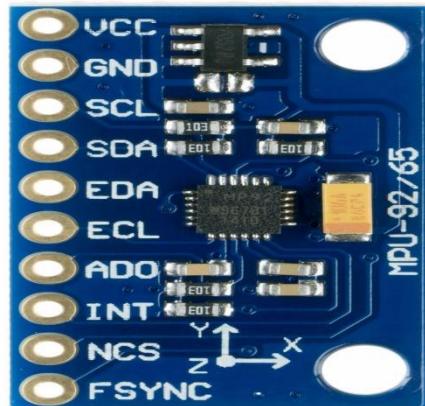


Fig3-11 MPU9025

La point qui se trouve dans le microcontrôleur du MPU9025 doit pointer vers le moteur arrière gauche , pour respecter le sens des axes déjà indiquer dans le code du drone

8/Equilibrage des Hélices

Les hélice utilisé sont des hélices ABS 8x4 .

Pour une quadracoptére on a 4 hélice (2 avec un sens horaire et les autres avec un sens anti-horaire) on doit faire un procédure de balance(consiste à vérifier que les deux pales qui constituent l'hélice ont le même poids) affin de réduire le bruit et la forte vibration qui peut affecter la stabilité du vol



Fig 3-12 une hélice déséquilibrer

La solution consiste à rendre les deux pales de l'hélice de même poids ,par exemple on peut coller quelque morceaux du scotch sur la partie de faible masse comme indique la figure si dessous



Fig3-13 ajoutant du scotch au pale droite de faible masse



Fig3-14 Résultat après l'équilibrage

Drone prêt



Fig3-15 conception finale du drone

II/Réalisation de la partie Soft

Le code est écrit sur l'environnement de développement **Arduino IDE** on essaie le maximum d'éviter les bibliothèque et les fonction prédéfinie pour que notre code soit le minimum possible de petit taille pour que le temps d'exécution soit plus rapide .

Algorithme d'un contrôleur de vol :

1/ La partie SETUP :

- Définir les variables globales
- Définir les gains des valeurs PID pour chaque axe (suite au résultats des tests)
- Initialisation des protocoles de communication
- Initialisation des différents capteurs
- Calibration du gyroscope (en effectuant la lecture des plusieurs échantillons puis on obtient la moyenne)
- Vérifier que les valeurs des chaines envoyés sont correctes avant de commencer le programme principale

2/ Le code principale (c'est le void loop ; un boucle infinie)

- Faire convertir les valeurs des chaines ("receiver values") dans un intervalle [1000µs-2000µs]
- Appliquer un filtre 80%-20% pour les données du gyroscope

- Activer ou désactiver les moteurs suite à des condition bien précis (pour l'activation chaine1 doit être au minimum et chaine 2 au maximum , pour stopper le moteur chaine1 au minimum et chaine 2 au minimum)
- Stocker l'inclinaison du drone désirer par le pilote suite aux valeurs des chaines 2,3 et 4 (YAW , ROLL , PITCH)
- Calculer le PID pour assurer la stabilité de drone et la consigne du pilote
- Calculer le niveau du batterie
- Envoyer la pulse PWM finale déterminée pour chaque variateur de vitesse ESC

3/ Routine d'interruption :

Il s'agit d'une interruption sur un front montant pour le signal PPM envoyé par le récepteur. Cette interruption calcule la durée entre deux fronts montant pour attribuer à chaque chaîne son valeur , elle contient aussi un procédure de synchronisation pour assurer la succession des chaines dans la même ordre

III/ Mise en œuvre du projet

Avant de tester le drone , on teste chaque composant séparé pour vérifier son fonctionnement , puis on doit calibrer les variateur de vitesse et les moteurs affin qu'ils commencent et s'arrêtent en même temps.

Les premières essais de vol la drone ont été sous une grande surveillance , on cherche pour chaque teste de vol les coefficients de PID qui donnent la meilleure stabilité et contrôle . Parfois la drone a été hors contrôle et ça peut être du au bruit , la soudure non professionnel des fils aussi le grand longueur des fils utilisée.

Pour la partie autonome du système on arrive au stade ou la drone peut s'orienter automatiquement pour suivre le chemin le plus court (en transformant le magnétomètre comme une boussole interne du drone)

Conclusion

Pour la premier version cette drone semble satisfait pour le travail qu'on a fait . Mais on doit toujours penser à améliorer et d'augmenter la performance du notre projet Et parmi les solutions qu'on doit l'implémenter le plut tôt possible , c'est de fabriquer une circuit imprimé qui englobe tous les capteurs avec le récepteur et le contrôleur de vol pour réduire le bruit et éviter les courts circuit lors de la vibration du drone , fabriquer une châssis imprimé en 3d (par exemple en fibre de carbone) ou bien avec la commande numérique pour que notre drone soit bien équilibré autour du centre de gravité , on doit aussi améliorer le microcontrôleur et travailler avec l'STM32F4 ou STM32F7 qui se caractérisent par un vitesse d'horloge plus forte , espace de mémoire plus grand et d'autre caractéristique plus forte que l'Atmega328.

Conclusion générale :

Au bout de ce projet on travail avec des différentes parties ; l'aérodynamique , le radiocommande , la programmation , les systèmes d'asservissement et d'autre domaines qui sont fortement liée à notre formation en mécatronique.

On apprend des plusieurs nouveaux notion et compétences pendant la réalisation de ce projet et surtout l'application de ce qu'on a apprend lors de ces deux année d'études .

Cette première version d'une drone autonome semble très satisfait lors des premières essai de vol avec un matériel non professionnel.

Ce projet ne s'arrête pas ici , on va améliorer cette version avec l'utilisation des plus fort microcontrôleur et capteur afin d'avoir une version professionnelle .

Bibliographies

A guide to understanding Lipo Batteries

<https://www.robotshop.com/media/files/pdf/hyperion-g5-50c-3s-1100mah-lipo-battery-User-Guide.pdf>

Applications of Magnetoresistive Sensors in Navigation Systems (Michael J. Caruso)
Honeywell Inc.

<https://www.generationrobots.com/media/module%20boussole%203%20axes%20HMC5883L/2913-3-Compass-Module-Application-Note-2.pdf>

Applications of Magnetic Sensors for Low Cost Compass Systems(Michael J. Caruso)
Honeywell, SSEC

<https://www.electronicwings.com/download/attachment=Fri-04-17-17-36-09.Applications-of-Magnetic-Sensors-for-Low-Cost-Compass-Systems.pdf>

Atmega8 datasheet

https://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-2486-8-bit-AVR-microcontroller-ATmega8_L_datasheet.pdf

Atmega328p datasheet

http://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-7810-Automotive-Microcontrollers-ATmega328P_Datasheet.pdf

NRF24L01 datasheet

https://www.sparkfun.com/datasheets/Components/SMD/nRF24L01Pluss_Preliminary_Product_Specification_v1_0.pdf

MPU9025 datasheet

<https://invensense.tdk.com/wp-content/uploads/2015/02/PS-MPU-9250A-01-v1.1.pdf>

30A BLDC ESC datasheet

https://www.optimusdigital.ro/index.php?controller=attachment&id_attachment=451

LM7805 5v Voltage regulator

<https://www.sparkfun.com/datasheets/Components/LM7805.pdf>

AMS1117 3.3v Voltage regulator

<http://www.advanced-monolithic.com/pdf/ds1117.pdf>

Netographie

<https://www.youtube.com/user/loraan>

<http://www.electrooobs.com/>

<https://www.youtube.com/user/Painless360/featured>

<https://www.youtube.com/channel/UC3c9WhUvKv2eoqZNSqAGQXg>

<https://www.ngdc.noaa.gov/geomag/WMM/image.shtml>

<http://www.brokking.net/>

<https://oscarliang.com/>

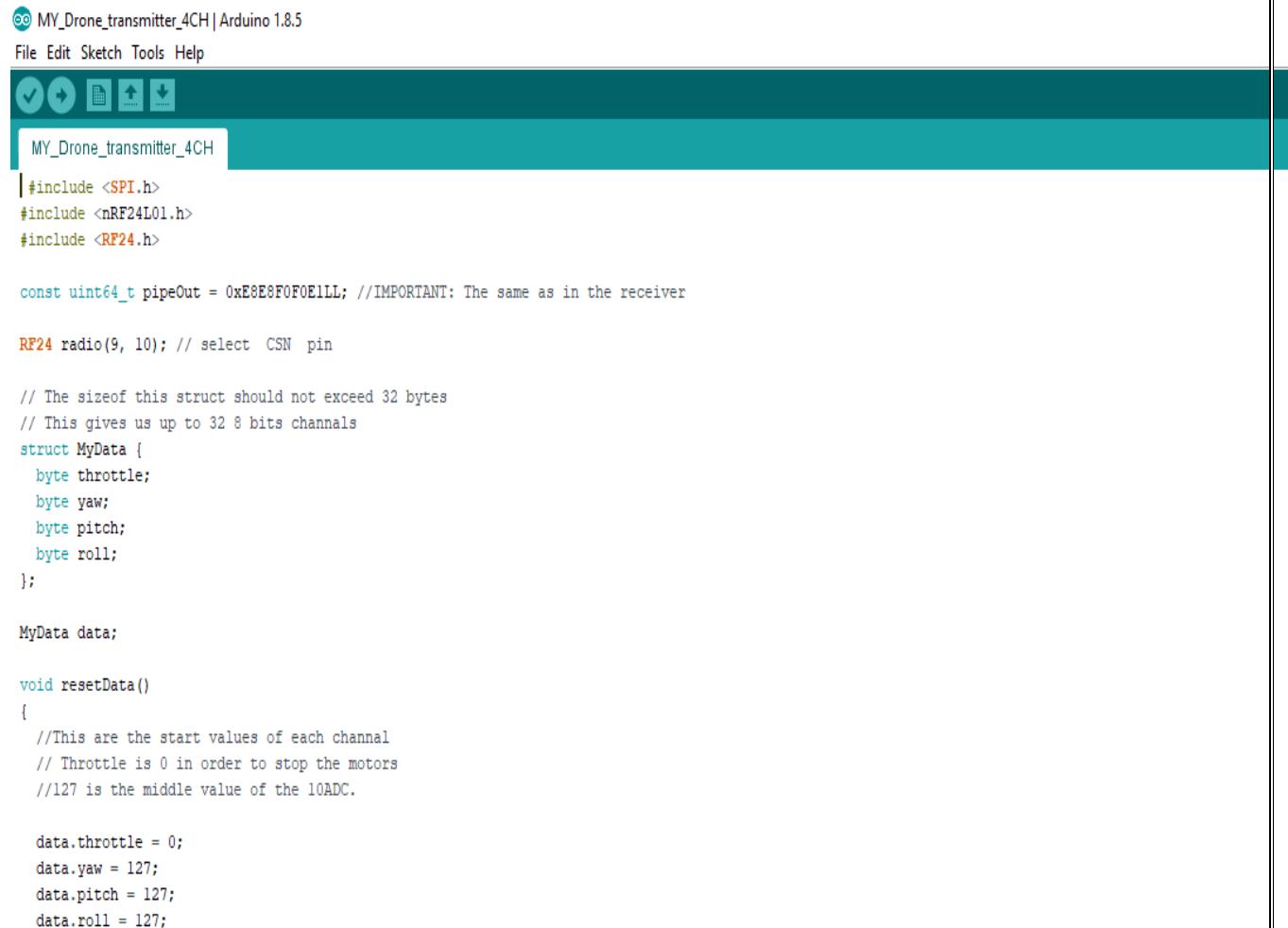
<https://www.movable-type.co.uk/scripts/latlong.html>

<https://www.igismap.com/map-tool/bearing-angle>

ANNEXE A

code du radiocommande

1/code du transmetteur



The screenshot shows the Arduino IDE interface with the sketch titled "MY_Drone_transmitter_4CH". The code is as follows:

```
MY_Drone_transmitter_4CH | Arduino 1.8.5
File Edit Sketch Tools Help
MY_Drone_transmitter_4CH
#include <SPI.h>
#include <nRF24L01.h>
#include <RF24.h>

const uint64_t pipeOut = 0xE8E8F0F0E1LL; //IMPORTANT: The same as in the receiver

RF24 radio(9, 10); // select CSN pin

// The sizeof this struct should not exceed 32 bytes
// This gives us up to 32 8 bits channels
struct MyData {
    byte throttle;
    byte yaw;
    byte pitch;
    byte roll;
};

MyData data;

void resetData()
{
    //This are the start values of each channal
    // Throttle is 0 in order to stop the motors
    //127 is the middle value of the 10ADC.

    data.throttle = 0;
    data.yaw = 127;
    data.pitch = 127;
    data.roll = 127;
```

```

void setup()
{
    //Start everything up
    radio.begin();
    radio.setAutoAck(false);
    radio.setDataRate(RF24_250KBPS);
    radio.openWritingPipe(pipeOut);
    resetData();
    Serial.begin(9600);

}

// *****
// Returns a corrected value for a joystick position that takes into account
// the values of the outer extents and the middle of the joystick range.
int mapJoystickValues(int val, int lower, int middle, int upper, bool reverse)
{
    val = constrain(val, lower, upper);
    if ( val < middle )
        val = map(val, lower, middle, 0, 128);
    else
        val = map(val, middle, upper, 128, 255);
    return ( reverse ? 255 - val : val ); //variable = (condition) ? optionA : optionB;
}

void loop()
{
    // The calibration numbers used here should be measured
    // for your joysticks till they send the correct values.
    data.throttle = mapJoystickValues( analogRead(A5), 0, 511, 1023, false );
    data.yaw      = mapJoystickValues( analogRead(A4), 0, 510, 1023, false );
    data.pitch    = mapJoystickValues( analogRead(A3), 0, 507, 1023, false );
    data.roll     = mapJoystickValues( analogRead(A2), 0, 512, 1023, false );

    radio.write(&data, sizeof(MyData));

}

```

2/Code de récepteur :

Le code reçoit les valeurs envoyées par le transmetteur , il va l'envoyer vers le contrôleur de vol sous forme d'un signal PPM

∞ MY_Drone_receiver_4CH | Arduino 1.8.5

File Edit Sketch Tools Help

```
#include <SPI.h>
#include <nRF24L01.h>
#include <RF24.h>

/////////////////////// PPM CONFIGURATION///////////////////
#define channel_number 4 //set the number of channels
#define sigPin A5 //set PPM signal output pin on the arduino
#define PPM_FrLen 27000 //set the PPM frame length in microseconds (1ms = 1000us)
#define PPM_PulseLen 400 //set the pulse length
/////////////////////// /////////////////////////////////

int ppm[channel_number];

const uint64_t pipeIn = 0xE8E8F0F0E1LL;

RF24 radio(9, 10);

// The sizeof this struct should not exceed 32 bytes
struct MyData {
    byte throttle;
    byte yaw;
    byte pitch;
    byte roll;
};

MyData data;
```

```

void resetData()
{
    // 'safe' values to use when no radio input is detected
    data.throttle = 0;
    data.yaw = 127;
    data.pitch = 127;
    data.roll = 127;

    setPPMValuesFromData();
}

void setPPMValuesFromData()
{
    ppm[0] = map(data.throttle, 0, 255, 1000, 2000);
    ppm[1] = map(data.yaw, 0, 255, 1000, 2000);
    ppm[2] = map(data.pitch, 0, 255, 1000, 2000);
    ppm[3] = map(data.roll, 0, 255, 1000, 2000);

}

/********************************************/

void setupPPM() {
    pinMode(sigPin, OUTPUT);
    digitalWrite(sigPin, 0); //set the PPM signal pin to the default state (off)

    cli();
    TCCR1A = 0; // set entire TCCR1 register to 0
    TCCR1B = 0;

    OCR1A = 100; // compare match register (not very important, sets the timeout for the first interrupt)
    TCCR1B |= (1 << WGM12); // turn on CTC mode
    TCCR1B |= (1 << CS11); // 8 prescaler: 0,5 microseconds at 16mhz
    TIMSK |= (1 << OCIE1A); // enable timer compare interrupt
    sei();
}

void setup()
{
    resetData();
    setupPPM();

    // Set up radio module
    radio.begin();
    radio.setDataRate(RF24_250KBPS); // Both endpoints must have this set the same
    radio.setAutoAck(false);

    radio.openReadingPipe(1,pipeIn);
    radio.startListening();
    Serial.begin(9600);
}

```

```
unsigned long lastRecvTime = 0;

void recvData()
{
    while ( radio.available() ) {
        radio.read(&data, sizeof(MyData));
        lastRecvTime = millis();
    }
}

/***********************/

void loop()
{
    recvData();
    Serial.println(data.throttle);

    unsigned long now = millis();
    if ( now - lastRecvTime > 1000 ) {
        // signal lost?
        resetData();
    }

    setPPMValuesFromData();
}

/****************/
```

Routine d'interruption pour la génération d'un signal PPM

```
#define clockMultiplier 2
ISR(TIMER1_COMPA_vect) {
    static boolean state = true;
    TCNT1 = 0;
    if ( state ) {
        //end pulse
        PORTC = PORTC & ~B00100000;
        // turn pin 2 off. Could also use: digitalWrite(sigPin,0) ;
        OCR1A = PPM_PulseLen * 2 ;
        state = false;
    }
    else {
        //start pulse
        static byte cur Chan numb;
        static unsigned int calc rest;
        state = true;
        if(cur Chan numb >= channel number) {
            PORTC = PORTC & ~B00100000;
            OCR1A = 35000 ;
            cur Chan numb = 0;
        }
        else {
            OCR1A = (ppm[cur Chan numb] - PPM_PulseLen) * 2;
            PORTC = PORTC | B00100000;
            calc rest += ppm[cur Chan numb];
            cur Chan numb++;
        }
    }
}
```

ANNEXE B

Code du contrôleur de vol (pour le mode normale)

Avant de téléverser le code on doit tester d'abord tous les capteurs et les moteurs pour vérifier leur fonctionnement , on doit aussi faire une calibration pour le moteur et son ESC pour qu'ils commencent et s'arrêtent à la même moment

Les variables globales du code

```
////////////////////////////////////////////////////////////////
//PID gain and limit settings
////////////////////////////////////////////////////////////////
float pid_p_gain_roll = 1.3 ;
float pid_i_gain_roll = 0.05 ;
float pid_d_gain_roll = 15 ;           )
int pid_max_roll = 400;           //Maximum output of the PID-controller (+/-)

float pid_p_gain_pitch = pid_p_gain_roll;
float pid_i_gain_pitch = pid_i_gain_roll;
float pid_d_gain_pitch = pid_d_gain_roll;
int pid_max_pitch = pid_max_roll;      //Maximum output of the PID-controller (+/-)

float pid_p_gain_yaw = 4 ;
float pid_i_gain_yaw = 0.02 ;
float pid_d_gain_yaw = 0 ;
int pid_max_yaw = 400;           //Maximum output of the PID-controller (+/-)

//Declaring global variables
////////////////////////////////////////////////////////////////
byte last_channel_1, last_channel_2, last_channel_3, last_channel_4;
byte eeprom_data[36];
byte highByte, lowByte;
int receiver_input_channel_1, receiver_input_channel_2, receiver_input_channel_3, receiver_input_channel_4;
int counter_channel_1, counter_channel_2, counter_channel_3, counter_channel_4, loop_counter;
int esc_1, esc_2, esc_3, esc_4;
int throttle, battery_voltage;

int cal_int, start, gyro_address;
int receiver_input[5];
unsigned long timer_channel_1, timer_channel_2, timer_channel_3, timer_channel_4, esc_timer, esc_loop_timer;
unsigned long timer_1, timer_2, timer_3, timer_4, current_time;
unsigned long loop_timer;
double gyro_pitch, gyro_roll, gyro_yaw;
double gyro_axis[4], gyro_axis_cal[4];
float pid_error_temp;
float pid_i_mem_roll, pid_roll_setpoint, gyro_roll_input, pid_output_roll, pid_last_roll_d_error;
float pid_i_mem_pitch, pid_pitch_setpoint, gyro_pitch_input, pid_output_pitch, pid_last_pitch_d_error;
float pid_i_mem_yaw, pid_yaw_setpoint, gyro_yaw_input, pid_output_yaw, pid_last_yaw_d_error;
unsigned long ch[4] ;
unsigned long t[5] ;
int pulse = 0;
```

Le routine Setup :

Il consiste à initialiser les protocoles de communication, les capteurs , faire leur calibration (on prendre la moyenne des plusieurs lectures, cet opération est marquée par le clignotement du LED) et vérifier si le contrôleur de vol reçoit un signal correct du radiocommande .

```

void setup(){
  //Serial.begin(57600);
  //Read EEPROM for fast access data.
  for(start = 0; start <= 35; start++) eeprom_data[start] = EEPROM.read(start);
  gyro_address = eeprom_data[32]; //Store the gyro address in the variable.
  Serial.begin(9600);
  Wire.begin(); //Start the I2C as master.
  DDRB |= B00111110; //Configure digital poort 9,10,11,12 and 13 as output.

  //Use the led on the Arduino for startup indication.
  digitalWrite(13,HIGH); //Turn on the warning led.
  //Check the EEPROM signature to make sure that the setup program is executed
  while(eeprom_data[33] != 'J' || eeprom_data[34] != 'M' || eeprom_data[35] != 'B') delay(10);
  set_gyro_registers(); //Set the specific gyro registers.///////////
  for (cal_int = 0; cal_int < 1250 ; cal_int ++){ //Wait 5 seconds before continuing.///////////
    PORTB |= B00011110; //Set digital poort 9, 10, 11 and 12 high.
    delayMicroseconds(1000); //Wait 1000us.
    PORTB &= B11100001; //Set digital poort 9, 10, 11 and 12 low.
    delayMicroseconds(3000); //Wait 3000us.
  } //////////////

  //Let's take multiple gyro data samples so we can determine the average gyro offset (calibration).
  for (cal_int = 0; cal_int < 2000 ; cal_int ++){ //Take 2000 readings for calibration.///////////
    if(cal_int % 15 == 0) digitalWrite(13, !digitalRead(13)); //Change the led status to indicate calibration.
    gyro_signalen(); //Read the gyro output.
    gyro_axis_cal[1] += gyro_axis[1]; //Ad roll value to gyro_roll_cal.
    gyro_axis_cal[2] += gyro_axis[2]; //Ad pitch value to gyro_pitch_cal.
    gyro_axis_cal[3] += gyro_axis[3]; //Ad yaw value to gyro_yaw_cal.

  //Now that we have 2000 measures, we need to devide by 2000 to get the average gyro offset.
  gyro_axis_cal[1] /= 2000; //Divide the roll total by 2000.
  gyro_axis_cal[2] /= 2000; //Divide the pitch total by 2000.
  gyro_axis_cal[3] /= 2000; //Divide the yaw total by 2000.*///////////

  cli();
  PCICR |= (1<<PCIE0);
  PCMSK0 |= (1<<PCINT0);
  sei();

  //Wait until the receiver is active and the throttle is set to the lower position.
  while(receiver_input_channel_1 < 990 || receiver_input_channel_1 > 1020 || receiver_input_channel_2 < 1400){
    receiver_input_channel_1 = convert_receiver_channel(1); //Convert the actual receiver signals for throttle to the standard 1000 - 2000us
    receiver_input_channel_2 = convert_receiver_channel(2); //Convert the actual receiver signals for yaw to the standard 1000 - 2000us
    receiver_input_channel_3 = convert_receiver_channel(3);
    receiver_input_channel_4 = convert_receiver_channel(4);
    start++; //While waiting increment start whith every loop.
  //We don't want the esc's to be beeping annoyingly. So let's give them a 1000us puls while waiting for the receiver inputs.
  PORTB |= B00011110;
  delayMicroseconds(1000); //Wait 1000us.
  PORTB &= B11100001;
  delay(3); //Wait 3 milliseconds before the next loop.
  if(start == 125){ //Every 125 loops (500ms).
    digitalWrite(13, !digitalRead(13)); //Change the led status.
    start = 0; //Start again at 0.
  }
}
start = 0; //Set start back to 0.//

```

Void Loop

C'est le lieu du code principale ,il s'agit d'un boucle infinie qui calcule les signal PWM à envoyer vers les variateurs de vitesse ESC , tous les calcule du système d'asservissement , le stockage des données des capteurs ,lecture des valeurs des chaines envoyés, le calcule de niveau de batterie les conditions pour le START et stop des moteurs

```

void loop(){

    receiver_input_channel_1 = convert_receiver_channel(1);      //Convert the actual receiver signals for pitch to the standard 1000 - 2000us.
    receiver_input_channel_2 = convert_receiver_channel(2);      //Convert the actual receiver signals for roll to the standard 1000 - 2000us.
    receiver_input_channel_3 = convert_receiver_channel(3);      //Convert the actual receiver signals for throttle to the standard 1000 - 2000us.
    receiver_input_channel_4 = convert_receiver_channel(4);      //Convert the actual receiver signals for yaw to the standard 1000 - 2000us.

    gyro_roll_input = (gyro_roll_input * 0.8) + ((gyro_roll/ 57.14286 ) * 0.2);           //Gyro pid input is deg/sec./57.14286
    gyro_pitch_input = (gyro_pitch_input * 0.8) + ((gyro_pitch/ 57.14286 ) * 0.2);           //Gyro pid input is deg/sec.
    gyro_yaw_input = (gyro_yaw_input * 0.8) + ((gyro_yaw/57.14286 ) * 0.2);                //Gyro pid input is deg/sec.

    //For starting the motors: throttle low and yaw left (step 1).
    if((receiver_input_channel_1 < 1050) && (receiver_input_channel_2 > 1950)){start = 1;}
    }

    //Stopping the motors: throttle low and yaw right.
    if(receiver_input_channel_1 < 1050 && receiver_input_channel_2 < 1050)start = 0;
    if(receiver_input_channel_1 > 2060 || receiver_input_channel_2 >2060 || receiver_input_channel_3>2060 || receiver_input_channel_4>2060 )start = 0 ;

    //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_4 > 1508)pid_roll_setpoint = (receiver_input_channel_4 - 1508)/3.0;
    else if(receiver_input_channel_4 < 1492)pid_roll_setpoint = (receiver_input_channel_4 - 1492)/3.0;

    //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_3 > 1508)pid_pitch_setpoint = (receiver_input_channel_3 - 1508)/3.0;
    else if(receiver_input_channel_3 < 1492)pid_pitch_setpoint = (receiver_input_channel_3 - 1492)/3.0;

    //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_1 > 1050){ //Do not yaw when turning off the motors.
        if(receiver_input_channel_2 > 1508)pid_yaw_setpoint = (receiver_input_channel_2 - 1508)/3.0;
        else if(receiver_input_channel_2 < 1492)pid_yaw_setpoint = (receiver_input_channel_2 - 1492)/3.0;
    }
    //PID inputs are known. So we can calculate the pid output.
    calculate_pid();           /////////////////////////////////
    //The battery voltage is needed for compensation.
    //A complementary filter is used to reduce noise.
    0.09853 = 0.08 * 1.2317.
    battery_voltage = battery_voltage * 0.92 + (analogRead(0) + 65) * 0.09853;
    //Turn on the led if battery voltage is to low.
    if(battery_voltage < 1030 && battery_voltage > 600)digitalWrite(13, HIGH);
    throttle = receiver_input_channel_1;           //We need the throttle signal as a base signal.
}

```

Calcule des valeurs de commande pour chaque ESC

```

if (start == 1){
    if (throttle > 1800) throttle = 1800;                                //The motors are started.
    //We need some room to keep full control at full throttle
    esc_1 = throttle - pid_output_pitch + pid_output_roll - pid_output_yaw; //Calculate the pulse for esc 1 (front-right - CCW)
    esc_2 = throttle+ pid_output_pitch + pid_output_roll + pid_output_yaw; //Calculate the pulse for esc 2 (rear-right - CW)
    esc_3 = (throttle+ pid_output_pitch - pid_output_roll - pid_output_yaw); //Calculate the pulse for esc 3 (rear-left - CCW)
    esc_4 = (throttle- pid_output_pitch - pid_output_roll + pid_output_yaw); //Calculate the pulse for esc 4 (front-left - CW)
    if (battery_voltage < 1240 && battery_voltage > 800){               //Is the battery connected?
        esc_1 += esc_1 * ((l1240 - battery_voltage)/(float)3500);          //Compensate the esc-1 pulse for voltage drop.
        esc_2 += esc_2 * ((l1240 - battery_voltage)/(float)3500);          //Compensate the esc-2 pulse for voltage drop.
        esc_3 += esc_3 * ((l1240 - battery_voltage)/(float)3500);          //Compensate the esc-3 pulse for voltage drop.
        esc_4 += esc_4 * ((l1240 - battery_voltage)/(float)3500);          //Compensate the esc-4 pulse for voltage drop.
    }
    if (esc_1 < 1100) esc_1 = 1100;                                     //Keep the motors running.
    if (esc_2 < 1100) esc_2 = 1100;                                     //Keep the motors running.
    if (esc_3 < 1100) esc_3 = 1100;                                     //Keep the motors running.
    if (esc_4 < 1100) esc_4 = 1100;                                     //Keep the motors running.
    if(esc_1 > 2000)esc_1 = 2000;                                     //Limit the esc-1 pulse to 2000us
    if(esc_2 > 2000)esc_2 = 2000;                                     //Limit the esc-2 pulse to 2000us.
    if(esc_3 > 2000)esc_3 = 2000;                                     //Limit the esc-3 pulse to 2000us.
    if(esc_4 > 2000)esc_4 = 2000;                                     //Limit the esc-4 pulse to 2000us.
}
else{
    esc_1 = 1000;                                                 //If start is not 2 keep a 1000us pulse for ess-1.
    esc_2 = 1000;                                                 //If start is not 2 keep a 1000us pulse for ess-2.
    esc_3 = 1000;                                                 //If start is not 2 keep a 1000us pulse for ess-3.
    esc_4 = 1000;                                                 //If start is not 2 keep a 1000us pulse for ess-4.
}

```

```

    //All the information for controlling the motor's is available.
    //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.
    PORTB |= B00011110 ;
    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.

    while(PORTB >= 2){
        esc_loop_timer = micros();
        if(timer_channel_1 <= esc_loop_timer)PORTB &= B11111101;
        if(timer_channel_2 <= esc_loop_timer)PORTB &= B11110111;
        if(timer_channel_3 <= esc_loop_timer)PORTB &= B11101111;
        if(timer_channel_4 <= esc_loop_timer)PORTB &= B1101111;
    }
}

```

Routine interruption

il s'agit d'une interruption sur front montant du signal PPM qui vient du récepteur affin de transformer le valeur de chaque chaine dans l'intervalle de [1000µs-2000µs] et les stocker

```

ISR(PCINT0_vect)
{
    if ( digitalRead(8)== HIGH )
    {
        switch (pulse)
        {
            case 1:
                t[1] = micros();
                ch[0] = t[1] - t[0] ;
                receiver_input[1] = ch[0] ;
                pulse++;
                if (ch [0] > 30000 )
                {
                    t[0] = t[1];
                    pulse=1;
                }
                break;

            case 2:
                t[2] = micros();
                ch[1] = t[2] - t[1] ;
                receiver_input[2] = ch[1] ;
                pulse++;
                if (ch [1] > 30000)
                {
                    t[0] = t[2];
                    pulse=1;
                }
        }
    }
}

```

```

    }
    break;
  case 3:
    t[3] = micros();
    ch[2] = t[3] - t[2] ;
    receiver_input[3] = ch[2] ;
    pulse++;

    if (ch [2] > 30000)
    {
      t[0] = t[3];
      pulse = 1;
    }
    break;
  case 4:
    t[0] = micros();
    ch[3] = (t[0] - t[3])-17900 ;
    receiver_input[4] = ch[3] ;
    pulse = 1 ;
    break;
  default:
    t[0] = micros();
    pulse++;
    break;
  }
}
}
/////////////////////////////////////////////////////////////////

```

Fonction pour la lecture des données de gyroscope

```

void gyro_signalen(){
  if(eeprom_data[31] == 1){
    Wire.beginTransmission(gyro_address);
    Wire.write(0x43); //Start communication with the gyro
    Wire.endTransmission(); //Start reading @ register 43h and auto increment with every read
    Wire.requestFrom(gyro_address,6); //End the transmission
    while(Wire.available() < 6); //Request 6 bytes from the gyro
    gyro_axis[1] = Wire.read()<<8|Wire.read(); //Wait until the 6 bytes are received
    gyro_axis[2] = Wire.read()<<8|Wire.read(); //Read high and low part of the angular data
    gyro_axis[3] = Wire.read()<<8|Wire.read(); //Read high and low part of the angular data
    gyro_axis[4] = Wire.read()<<8|Wire.read(); //Read high and low part of the angular data
  }

  if(cal_int == 2000){
    gyro_axis[1] -= gyro_axis_cal[1]; //Only compensate after the calibration
    gyro_axis[2] -= gyro_axis_cal[2]; //Only compensate after the calibration
    gyro_axis[3] -= gyro_axis_cal[3]; //Only compensate after the calibration
  }
  gyro_roll = gyro_axis[eeprom_data[28] & 0b00000011];
  if(eeprom_data[28] & 0b10000000)gyro_roll *= -1;
  gyro_pitch = gyro_axis[eeprom_data[29] & 0b00000011];
  if(eeprom_data[29] & 0b10000000)gyro_pitch *= -1;
  gyro_yaw = gyro_axis[eeprom_data[30] & 0b00000011];
  if(eeprom_data[30] & 0b10000000)gyro_yaw *= -1;
}

```

Fonction pour le calcul de PID (les formules sont les mêmes indiquées en chapitre II)

```

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;

    //Pitch calculations
    pid_error_temp = gyro_pitch_input - pid_pitch_setpoint;
    pid_i_mem_pitch += pid_i_gain_pitch * pid_error_temp;
    if(pid_i_mem_pitch > pid_max_pitch)pid_i_mem_pitch = pid_max_pitch;
    else if(pid_i_mem_pitch < pid_max_pitch * -1)pid_i_mem_pitch = pid_max_pitch * -1;

    pid_output_pitch = pid_p_gain_pitch * pid_error_temp + pid_i_mem_pitch + pid_d_gain_pitch * (pid_error_temp - pid_last_pitch_d_error);
    if(pid_output_pitch > pid_max_pitch)pid_output_pitch = pid_max_pitch;
    else if(pid_output_pitch < pid_max_pitch * -1)pid_output_pitch = pid_max_pitch * -1;

    pid_last_pitch_d_error = pid_error_temp;

    //Yaw calculations
    pid_error_temp = gyro_yaw_input - pid_yaw_setpoint;
    pid_i_mem_yaw += pid_i_gain_yaw * pid_error_temp;
    if(pid_i_mem_yaw > pid_max_yaw)pid_i_mem_yaw = pid_max_yaw;
    else if(pid_i_mem_yaw < pid_max_yaw * -1)pid_i_mem_yaw = pid_max_yaw * -1;

    pid_output_yaw = pid_p_gain_yaw * pid_error_temp + pid_i_mem_yaw + pid_d_gain_yaw * (pid_error_temp - pid_last_yaw_d_error);
    if(pid_output_yaw > pid_max_yaw)pid_output_yaw = pid_max_yaw;
    else if(pid_output_yaw < pid_max_yaw * -1)pid_output_yaw = pid_max_yaw * -1;

    pid_last_yaw_d_error = pid_error_temp;
}

```

Fonction pour convertir les valeurs des chaines et leur axe s'il est inverser ou non

```

int convert_receiver_channel(byte function){
    byte channel, reverse;                                //First we declare some local variables
    int low, center, high, actual;
    int difference;

    channel = eeprom_data=function + 23] & 0b00000111;      //What channel corresponds with the specific function
    if(eeprom_data=function + 23] & 0b10000000)reverse = 1;    //Reverse channel when most significant bit is set
    else reverse = 0;                                      //If the most significant is not set there is no reverse

    actual = receiver_input=function];                      //Read the actual receiver value for the corresponding function///////////
    low = (eeprom_data[channel * 2 + 15] << 8) | eeprom_data[channel * 2 + 14]; //Store the low value for the specific receiver input channel
    center = (eeprom_data[channel * 2 - 1] << 8) | eeprom_data[channel * 2 - 2]; //Store the center value for the specific receiver input channel
    high = (eeprom_data[channel * 2 + 7] << 8) | eeprom_data[channel * 2 + 6]; //Store the high value for the specific receiver input channel

    if(actual < center){                                 //The actual receiver value is lower than the center value
        if(actual < low)actual = low;                     //Limit the lowest value to the value that was detected during setup
        difference = ((long)(center - actual) * (long)500) / (center - low); //Calculate and scale the actual value to a 1000 - 2000us value
        if(reverse == 1)return 1500 + difference;          //If the channel is reversed
        else return 1500 - difference;                     //If the channel is not reversed
    }
    else if(actual > center){                            //The actual receiver value is higher than the center value
        if(actual > high)actual = high;                  //Limit the lowest value to the value that was detected during setup
        difference = ((long)(actual - center) * (long)500) / (high - center); //Calculate and scale the actual value to a 1000 - 2000us value
        if(reverse == 1)return 1500 - difference;          //If the channel is reversed
        else return 1500 + difference;                     //If the channel is not reversed
    }
    else return 1500;
}

```

Fonction pour initialiser le capteur 9025

```

void set_gyro_registers(){
    if(eeprom_data[31] == 1){
        Wire.beginTransmission(gyro_address);
        Wire.write(0x6B);
        Wire.write(0x00);
        Wire.endTransmission();

        Wire.beginTransmission(gyro_address);
        Wire.write(0x1B);
        Wire.write(0x08);
        Wire.endTransmission();

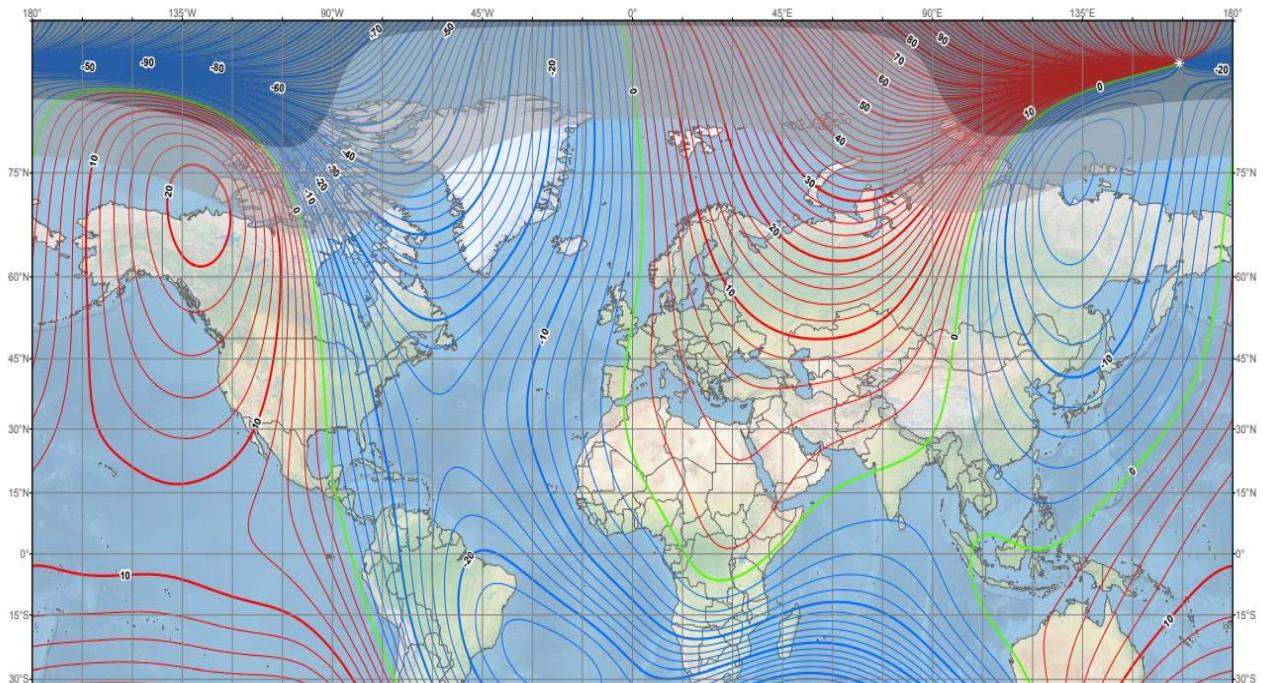
        Wire.beginTransmission(gyro_address);
        Wire.write(0x1B);
        Wire.endTransmission();
        Wire.requestFrom(gyro_address, 1);
        while(Wire.available() < 1);
        if(Wire.read() != 0x08){
            digitalWrite(13,HIGH);
            while(1)delay(10);
        }
        Wire.beginTransmission(gyro_address);
        Wire.write(0x1A);
        Wire.write(0x03);
        Wire.endTransmission();
    }
}

```

ANNEXE C

Cette carte représente la différence en degré entre le nord magnétique et le nord géographie et ça varie selon la position géographique (par exemple en Tunisie c'est 3°)

**US/UK World Magnetic Model - Epoch 2020.0
Main Field Declination (D)**



Cette carte représente l'inclinaison de la direction du champ magnétique par rapport à la surface de la terre, on doit prendre en compte ce valeur d'inclinaison selon notre position géographique ou on travaille pour bien déterminer le nord géographie à l'aide du magnétomètre

US/UK World Magnetic Model - Epoch 2020.0
Main Field Inclination (I)

