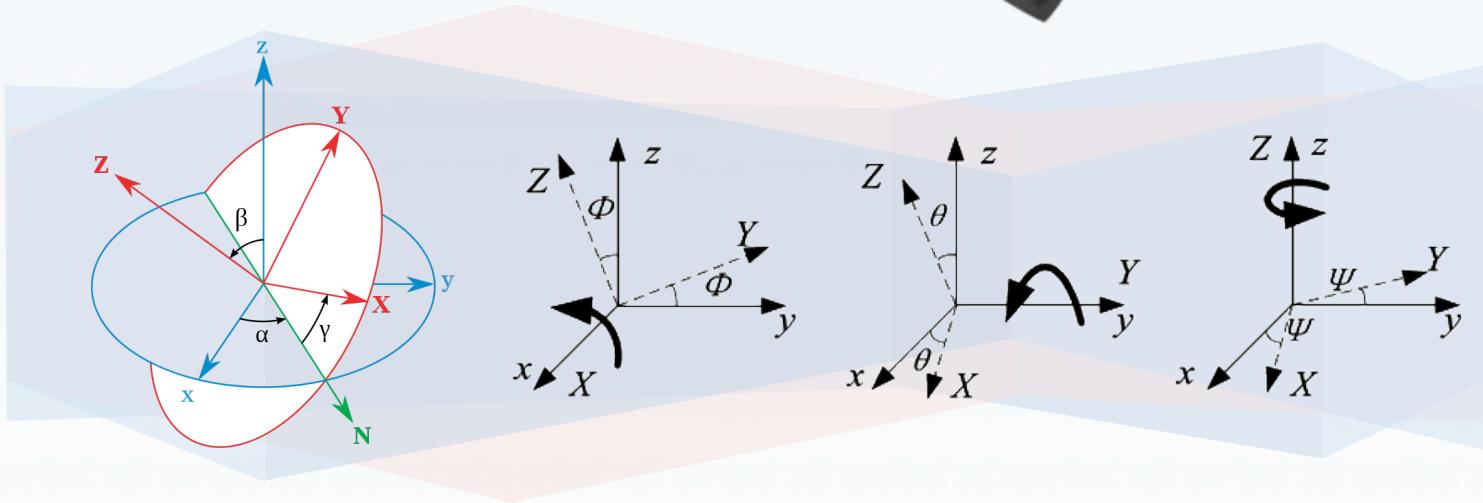


# Quadricoptère



dsPIC33 & MPU-9150

Bonzel Pierre – Eveillard Antoine – Renolleau Cyril



<b>Introduction .....</b>	<b>3</b>
<b>I. Principe de vol d'un quadricoptère .....</b>	<b>4</b>
1. Général .....	4
<b>II. Éléments constitutifs .....</b>	<b>7</b>
1. Structure – Châssis .....	7
2. Radiocommande Hautes fréquences .....	7
3. Electronic Speed Control (ESC) .....	8
4. Moteurs .....	9
5. Pales .....	11
6. Batterie & autonomie .....	11
7. Carte de distribution .....	12
<b>III. Composants de la carte de contrôle .....</b>	<b>13</b>
1. Microcontrôleur dsPIC33FJ128MC802 .....	13
2. Inertial Measurement Unit – MPU-9150 – 9 Dof .....	13
3. MAX232 .....	15
4. Régulateurs de tension .....	15
5. Autre composants .....	15
<b>IV. Réalisation de la carte de contrôle .....</b>	<b>17</b>
1. Alimentation .....	17
2. Tests et implémentation du microcontrôleur .....	18
3. Décodage d'un signal PWM a l'aide de l'Input Capture .....	19
4. I2C Master mode .....	20
<b>V. Système fonctionnel .....</b>	<b>21</b>
1. Montage du quadricoptère .....	21
2. Limitation des vibrations .....	21
3. Tests de vol .....	22
a. Sans manette ni IMU .....	22
b. Avec manette, sans IMU .....	22
<b>V. Architecture logiciel .....</b>	<b>23</b>
1. Configurations et Horloge .....	23
2. Structure générale : utilisation des timers .....	24
2. Module input capture .....	26
3. Présentation de la fonction UART .....	26
<b>VI. Solutions pour la stabilisation .....</b>	<b>28</b>
1. De la donnée brute aux angles .....	28
2. Angles d'Euler .....	28
3. Filtre de Kalman .....	29
4. Implantation et réglage des PIDs .....	30
<b>Conclusion .....</b>	<b>31</b>
<b>Annexes .....</b>	<b>32</b>
1. Annexe 1 – Schéma de la carte de contrôle .....	32
2. Annexe 2 – Bill Of Material .....	33
3. Annexe 3 – Liste des tâches .....	34

# Introduction

Un quadrioptère est un aéronef à voilure tournante comportant quatre hélices montées sur autant de moteurs pour sa sustentation. Un drone est un aérodynne sans pilote embarqué et télécommandé qui emporte une charge utile destinées à des missions de surveillance, renseignement, combat etc.

Depuis les années soixante, le drone a été fortement développé à des fins militaires, il permet, en effet, la surveillance et l'intervention militaire chez l'ennemi sans encourir de risques humains. Pour preuve, à partir des années 2000, le drone a été de tous les conflits et opérations de maintien de la paix.

Or, le design du quadrioptère est très utilisé lors de la conception des drones. Il possède, en effet, plusieurs avantages par rapport à d'autres types d'aérodynes bien connus.

Avantages du quadrioptère sur l'avion :

- Plus manœuvrable et facile à piloter
- Peu rester stable et immobile en vol
- Ne nécessite pas de rampe de lancement pour décoller

Avantages du quadrioptère sur l'hélicoptère :

- Les moteurs sont plus petits ce qui fait dépenser moins d'énergie cinétique
- On peut plus facilement créer une structure pour protéger les moteurs ce qui permet des vols dans des environnements hostiles

Comme nombre d'innovations ayant eu une première application militaire (internet, GPS etc.) le drone est, depuis quelques années, développé dans le domaine civil et accessible au grand public. Le drone civil, ou aéromodèle télépiloté comme il convient de l'appeler connaît dès lors un succès spectaculaire. En effet, selon la Direction Générale de l'Aviation Civile (DGAC) le nombre d'entreprises créées autour du drone civil en France a augmenté de 350% en 2013. Aujourd'hui, le marché des drones civils est très varié, on en trouve de 34€ à 4000€ et on répertorie de 150 000 à 200 000 aéromodèles télépilotés en France. On peut notamment citer le plus célèbre de ces drones accessible au public, le AR Drone de l'entreprise française Parrot qui a annoncé en juin 2013 au Paris Air Show avoir vendu plus de 500 000 de leurs quadrioptères.

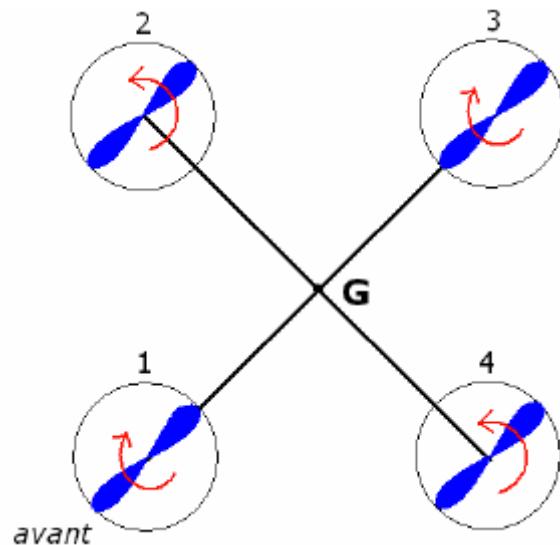
Face à l'actuel développement du marché des drones de type quadrioptère et étant passionnés de modélisme, nous avons décidé dans le cadre de ce projet de quatrième année d'en construire un nous-mêmes. En effet, nous avons pu voir dans les labos de l'école qu'un groupe de cinquième année travaillait sur la conception d'un quadrioptère et voyant les prouesses que leur appareil peut réaliser nous sommes alors demandés pourquoi pas nous ?

De plus, ce projet mélangeant plusieurs disciplines nous a permis de mettre en pratique les enseignements dispensés à l'ESME Sudria

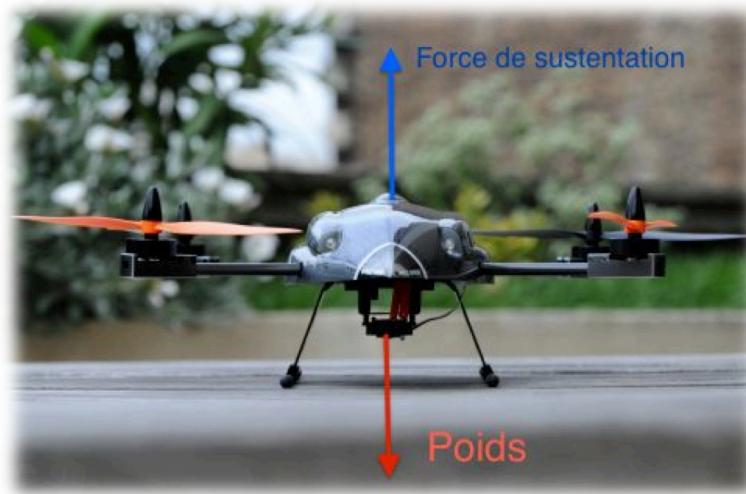
# I. Principe de vol d'un quadricoptère

## 1. Général

Un quadricoptère est composé de quatre hélices montées sur quatre moteurs. Pour qu'il puisse décoller, il est nécessaire que deux d'entre eux tournent dans le sens horaire, et les deux autres dans le sens antihoraire. De plus, les moteurs qui sont face à face doivent tourner dans le même sens. Le principe est schématisé ci dessous :



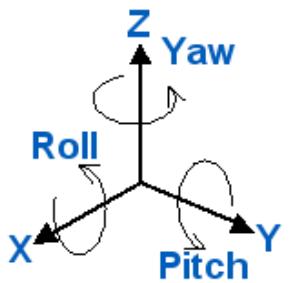
Lorsque les hélices tournent, une force opposée au poids du quadricoptère se créer. On appelle cette force, force de sustentation, son but est de maintenir un corps au dessus d'une surface sans contact avec elle. Par conséquent lorsque la force de sustentation créée est supérieure au poids du quadricoptère, celui-ci décolle. Nous avons représenté les forces qui s'exercent sur un quadricoptère sur la photo ci dessous :



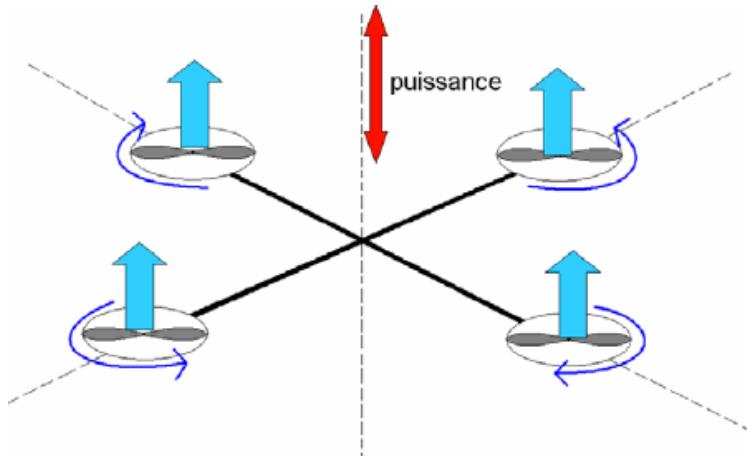
Un quadrioptère est contrôlable sur trois axes de rotation :

- Le yaw (ou lacet)
- Le roll (ou roulis)
- Le pitch (ou tangage)

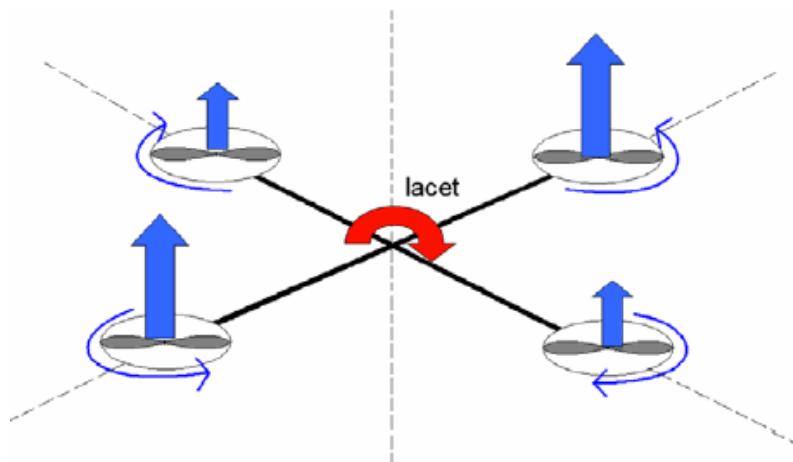
Ce qui implique quatre types de déplacements possibles :



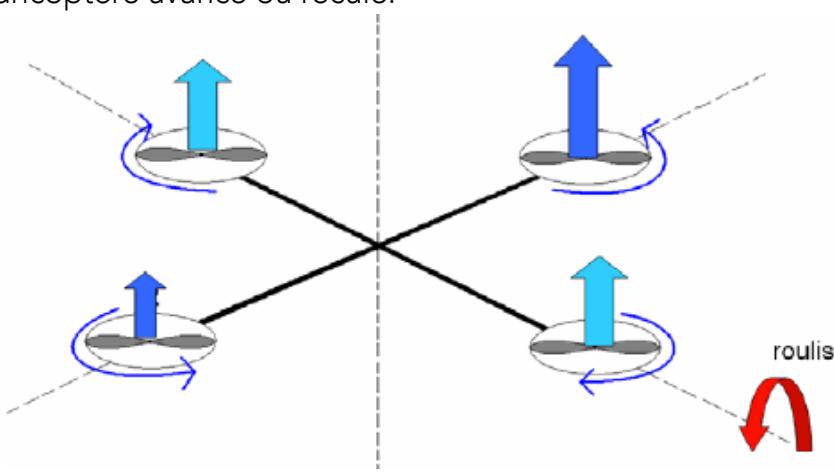
- Déplacement vertical : les quatre moteurs délivrent une puissance uniforme, le quadrioptère gagne en altitude.



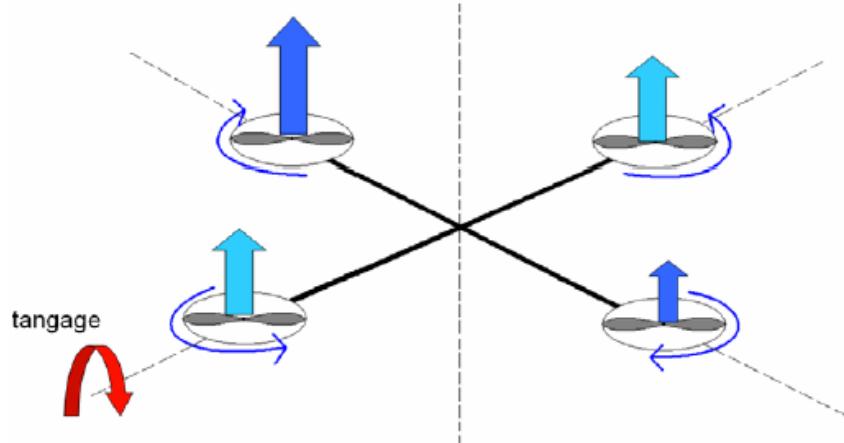
- Rotation autour de l'axe de lacet (ou yaw) : un couple de moteurs fournit plus de puissance que l'autre couple. Le quadrioptère tourne sur lui-même.



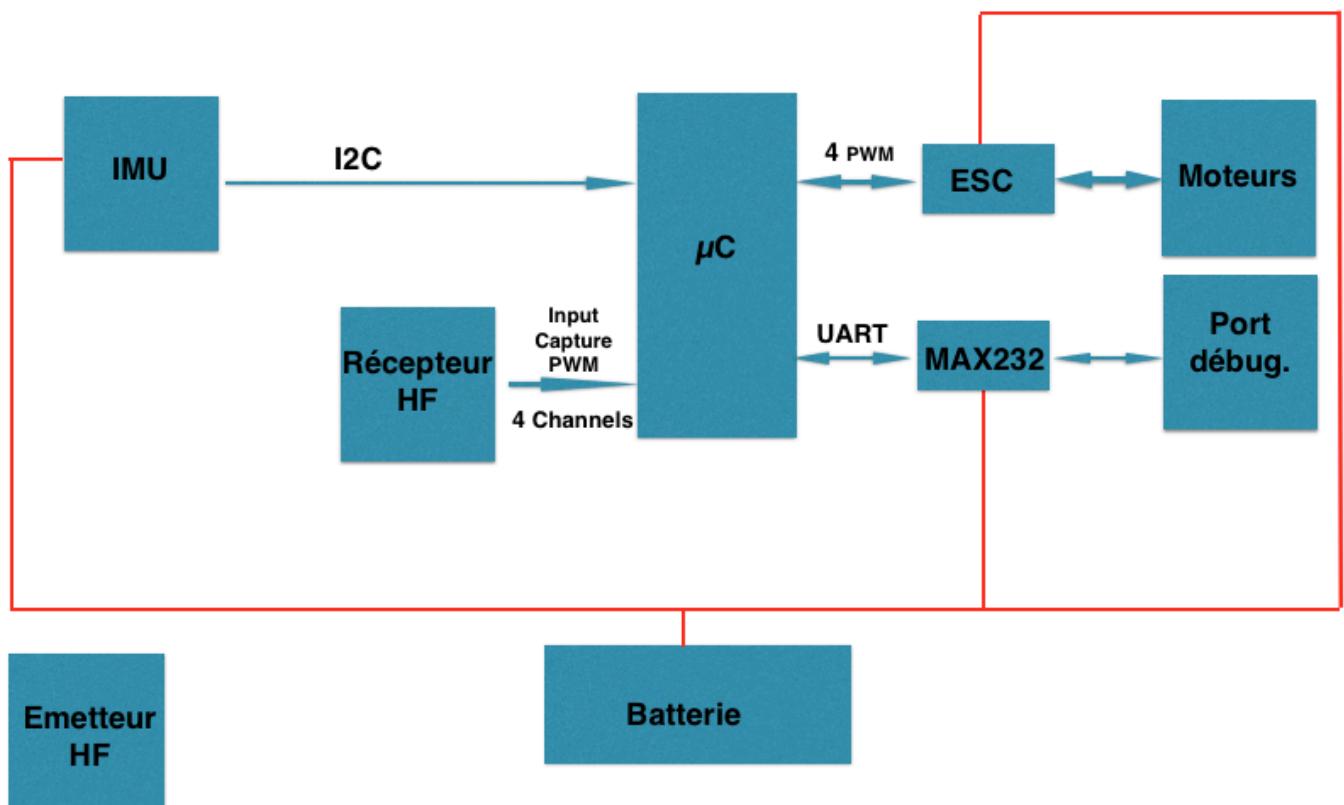
- Rotation autour de l'axe de roulis (ou roll) : un couple de moteurs fournit une puissance uniforme tandis que dans l'autre couple un des moteurs fournit plus de puissance que l'autre. Le quadrioptère avance ou recule.



- Rotation autour de l'axe de tangage (ou pitch) : Même fonctionnement que ci-dessus avec les couples de moteurs inversés. Le quadrioptère se déplace vers la gauche ou vers la droite.



Après avoir étudier le fonctionnement « mécanique » d'un quadrioptère nous avons conçu le schéma synoptique regroupant tous les composants essentiels à l'aboutissement de notre projet.



## II. Eléments constitutifs

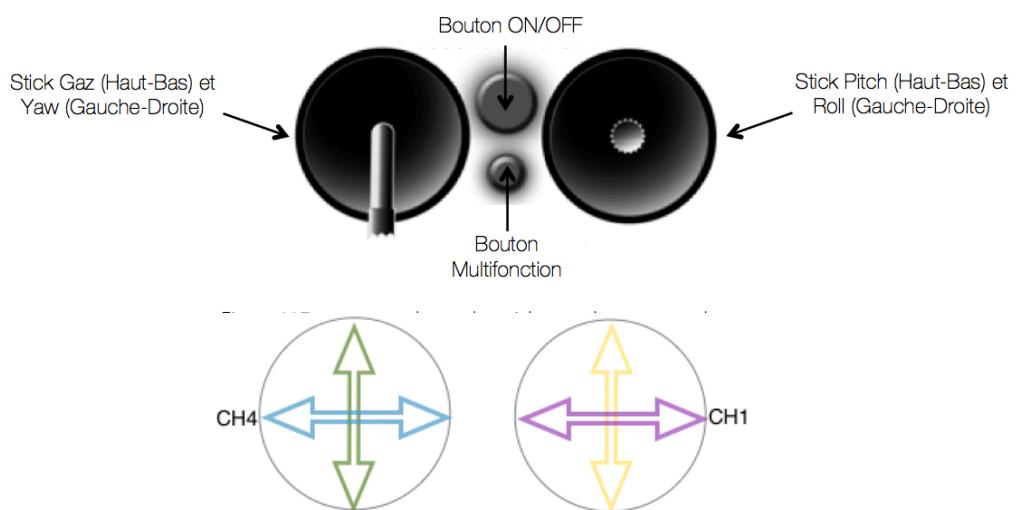
### 1. Structure – Châssis



La structure principale du quadricoptère est un cadre en fibre de carbone mesurant 55cm de long pour un poids de 280 grammes. C'est sur cette structure que seront fixés tous les composants du quadricoptère, les moteurs, les cartes de contrôle et de distribution, les quatre ESC, le récepteur haute fréquence ainsi que la batterie.

### 2. Radiocommande Hautes fréquences

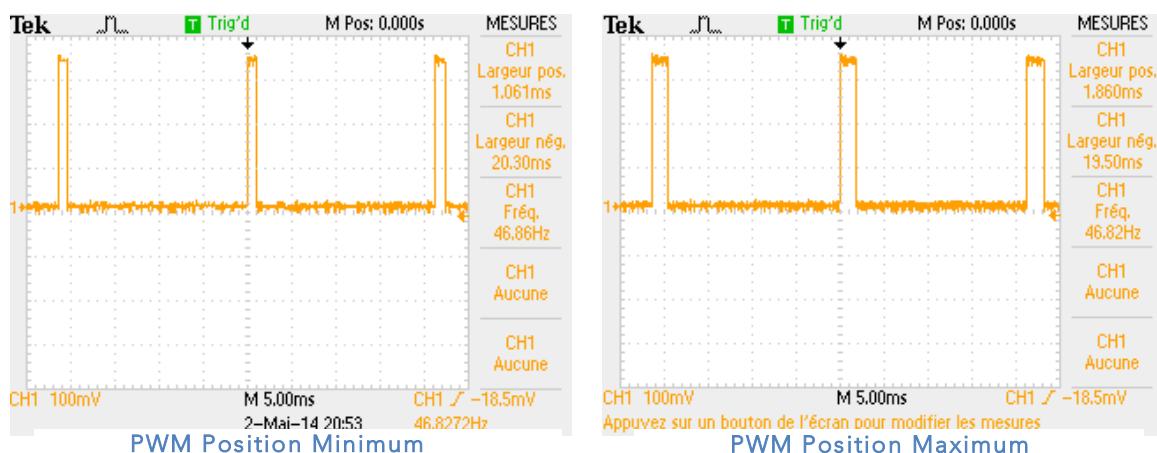
Pour commander notre drone nous utilisons une radiocommande haute fréquence pouvant émettre sur 9 channels différents mais nous n'utiliserons que 4 channels répartis sur 2 sticks possédants 2 degrés de liberté chacun comme le montre les figures ci dessous :



Les signaux émis par la radiocommande sont reçus par le récepteur haute fréquence Turnigy. Ce récepteur pèse 18 grammes, possède 8 channels différents ainsi que 2 broches pour connecter la batterie (le récepteur est alimenté par une tension de 5 volts).



Les signaux récupérés par le récepteur sont identiques, il s'agit de créneaux d'une fréquence de 46,86Hz dont le rapport cyclique varie de 1,061ms lorsque le stick est en position minimale à 1,860ms en position maximale.



On pourrait brancher les ESC et les moteurs directement à la sortie de ce récepteur mais dans notre cas nous redirigeons les 4 channels vers le pic afin de récupérer les signaux grâce à des inputs capture, puis nous générerons 4 pulse-width modulations (PWM) pour contrôler la puissance des moteurs en fonction de la position du stick de la manette.

### 3. Electronic Speed Control (ESC)

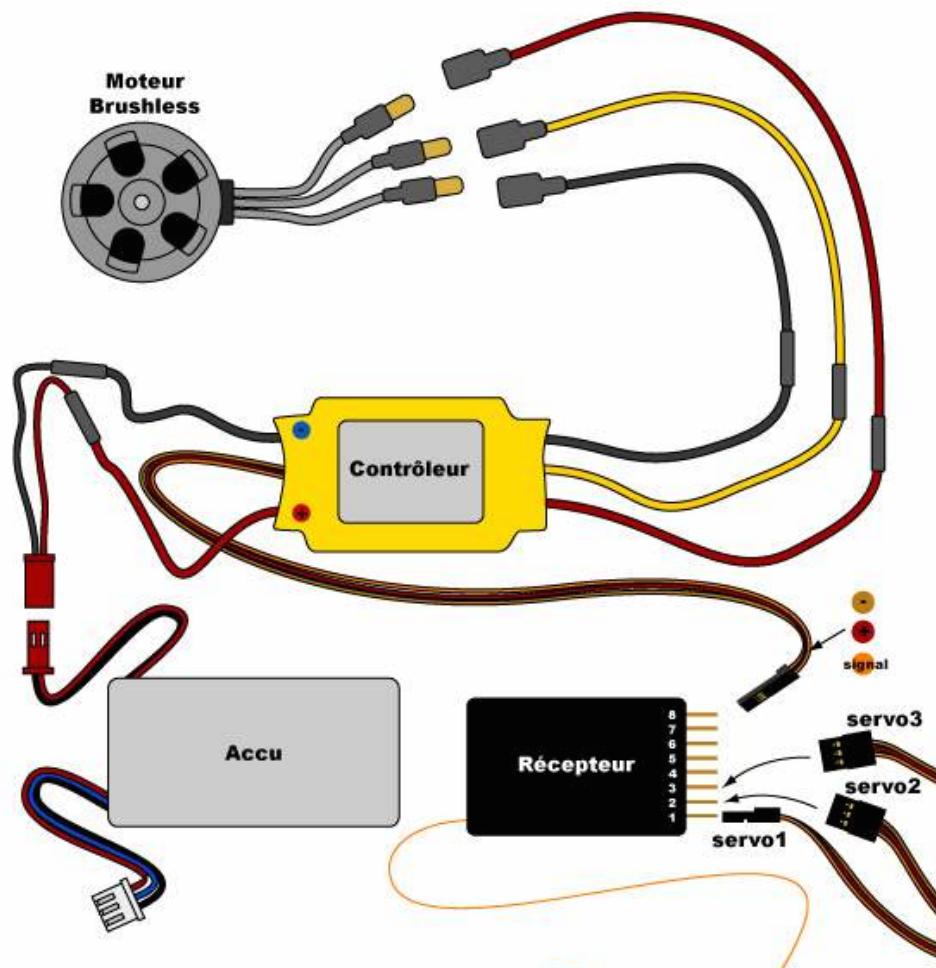
Le contrôle des moteurs est assuré par 4 ESC. Les ESC sont alimentés par la batterie avec une tension de 5 volts et sont contrôlés par les PWM produites par le pic.

L'ESC est contrôlé par une PWM et contrôle lui la vitesse du moteur en allumant puis en coupant l'apport



d'énergie du moteur. Lorsque la vitesse est maximale le moteur est toujours alimenté, lorsque la consigne de vitesse est à la moitié le moteur est alimenté la moitié du temps et non alimenté l'autre moitié du temps. Plus la vitesse de commutation est importante plus le rendement du moteur sera bon.

Le schéma de câblage que nous avons effectué est le suivant (à l'exception de la commande des ESC n'est pas branché directement sur le récepteur HF mais sur la carte de contrôle) :



#### 4. Moteurs

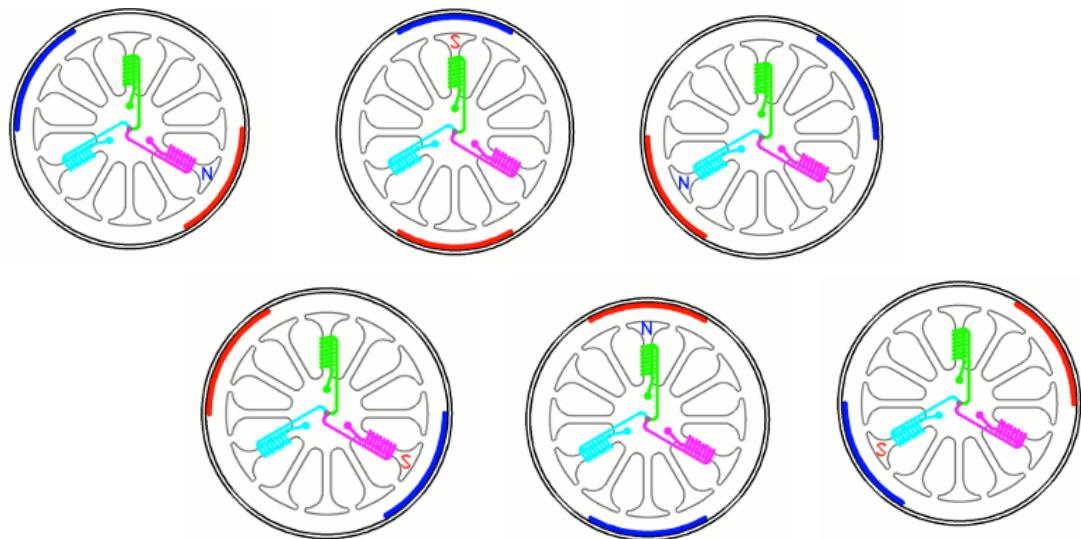
Le quadrioptère est propulsé par quatre pales qui sont entraînées par quatre moteurs synchrones brushless, c'est à dire sans balais. On a choisi ce type de moteur car il nous permet de supprimer les inconvénients du moteur à courant continu classique, les problèmes de commutations au niveau du collecteur, les problèmes dues aux balais tel que la diminution de la durée de vie, la nécessité d'un système de refroidissement ainsi que la perte de puissance dues.



## Fonctionnement simplifié d'un moteur brushless :

Dans un moteur sans balais, les bobines sont alimentées tour à tour ce qui engendre un champ magnétique tournant de même fréquence que les tensions d'alimentation. L'aimant permanent du rotor va suivre le sens du champ qui devra systématiquement rester en avance sur la position du rotor pour créer un couple. Plus l'angle entre le champ magnétique et le rotor est important plus le couple fourni par le moteur est puissant. Pour un couple maximal on aura donc un angle de  $\pi/2$ .

On peut illustrer le fonctionnement d'un moteur grâce aux schémas ci dessous :

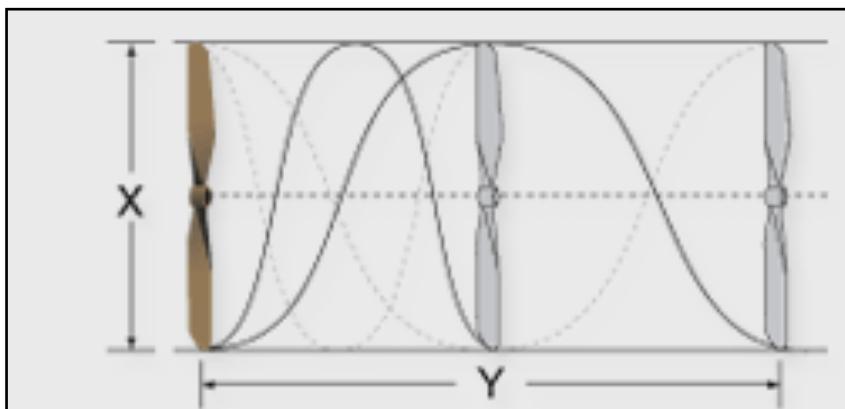


Pour notre projet nous avons choisi quatre moteurs Prop Driv de 1100rpm/v capable de fournir 189W, possédants 8 pôles et pesants 57 grammes chacun et alimentés par un courant de 20 ampères.

## 5. Pales



Pour faire voler notre drone nous avons évidemment besoin de hélices à fixer sur nos 4 moteurs. Les pales que nous avons achetées sont identiques 2 à 2, en effet elles n'ont pas la même inclinaison suivant le sens dans lequel elles seront entraînées (voir principe de vol du quadricoptère). Les hélices mesurent 10 pouces de long et ayant un pitch de 4,5 pouces par rapport à l'horizontal.



Longeur	10
Pitch	4,5

## 6. Batterie & autonomie

### Autonomie du quadricoptère à la puissance maximale :

Nous avons choisi la batterie Turnigy nano-tech 2200mA/h. C'est une batterie lithium polymère 3 cellules, et son autonomie est suffisante pour les vols que nous voulons réaliser avec notre quadricoptère.

En effet, pour s'assurer que cette batterie était suffisante pour l'utilisation que nous faisons de notre quadricoptère, soit tenter de faire voler celui-ci le plus stablement possible. Nous avons calculé l'autonomie du quadricoptère lorsqu'il fonctionne à puissance maximale.



En étudiant les datasheets nous avons relevé, lorsque cela était possible, les consommations maximales en courant des composants alimentés par la batterie :

- Microcontrôleur dsPIC33FJ128MC802 : 200mA
- IMU MPU-9150 – 9Dof : 5,15mA
- Moteur : 20A ce qui représente une consommation de 80A car il y a quatre moteurs
- Total : 80,2A

On observe logiquement que la partie puissance du quadricoptère, soit l'alimentation des moteurs (les ESC ayant leur propre batterie intégrée) représente presque entièrement la consommation totale de celui-ci. A la puissance maximale, les quatre moteurs consomment en effet 99,7% de la consommation totale du quadricoptère.

Nous avons choisi une batterie fournissant 2200mA/h en fonctionnement normal soit 2,2A/h.

$$Capacité = \frac{80,2}{2,2} = 36,45 \approx 37 \text{ fois la capacité normale de la batterie}$$

$$Autonomie = \frac{2,2}{37} = 0,059 \text{ heure soit } 0,059 \times 60 = 3,54 \text{ minutes}$$

Or  $\frac{54 \times 60}{100} = 32,4 \approx 33 \text{ secondes}$

L'autonomie du quadricoptère à la puissance maximale est donc de **3 minutes et 33 secondes** ce qui est suffisant pour tenter de faire voler celui-ci le plus stablement possible, comme nous voulons le faire dans ce projet.

De plus les 3 cellules indépendantes de la batterie rendent l'alimentation des composants plus facile. En effet chaque cellule fournie :

$$\frac{11,1}{3} = 3,7V$$

Nous avons donc connecté deux cellules au régulateur de voltage 5V ainsi nous évitons toute dissipation de chaleur inutile, au lieu d'alimenter le régulateur de tension en 11,1V et de dégager beaucoup de chaleur nous avons pu alimenter en 7,4V.

## 7. Carte de distribution

Cette carte de distribution permet d'alimenter tous les ESC avec un courant de 20 ampères à partir d'une seule batterie. Les connecteurs utilisés pour relier les ESC et la carte sont des prises male de 3,5mm. L'ensemble de la carte et des composants pèsent 27,3 grammes.

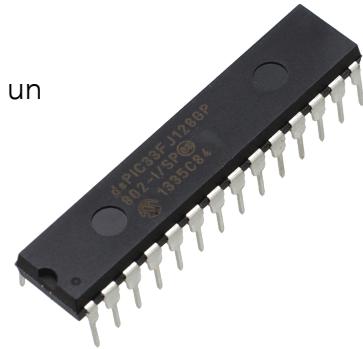


### III. Composants de la carte de contrôle

#### 1. Microcontrôleur dsPIC33FJ128MC802

Le pic que nous avons choisi est le dsPIC33FJ128MC802. Nous avons choisi ce pic car nous avions besoin de :

- 2 broches pour la liaison série pour communiquer avec un ordinateur
- 2 broches pour la liaison avec le PIC kit
- 4 inputs capture pour acquérir les signaux de nos 4 channels dans le pic
- 4 PWM pour commander les moteurs
- 2 broches pour la liaison avec l'IMU (liaison I2C)



Les critères principaux de notre sélection ont été le nombre de ports pour les PWM, le nombre de ports reprogrammables, ainsi que l'alimentation et la consommation du pic. Il devait également posséder 4 timers, une puissance de calcul et une mémoire suffisante au stockage et à l'exécution des programmes nécessaire au vol du quadricoptère.

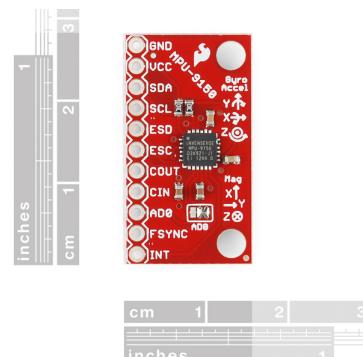
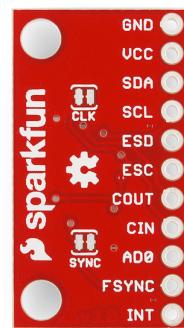
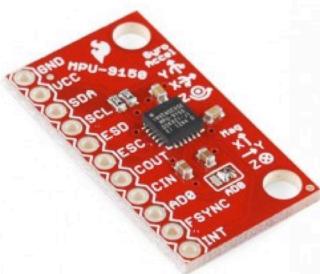
Au final notre choix c'est porté sur dsPIC33FJ128MC802 qui possédait toutes les broches nécessaires, une mémoire flash de 128KB et une capacité de calcul de 40 MIPS.

Le quartz que nous utilisons pour cadencer notre pic est un quartz de 12MHz or :

$$F_{cy} = \frac{F_{osc}}{2} = \frac{12 \cdot 10^6}{2} = 6MHz$$

La fréquence maximale d'utilisation de notre pic étant de 80 MHZ. Le Quartz est donc bien adapté à notre pic. De plus la tension d'alimentation du pic est du même ordre de grandeur que le reste des soit 3,3V.

#### 2. Inertial Measurement Unit – MPU-9150 – 9 Dof

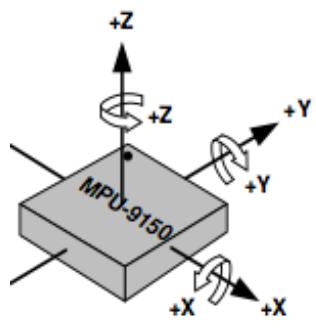


L'IMU utilisé dans notre projet est le MPU-9150 – 9Dof, nous l'avons choisi car il est petit (28,5x16mm), ce qui est pratique puisque nous souhaitons minimiser la taille de la carte de contrôle. Il est également très précis grâce aux 2 différentes puces qu'il possède.

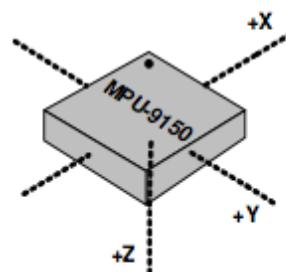
Une puce MPU-6050 qui contient 3 différents type de capteurs possédant des plages de mesures suffisantes pour notre application.

Les 3 capteurs sont :

- un gyroscope 3 axes :
  - ✓ Plage de mesure :  $\pm 250/500/1000/2000^{\circ}/s$
  - ✓ Tolérance de calibration:  $\pm 3\%$
- un accéléromètre 3 axes :
  - ✓ Plage de mesure :  $\pm 2 g - \pm 4 g - \pm 8 g - \pm 16 g$
  - ✓ Tolérance de calibration:  $\pm 3\%$
- un magnétomètre 3 axes :
  - ✓ Plage de mesure :  $\pm 1200 \mu T$
  - ✓ Sensibilité :  $0,3 \mu T/LSB$



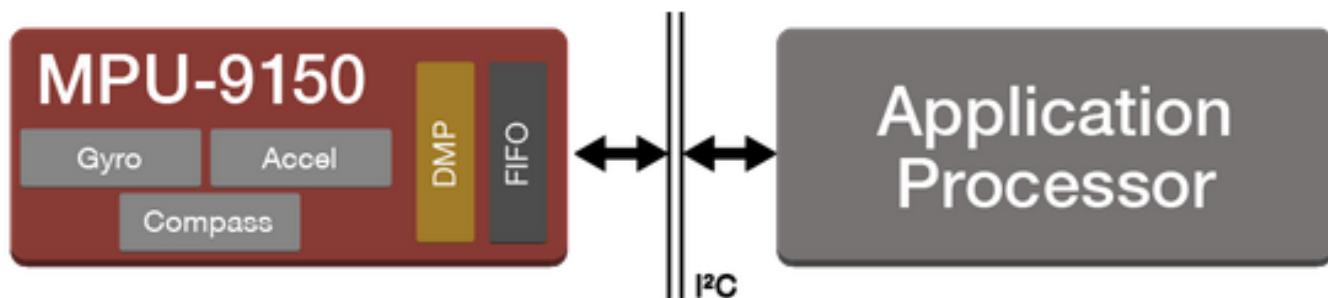
Axe de sensibilité et  
orientation de l'Accéléromètre  
et du Gyroscope



Axe de sensibilité et  
orientation du

Ces 3 capteurs combinés donne un IMU capable de détecter la position de la carte sur 9 degrés de liberté différents. Grâce au « digital motion processor » (DMP) capable de procéder à la fusion des données de l'accéléromètre et du gyroscope, ce capteur est l'un des plus utilisé dans la conception « Do It Yourself » des quadricoptères. La centrale est également munie d'un compas numérique 3 axes, le AK8975

Pour étalonner la centrale inertuelle, retourner les résultats des capteurs et les envoyer vers le PIC cet IMU possède une interface I2C 400KHz.



MPU-9150 - Diagramme du system

### 3. MAX232

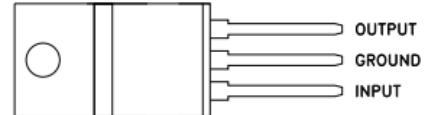
Pour pouvoir analyser les résultats de l'IMU et régler les correcteurs PID nous avons besoin d'une liaison série. Nous avons donc choisi d'utiliser un MAX 232 que nous alimenterons avec une tension de 5v et un courant de 8mA et pouvant effectuer des transmissions à un débit de 120Kbit/s ce qui sera suffisant pour retourner les résultats de nos capteurs. En plus du max 232 nous aurons également besoins d'un port série comme sur l'image ci dessous :



Ensemble Max 232 et port série

### 4. Régulateurs de tension

Nous avons vu dans la partie dédiée à la batterie que nous utilisons 2 cellules de la batterie ce qui délivre une tension de 7,4V. Or pour alimenter les composants de la carte de commande du quadrioptère nous avons besoin de 2 tensions différentes, une de 3,3V et une autre de 5V. Pour ce faire nous utilisons 2 régulateurs de voltage. Dans un premier temps nous branchons les 2 cellules de la batterie à un régulateur 5V afin d'abaisser la tension et ainsi alimenter correctement les composants nécessitant 5V, puis nous alimentons un second régulateur de tension 3,3V avec les 5V créées précédemment. Nous aurions pu brancher directement le régulateur 3,3V à la batterie mais celui si aurait dégager beaucoup de chaleur et il aurait donc fallu utiliser un système de refroidissement ce qui aurait alourdi notre quadrioptère et qui lui aurait fait perdre inutilement de l'énergie et donc diminuer son autonomie.



### 5. Autre composants

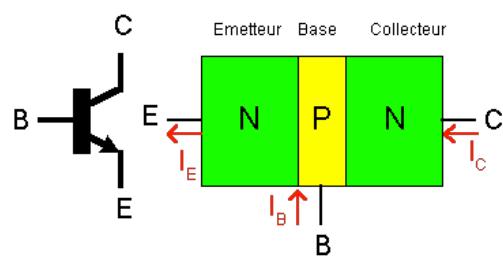
Pour réaliser la carte de commande du quadrioptère nous avons d'abord étudier les datasheets des différents composants afin de trouver les branchements minimums et les composants électroniques nécessaires à chaque composant. Nous avons ensuite été capable d'élaborer le schéma de la carte de commande.

Les branchements minimums des composants font très souvent appel à des résistances et des condensateurs. Comme par exemple dans le montage permettant à notre quartz de 12MHz d'osciller.

En effet nous utilisons des composants électroniques non parfait, par conséquent une partie de l'énergie est dissipée à chaque oscillation jusqu'à l'arrêt total des oscillations. Nous avons donc branché en parallèle le quartz avec 2 condensateurs 20 $\mu$ F afin de compenser les pertes énergétiques et d'entretenir les oscillations.

De même pour le bouton reset nous avons utilisé un bouton poussoir, une résistance et un condensateur. Lorsque le bouton poussoir est actionné il relie l'alimentation Vcc à la masse ce qui engendre une tension nul au borne de la broche MCLR et permet ainsi au pic de redémarrer car la broche MCLR est actif sur l'état bas. Le condensateur permet lui au PIC d'avoir le temps de redémarrer le temps qu'il se décharge.

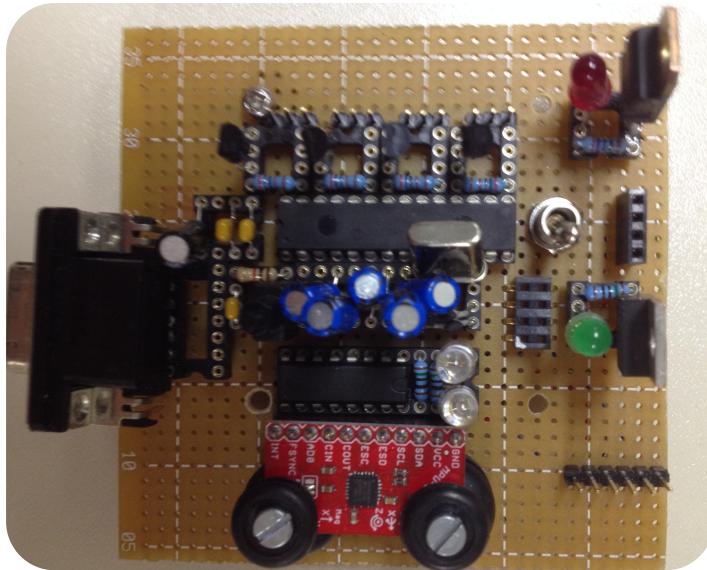
Dans la carte de contrôle nous avons également utilisé des transistors de type NPN. Un transistor peut être utilisé comme un interrupteur contrôlé en tension, en effet si aucune tension n'est appliquée entre la base et l'émetteur le transistor ne permettra pas au courant de passer (interrupteur ouvert). Un transistor permet également d'amplifier un signal ce qui nous sera utile pour alimenter nos moteurs. Un transistor est constitué par alternance de couche de 2 semi-conducteurs qui laissent plus ou moins passer le courant selon la tension.



Comme vous pouvez le voir dans la figure ci dessus un transistor est composé de 3 électrodes, le collecteur, la base et l'émetteur. Pour activer un transistor il faut au moins que la tension de la base dépasse de 0,6V celle de l'émetteur (cela correspond à la différence nécessaire pour traverser une diode).

Dans notre montage les transistors servent d'amplificateurs pour pouvoir alimenter les moteurs à courant continu. En effet au début nous avions branché directement les ESC à la sortie du PIC et on s'apercevait que le signal des PWM s'effondrait complètement car le microcontrôleur n'est pas capable de fournir un courant suffisant sur ses broches. Nous avons donc envoyé le courant sortant du PIC sur la base du transistor ce qui permet le passage d'un courant important entre le collecteur et l'émetteur, courant suffisant pour alimenter nos moteurs.

## IV. Réalisation de la carte de contrôle



### 1. Alimentation

La première étape de la réalisation de la carte de contrôle a été de câbler l'alimentation et de vérifier le bon fonctionnement des régulateurs de tension. Comme nous l'avons dit auparavant, nous avons besoin de deux tensions sur la carte pour alimenter les différents composants de celle-ci. Nous pouvons découper la carte en deux parties selon la tension d'alimentation nécessaire aux différents composants.

- Une partie « programmation et mesures » alimenté en 3,3V composée de :
  - Microcontrôleur et PicKit 3 afin de le programmer
  - IMU et MAX 232 afin d'effectuer des mesures et pouvoir les exploiter sur un terminal
  - Boutons (ON/OFF, RESET) et diodes de contrôle (ON/OFF, chargement du programme)
- Une partie « contrôle des moteurs » alimenté en 5V composée de :
  - Quatre ESC
  - Récepteur Haute Fréquence

Avant de câbler tous ces composants sur la carte nous avons vérifié le bon fonctionnement des régulateurs de tension. A l'aide d'un GBF, nous avons injecté une tension de 7,4V dans le régulateur de tension 5V puis nous avons observé à l'oscilloscope à l'aide d'une sonde que nous avions bien une tension de 5V en sortie du régulateur.

Puis nous avons injecté, toujours à l'aide d'un GBF, une tension de 5V dans le régulateur 3,3V et nous avons encore une fois observé à l'oscilloscope que nous avions bien 3,3V en sortie du régulateur.

Une fois que nous avons eu les bonnes tensions nécessaires pour alimenter les différents composants de la carte, nous avons pu commencer à câbler ceux-ci étape par étape et notamment le plus important d'entre eux, le microcontrôleur.

## 2. Tests et implémentation du microcontrôleur

Dans cette partie nous allons décrire le processus générale de notre projet, l'implémentation sera décrit dans la partie suivante.

Une fois le câblage de l'alimentation effectué, nous nous sommes attaqués au câblage du microcontrôleur qui représente le cœur de notre carte de contrôle, et des composants nécessaires à son bon fonctionnement (quartz, bouton RESET, support du PicKit 3, différents condensateurs).

Comme nous l'avons précédemment expliqué, le microcontrôleur va nous être utile car relié au récepteur haute fréquence, il reçoit les signaux PWM que celui-ci envoie, la taille des créneaux dépendant de la position des sticks sur la manette. Il va ensuite ré-envoier ces signaux vers les ESC afin de contrôler la vitesse de rotation des moteurs, la taille des créneaux influant sur la vitesse des moteurs.

Néanmoins, avant de relier le microcontrôleur avec le récepteur haute fréquence, nous avons tout d'abord vérifier que celui-ci programmait bien à l'aide de simples programmes dont nous avons pu observer la fonctionnalité sur l'oscilloscope.

Cela nous a aussi permis de vérifier le bon fonctionnement du quartz servant de signal d'horloge et du bouton RESET servant pour redémarrer le PIC.

Une fois que nous avons vérifié que le microcontrôleur programmait bien et que les composants nécessaires à son bon fonctionnement étaient fonctionnels, nous avons alors tester le programme de la PWM.

Nous avons pu observer que ce programme était fonctionnel néanmoins lorsque nous avons brancher un ESC essayer de faire tourner un moteur nous nous sommes aperçus que la PWM chutait. Comme nous l'avons expliqué ci-dessus, il a suffit de rajouter un transistor NPN entre le microcontrôleur et l'ESC pour régler ce problème. On a pu observer lors de notre premier essai que la PWM se retrouver inversé, cela est du au type de transistor utilisé (NPN).

On solutionne ce problème en envoyant sur le registre contrôlant la période (OCxRS) de la PWM en y envoyant son complément. Dès lors nous avons pu faire tourner et contrôler la vitesse d'un des moteurs pour la première fois, puis nous avons fait la même chose sur les quatre autres par réciprocité. Nous avons alors pu, pour la première fois, faire décoller légèrement du sol le quadrioptère, une première partie du cahier des charges était alors remplie.

Afin de vraiment pouvoir faire voler le quadrioptère (sans avoir besoin que le PicKit soit toujours connecté), la prochaine étape a été de relié le récepteur haute fréquence au microcontrôleur. De cette manière, nous avons pu faire décoller le quadrioptère à l'aide de la manette.

### 3. Décodage d'un signal PWM a l'aide de l'Input Capture

Quatre modules d'input capture sont présents sur notre PIC. Sa fonction est de capturer la valeur du timer associée (ici le 2 ou le 3) entre deux front montant ou descendant d'une transition, ou sur les deux. Il est également possible de capturer chaque 4<sup>ème</sup> ou 16<sup>ème</sup> front, le module agit dans ce cas comme un diviseur d'horloge.

Nous avons l'intention de l'utiliser pour générer une interruption sur les deux fronts montants et descendants. L'interruption de front montant marque le début de notre impulsion PWM, ce qui signifie le timer est remis à zéro à ce point. L'interruption de front descendant est ensuite utilisée pour capturer la valeur du timer, qui représentera la longueur de l'impulsion, divisée par un facteur d'échelle constant déterminé par la configuration de ce dernier.

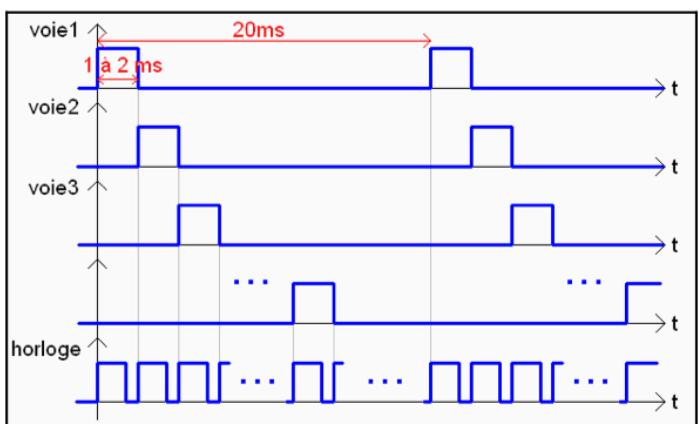
Mise en place et configuration du module d'entrée de capture :

Comme avec n'importe quel module, la première étape est de configurer les ports d'entrée et de sortie. Ensuite, nous allons configurer les modules ICx se utilisant le registre de ICxCON. On peut donc maintenant mettre en place les interruptions sur chacune des broches correspondantes au channels du récepteur HF. Ci dessous l'exemple de l'interruption correspondant au yaw :

```
void _ISR __attribute__((__interrupt__,no_auto_psv)) _IC1Interrupt(void){  
    if(PORTBbits.RB9==1){  
        restartTimer2();  
        trash=IC1BUF;  
    }  
    else if(PORTBbits.RB9==0){  
        ICD.rollInput=(float)ReadTimer2();  
        ICO.rollTarget=RollAngleRange*((ICD.rollInput-RollMid)/(RollMax-RollMin));  
    }  
    restartTimer4();  
    IFS0bits.IC1IF=0; //Clear flag  
}
```

Cette fonction est appelée à chaque fois que le drapeau d'interruption est lu comme 1. La première instruction if est vrai quand un front montant a été capturé, qui dans notre cas marque le début d'une impulsion PWM nous souhaitons mesurer. Le timer 2 redémarrage et la valeur capturée lors de l'événement de front montant est vidé de la mémoire tampon IC1 en lisant à une variable de poubelle.

Étant donné que notre signal PWM est grand, l'événement suivant à être capturé sera le front descendant du signal. Lorsque cela se produit, RB9 se lira comme 0 et la condition du else if sera vraie. On lit donc la valeur du timer 2 capturé à l'aide de la méthode ReadTimer2 présent dans la librairie timer.h. On stock cette valeur dans une structure nous permettant de la réutilisé ailleurs par la suite. Enfin, le drapeau d'interruption IC1 est effacé et la boucle est libre de se répéter.



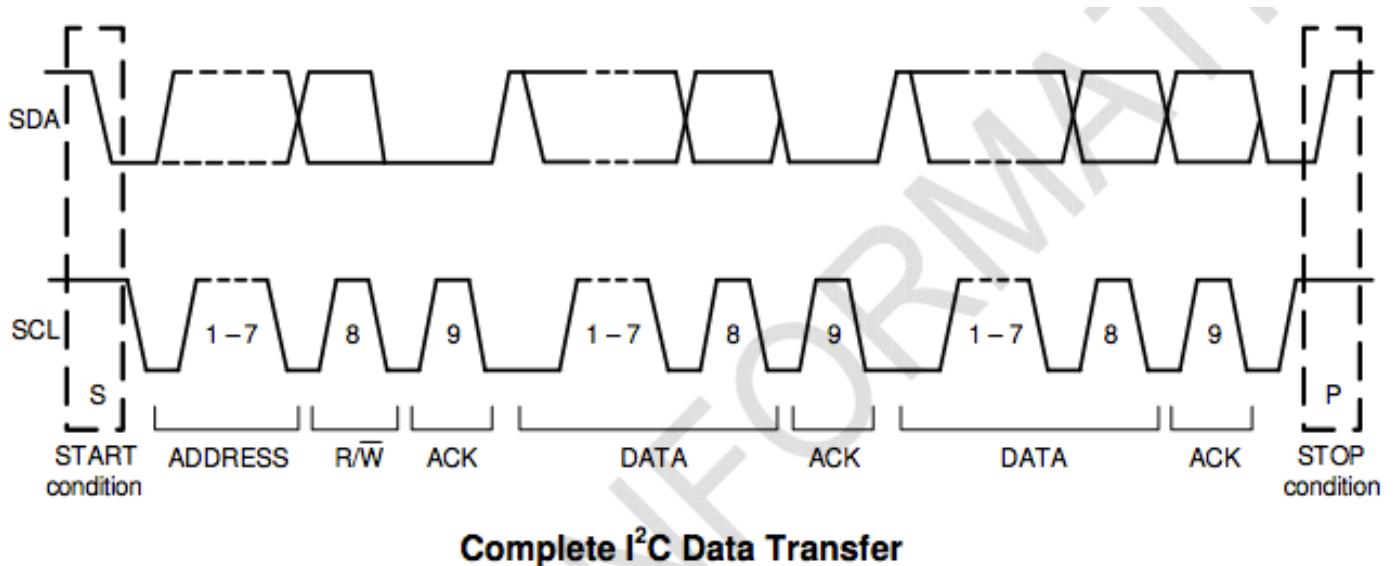
Ce code est ensuite répété pour chaque module d'entrée de capture, ce qui permet un certain nombre de canaux PWM à décoder simultanément. Maintenant, on peut se demander si nous aurions besoin d'utiliser une minuterie différente pour chaque canal PWM pour éviter tout chevauchement ? Heureusement, entre l'émetteur et le récepteur HF, les PWM doivent être transmises sur un seul canal, le multiplexage temporel est donc utilisé, ce qui signifie que chaque canal possède sa propre portion du signal. Cela signifie que lorsque le signal est dé-multiplexé, chaque signal occupe toujours sa propre période dans le signal total de 20 ms, ce qui signifie qu'il n'y a pas de chevauchement entre les canaux, ce qui permet d'utiliser un seul timer pour tous les canaux.

On remarque que la valeur capturée est convertie et stockée en un float correspondant à un ratio (compris donc entre 0 et 1), cela nous sera utile par la suite pour utiliser ces valeurs dans le calcul de la période de la PWM des moteurs. Le même traitement est appliqué aux quatre axes.

#### 4. I2C Master mode

Le bus I2C permet de faire communiquer entre eux des composants électroniques très divers grâce à seulement trois fils : Un signal de donnée (SDA), un signal d'horloge (SCL), et un signal de référence électrique (Masse).

Ici ce protocole est utilisé pour communiquer avec l'IMU. On configure donc le module I2C du dsPIC à une fréquence de 400kHz. On a ensuite développé deux méthodes, en accord avec la datasheet du capteur, permettant d'écrire et de lire un unsigned char sur le bus I2C. On peut donc accéder en lecture et en écriture aux différents registres de configuration et aux valeurs mesurées par l'IMU.



## V. Système fonctionnel

### 1. Montage du quadrioptère

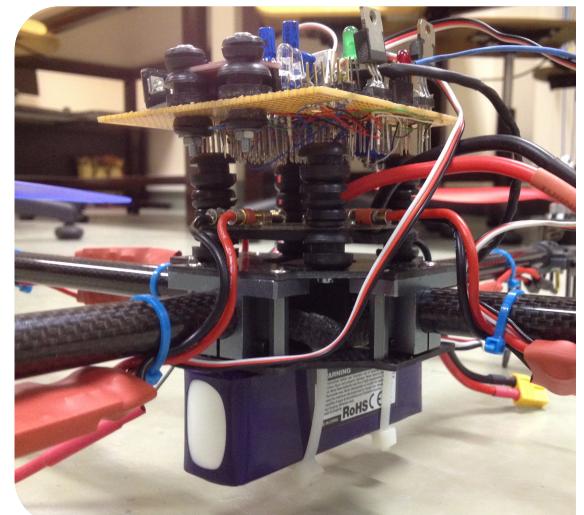


Lors de la réalisation de la carte de contrôle le critère principal était d'optimiser la taille de la carte, en effet nous voulions fixer celle-ci au centre du quadrioptère sans qu'elle dépasse (après découpage) de la plateforme prévue à cet effet.

Concernant les composants à implémenter sur la carte nous avons décidé de mettre le pic au milieu de la carte et de répartir les autres composants autour, en effet tous les composants sont branché au pic cela semblait donc être la meilleure solution.

Sur la partie haute de la carte nous avons placer toute la partie alimentation, c'est à dire le connecteur pour la batterie et les deux régulateurs de tensions accompagnés de LED rouge et verte afin de vérifier le bon fonctionnement des deux différentes tensions mais également d'instaurer un code couleur qui sera suivi au niveau du câblage.

Concernant les éléments plus volumineux tels que le batterie ou le récepteur haute fréquences nous avons placé le récepteur haute fréquence sous la carte de contrôle et la batterie en dessous du châssis en essayant de la centrer le plus possible afin de limiter le déséquilibre du quadrioptère.



### 2. Limitation des vibrations

Comme nous l'avons dis précédemment pour équilibrer le quadrioptère nous utilisons une centrale inertielle qui est fixée sur la carte de contrôle. Cependant nous ne pouvions pas fixer la centrale directement sur la carte, en effet les vibrations produites par les moteurs auraient bruitées les résultats des capteurs et aurait rendu la stabilisation plus compliquée et moins précise. Pour limiter le bruit et les vibrations nous avons donc fixé des boules de caoutchouc entre la carte et la centrale inertielle ainsi que des plaques de silicones sous les 4 moteurs.



### 3. Tests de vol

Pour tester notre carte de commande et suivre l'évolution pas à pas de notre projet nous avons testé le vol du quadrioptère en passant par 3 étapes différentes.

#### a. Sans manette ni IMU

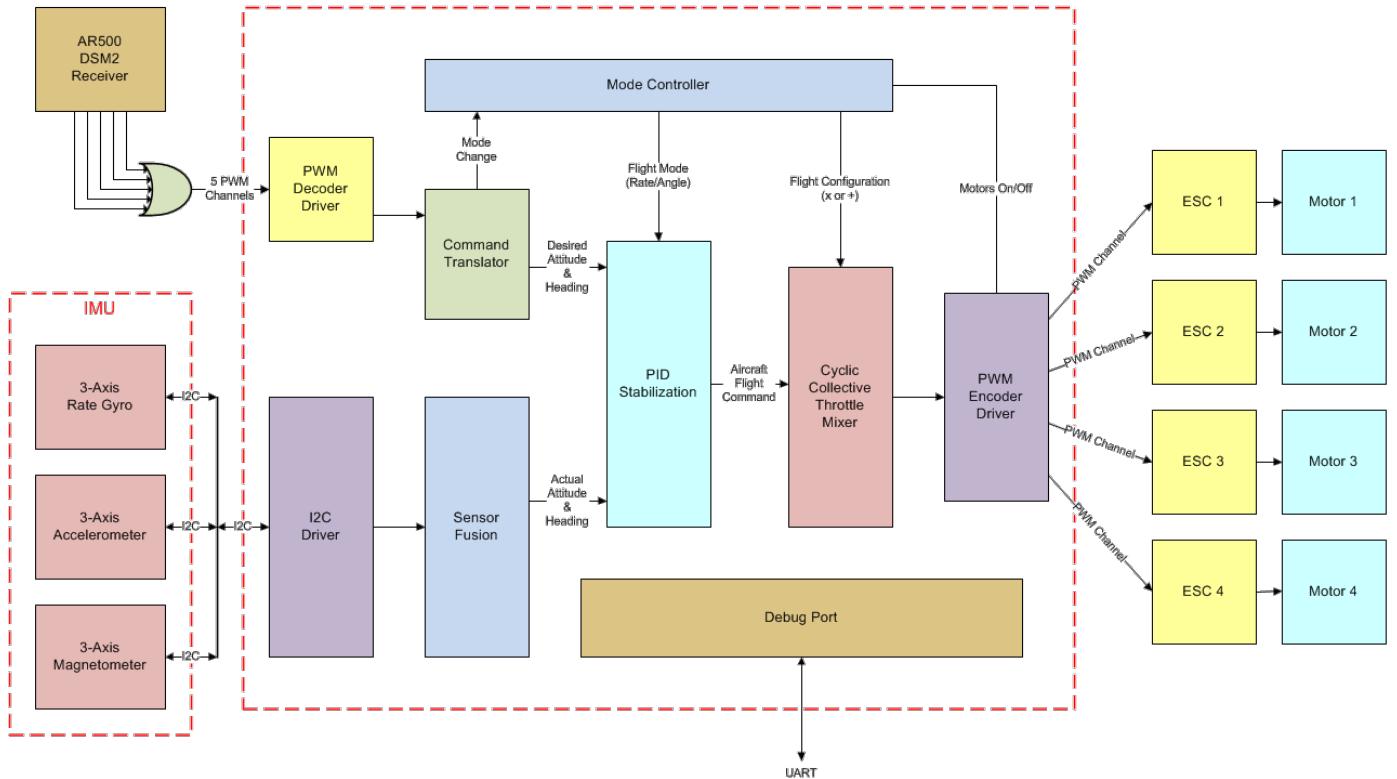
Tout d'abord nous avons testé le vol sans manette et sans IMU. En effet au début nous n'avions pas encore codé le programme permettant d'utiliser la manette et nous n'avions pas connecté le récepteur haute fréquence et l'IMU. Cependant nous souhaitions quand même tester que tout ce que nous avions réalisé jusqu'ici marchait correctement, pour cela nous avons réaliser un programme incrémentant progressivement la taille des créneaux de la PWM et donc la puissance des moteurs. Nous avons alors constaté que les moteurs s'initialisaient correctement, démarraient et accéléraient en même temps.

#### b. Avec manette, sans IMU

Nous avons ensuite codé le programme permettant de contrôler la puissance des moteurs du quadrioptère grâce à la radiocommande puis nous avons branché le récepteur haute fréquence. Nous avons alors constaté que les moteurs réagissaient bien en fonction de la position du stick de la manette et qu'il était même possible de faire décoller notre quadrioptère si il était attaché et que la puissance des moteurs étaient suffisamment importante pour qu'il se stabilise avec l'aide des ficelles.



## V. Architecture logiciel



### 1. Configurations et Horloge

Tous les systèmes séquentiels fonctionnent avec une horloge, elle a pour fonction de cadencer l'exécution des tâches, le programme. Electriquement une horloge est un signal périodique sinusoïdale ou carré. Le composant fondamental de cette fonction est le quartz utilisé pour ces caractéristiques piézoélectriques. On fait résonner le quartz à une fréquence connue qui nous donne notre signal périodique.

Le signal de période connue est donc transmis au PIC et cela s'arrête là ? Bien entendu non ! Dans notre cas, le but du jeu est d'avoir une horloge la plus rapide possible pour avoir une vitesse de calcul la plus élevée possible. Le microcontrôleur nous propose de multiplier la fréquence du quartz ce qui nous permet d'avoir un quartz de 12Mhz et une fréquence d'horloge de 40Mhz par exemple. Le revers de la médaille c'est que la consommation augmente avec la vitesse de calcul... L'organe qui permet de multiplier la fréquence du quartz est une PLL.

La fréquence maximale d'utilisation d'un dsPIC est de 80Mhz. On l'appel Tcy, c'est l'unité de temps utilisée à l'intérieur du pic. Sa valeur est de 2Thorloge.

Nous pouvons configurer plusieurs options tel que le WatchDog "chien de garde". Il vérifie le bon fonctionnement du programme et s'il remarque des problèmes il redémarre le dsPIC et notre quadricopter est explosé au sol. On le désactive donc.

Il est possible d'utiliser le quartz qui est intégré dans le dsPIC, ce n'est pas notre cas il faut donc préciser qu'on ne s'en sert pas. Nous avons donc trois fonctions à activer ou désactiver. La PLL, le WatchDog et la sélection du quartz (interne ou externe). Pour cela on édite les #pragma présent dans le fichier config.h.

## Configuration de la PLL :

La formule suivante est fournie dans la datasheet, (page 147) :

$$F_{clock} = Fin * \frac{M}{(N1 * N2)}$$

Fclock : Fréquence d'horloge finale.

Fin : Fréquence d'horloge du quartz externe.

M : Registre PLLFBD : valeur comprise entre 2 et 513.

N1 : Registre CLKDIV : bits PLLPRE : valeur comprise entre 2 et 33.

N2 : Registre CLKDIV : bits PLLPOST : choix de valeur : 2,4 ou 8.

Nous utilisons un quartz de 12Mhz, l'objectif est d'arriver à une fréquence d'horloge de 80 Mhz. Il nous faut maintenant remplir les registres pour avoir :

$$80 \text{ MHz} = 12 \text{ MHz} * \frac{40 \text{ MHz}}{(3 * 2)}$$

PLLFBD=38; car M=PLLDIV+2

PLLPRE=0; car N2=3

PLLPOST=1; car N1=2

## 2. Structure générale : utilisation des timers

Le timer est un compteur incrémenté périodiquement toutes les X périodes d'horloge, c'est ce que l'on va calculer. Il est donc impératif d'avoir configuré son horloge avant de traiter le timer. Il est aussi dans le cadre de notre utilisation étroitement lié avec une interruption. On est donc capable de compter un certain temps et d'interrompre le programme à ce moment là. L'interruption est déclenchée via un Flag timer, c'est ce qui permet de fixer une fréquence par timer.

Il va donc nous permettre d'échantillonner notre boucle. C'est à dire fixer un temps de boucle constant, nous souhaitons que notre boucle dure 1 ms. Il est donc impératif que les calculs soient faits en moins de 1 ms. Mais pourquoi rester en attente alors que les calculs sont finis ? La raison principale réside dans la simplification des calculs de dérivation et d'intégration, pour le calcul de l'angle par exemple mais c'est aussi utile pour effectuer la régulation.

Vous trouverez des renseignements à ce sujet à la page 195 de la datasheet. Nous nous servirons des 4 timers, chacun sera dédié à une fonction. Comme je vous l'ai dit plus haut l'incrémentation du timer est directement lié à la période d'horloge. Il y est donc important de noter les éléments dont on dispose :

Fréquence d'horloge = 80Mhz ;

Période d'échantillonnage = dt ;

La relation qui lie la fréquence d'échantillonnage et la fréquence d'horloge est la suivante :

$$dt = (65536 - N) * T_{CY} * Prescale$$

$$dt = \text{Période d'échantillonnage}$$

$$T_{CY} = 2 * T_{osc} = 2 * \frac{1}{F_{osc}} = 2 * \frac{1}{80 \text{ MHz}} = 1/40 \text{ us}$$

## Calcul de Prescale :

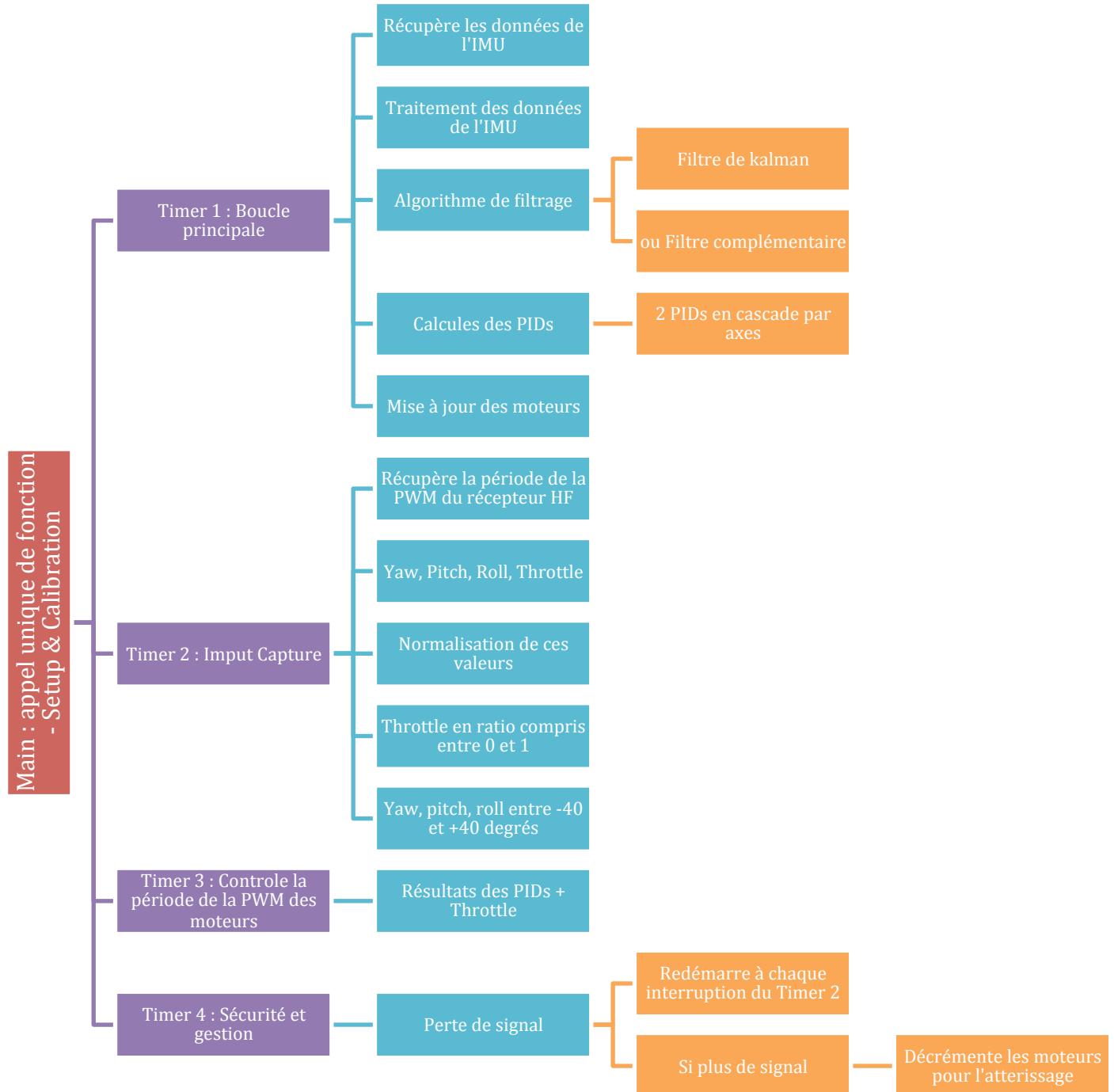
Il nous faut donc configurer N et Prescale. On commence par calculer Prescale, on prend donc N=0.

$$\text{Prescale} = \frac{dt}{(65536 * T_{CY})}$$

On choisit la valeur du dessus présente dans le registre.

## Calcul de N :

$$N = 65536 - \frac{dt}{(\text{Prescale} * T_{CY})}$$



## 2. Module input capture

L'avantage considérable du dsPIC est que l'on peut directement générer une IT à chaque front (montant et descendant). Le nombre d'instruction est donc réduit. Le deuxième avantage du DSPIC réside dans sa structure 16bits, l'accès au résultat est facilité.

On choisit le mode de capture en agissant sur les bits de commande ICM dans le registre ICxCON. De plus, en réglant les bits de commande ICM le prescaler varie (1,4 ou 16). Le choix de l'horloge est effectué via le bit ICTMR (ICxCON), ICTMR permet de sélectionner le timer employé. Le module capture comprend une mémoire à quatre niveaux, réglable via le bit de commande ICI (ICxCON).

Il est possible de sélectionner le nombre de capture avant qu'une interruption soit générée (ICxF). Nous lisons la valeur dans le registre ICxBUF.

## 3. Présentation de la fonction UART

La liaison série est un procédé de communication, elle permet à deux dispositifs de communiquer entre eux. C'est à dire d'échanger des informations, dans notre cas nous échangerons des caractères. Elle nous permet entre autre de récupérer la valeur de la variable "angle" à un moment déterminé de la boucle. Le dsPIC délivre un signal TTL qui peut être ensuite gérée par un MAX232, pour effectuer la conversion en protocole RS232 ou par un module ZigBee. Le but étant bien sûr d'obtenir ces informations sur le terminal de notre PC.

Il est important de savoir que la datasheet du dsPIC ne se limite pas à un seul fichier, il y a aussi une multitude de sections annexes qui nous renseigne de manière bien plus précise. La section 17 (DS70188), correspond à celle de l'UART. A la page 4 de celle-ci nous trouvons les registres qui nous serviront à configurer cette fonction.

UxBRG: Définition de la vitesse de transmission :

**Les formules :**

$$U1BRG = \frac{FCY}{16 * BaudRate} - 1$$

$$BaudRate = \frac{FCY}{(16 * (BRG + 1))}$$

$$error = \frac{DesirateValue - Value}{DesirateValue}$$

Avec BaudRate correspondant à une valeur au choix, nous prenons pour exemple BaudRate=57600.

**Application numérique :**

$$U1BRG = \frac{40M}{16 * 57600} - 1 = 42$$

Donc :

$$BaudRate = \frac{40M}{(16 * (42 + 1))} = 58140$$

Soit :

$$error = \frac{58140 - 57600}{57600} = 0.9\%$$

Nous pouvons conclure que U1BRGH=42 et que l'erreur est faible. Cette vitesse est donc adaptée à notre horloge.

Grace à cette fonction nous avons une manière rapide de debugger notre carte. Vous pouvez voir si après la procédure d'initialisation de la carte de contrôle lors d'un démarrage normal (la calibration de l'IMU n'est pas présente).

```
***** dsPIC Quadcopter`Controller *****  
-----  
I2C`Setup Complete  
*I2C`Read Test 0 Passed, MPU9150 Address: 0x68  
MPU9150`Setup Complete  
*Register value check passed  
PWM Setup Complete  
Timer 1 - Setup Complete  
Timer 2 - Setup Complete  
Timer 3 -`Setup Complete  
Input Capture Setup Complete  
Timer 4 -`Setup Complete|
```

## VI. Solutions pour la stabilisation

Étant donné que ces capteurs ne peuvent pas être utilisés indépendamment pour estimer précisément la dynamique en temps réel du quadricopter, ils doivent être combinés en utilisant un concept connu sous le nom d'algorithme de fusion. Le but étant de palier les faiblesses de l'un des capteurs à l'aide des forces d'un des autres capteurs.

### 1. De la donnée brute aux angles

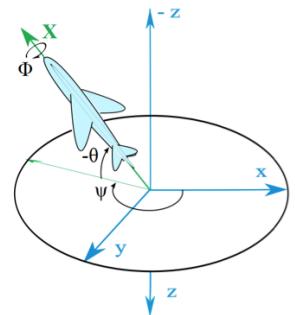
Le cœur du problème se trouve dans le passage des valeurs brutes du MPU9150 à des angles habituels. En effet les valeurs brutes ont trois problèmes :

- premièrement, elles comportent du bruit c'est-à-dire que le signal n'est pas stable,
- deuxièmement, elles n'ont pas d'unité, ce sont des valeurs qui ne correspondent à rien de concret,
- pour finir, ces données d'accélération et de rotation ne sont pas combinées en trois angles comme nous le souhaitons.

La bonne nouvelle c'est que le MPU6050 possède une puce "DMP" (Digital Motion Processing) qui permet de filtrer ces valeurs hors du PIC, donc en temps réel et de manière bien plus efficace que par un algorithme implémenté à la main. La mauvaise nouvelle c'est qu'Invensense protège jalousement le fonctionnement de cette puce au nom du secret industriel et refuse de documenter son fonctionnement. C'est ici qu'intervient Noah Zerkin, un entrepreneur en réalité augmenté qui a obtenu des dumps mémoires par retroingénierie de la puce en fonctionnement DMP à partir de démos du constructeur. C'est sur son excellent travail, intégré à I2CDevLib que se base notre code.

### 2. Angles d'Euler

La représentation en angle d'Euler utilise trois rotations d'axes individuels séparés pour représenter une rotation 3D complète. Ces trois rotations peuvent être organisées en 12 conventions différentes, 6 connus comme angles Tait-Bryan, 6 connus comme bons angles d'Euler. Ici on ne tient compte que la "convention Tait-Bryan Z-X'-Z, comme il s'agit d'une rotation autour de l'axe de lacet, une rotation autour du nouvel axe de tangage, et une rotation autour du nouvel axe de roulis. On retrouve cette convention dans l'illustration de droite, les angles  $\Psi\theta\phi$  étant respectivement le yaw, le pitch et le roll.



Ce procédé de représentation d'orientation souffre d'un nombre important d'inconvénients.

Le premier d'entre eux est simplement la possibilité d'une grande quantité de confusion due aux 12 séquences de rotation différentes possibles. Un certain nombre de singularités existent dans les représentations d'angle d'Euler. Ces changements soudains peuvent faire des ravages sur des algorithmes de fusion, nécessitant une utilisation intensive et désordonnée de déclarations conditionnelles pour tenter de corriger le signe des angles ou leur complémentarité, ce qui devient de plus en difficile à mettre en œuvre avec l'augmentation de la complexité du filtre.

Un dernier problème reste présent : si les angles d'Euler décrivent toutes les possibilités de positions angulaires, ils ne sont pas à l'abri pour autant d'un "blocage de cardan" ou Gimbal Lock. Ce problème arrive lorsque deux axes du gyroscope pointent dans la même direction, le système perd alors un degré de liberté et poursuit un mouvement imprédictible. Ce problème qui semble anodin aurait pu faire tourner court la mission Apollo 11, mais on n'en retiendra qu'une citation de Michael Collins "Que diriez-vous de m'envoyer un quatrième cardan pour Noël ?".

La solution, lorsque l'on ne croit pas au père Noël, serait d'utiliser des Quaternions, des nombres hypercomplexes fait d'une combinaison linéaire de quatre paramètres, dont trois définissent un axe et le dernier définit une rotation autour de cet axe. La manipulation de cet ensemble nous a été facilité par l'existence de librairie mise à disposition par le fabricant de l'IMU.

### 3. Filtre de Kalman

Le filtre de Kalman est un filtre à réponse impulsionnelle infinie qui estime les états d'un système dynamique à partir d'une série de mesures incomplètes ou bruitées. Il s'agit de la méthode de fusion de données de capteurs dans des applications AHRS (Attitude and Heading Reference System), car il peut être conçu pour estimer à la fois l'orientation ainsi que d'autres variables telles que les biais de capteur.

Le filtre de Kalman peut être divisée en deux étapes. L'étape de prédiction donne une estimation des variables de l'état en cours, ainsi que leurs incertitudes. Ceci peut être effectué de nombreuses fois successives, avec une prédiction de moins en moins certaine à chaque itération. L'étape de mise à jour compare ensuite ces estimations avec la mesure de bruit, en mettant à jour l'estimation en utilisant une combinaison pondérée de la prédiction et des valeurs mesure, avec plus de poids donnée à l'estimation ayant la certitude la plus élevée. Enfin, la covariance de cette estimation peut être mis à jour, ce qui représente l'exactitude de la nouvelle estimation de l'état suivant pour la comparaison des données mesurées précédemment.

Cette boucle peut être répété de manière récursive à chaque pas de temps, ne nécessitant pas l'historique des variables utilisées. L'estimation produite peut être décrite comme la solution optimale, mais seulement lorsque le système sous-jacent peut être supposé être totalement linéaire et seulement affecté par le processus et les sources de bruit de mesure avec des distributions gaussiennes.

Le filtre de Kalman étendu est la version non linéaire du filtre de Kalman régulière, qui utilise la Jacobienne des fonctions de prédiction et de mesure pour linéariser une estimation de l'état et la covariance en cours. Cela permet de régler les problèmes de type "blocage de cardan" grâce à l'utilisation des Quaternion.

De nombreux algorithmes de ce type sont disponibles sur internet. Nous n'avons pas eu le temps d'implémenter ce type de filtre. Cela sera fait par la suite.

## 4. Implantation et réglage des PIDs

Lorsque le quadrioptère confronte les trois angles d'Euler mesurés par l'IMU, comme vu dans la partie précédente, et les angles voulus envoyés par la radiocommande au sol, la différence entre ces deux données produit une erreur soudaine. Si les vitesses des moteurs sont changées en même temps que l'apparition de l'erreur, l'impulsion de commande risque : au mieux de dépasser l'angle souhaité et de revenir en arrière indéfiniment créant une instabilité, au pire de renverser immédiatement le quadrioptère. On comprend donc bien qu'il est nécessaire d'avoir un algorithme "lissant" cette transition.

Dans le cadre de notre quadrioptère, les mesures se faisant sur trois angles il est nécessaire d'utiliser trois régulateurs PID différents (dans sa configuration finale, ayant la capacité de calcul nécessaire, on cascadera deux PIDs par axe). Cet algorithme prend pour entrée la différence entre l'angle voulu et l'angle mesuré, par exemple lorsque la télécommande au sol n'envoie rien cette erreur correspond au défaut de parallélisme au sol. Des opérations mathématiques sont appliquées à ces trois erreurs et déterminent les vitesses à envoyer aux quatre moteurs.

Une fois le code du régulateur PID implémenté, il s'agit de déterminer les trois gains de chaque régulateur. Comme chaque quadrioptère a des caractéristiques particulières il n'existe pas de constantes PID universelles, cependant la détermination de ces valeurs ne relève pas non plus du hasard.

Tout d'abord, il faut noter qu'une méthode de détermination rapide, nommée "méthode de Ziegler et Nichols" existe et permet d'obtenir des  $K_i$  et  $K_d$  corrects à partir de la seule valeur de  $K_p$ . Les gains peuvent ensuite être ajustés en fonction des paramètres vus ci-dessus.

Par ailleurs, un quadrioptère (contrairement à un tricopter ou hexacopter) est à peu de choses près symétrique, les constantes des PIDs pitch et roll sont donc similaires ce qui fait gagner du temps.

Pour finir, il est important de noter que pour des raisons de sécurité évidentes l'expérimentation de ces PIDs ne se fait jamais en conditions réelles en extérieur. En accrochant solidement le quadcopter à une barre parallèle à un axe de rotation on est en mesure de bloquer la rotation des autres angles et de travailler sur un seul PID à la fois.

## Conclusion

Quatre mois se sont écoulés depuis la réunion de lancement de notre projet. Dans cet intervalle de temps qui nous a été donné, nous avons été capables de construire et de faire décoller un aéronef, plus précisément un quadrioptère.

Afin d'arriver à ce résultat final, nous avons procédé étape par étape :

- Le choix des composants présents sur la carte de contrôle
- La conception et la réalisation de la carte de contrôle
- La programmation du microcontrôleur
- Le montage du châssis du quadrioptère et l'implémentation de la carte de contrôle sur celui-ci

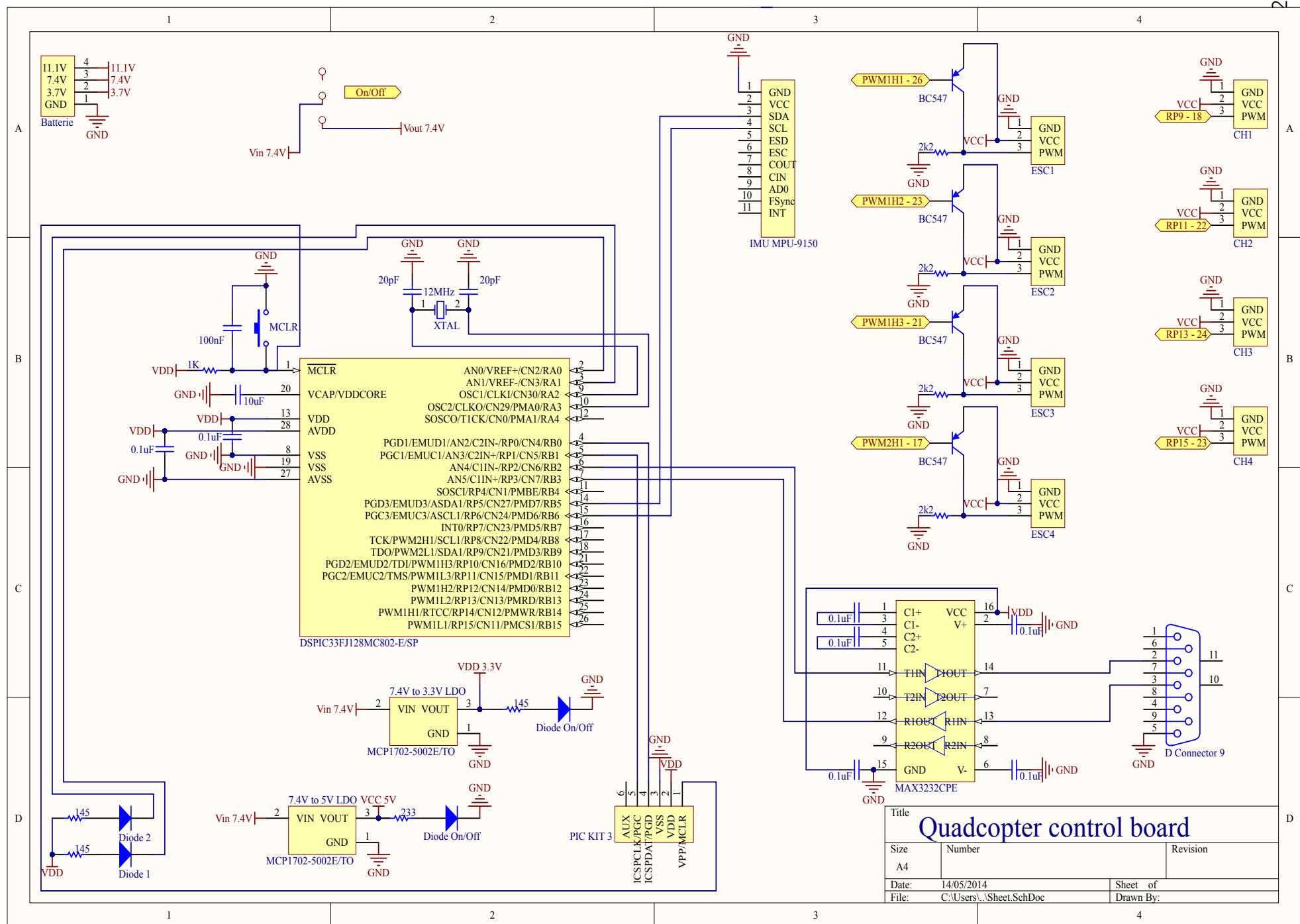
Une fois le quadrioptère capable de voler, nous avons pu passer à la deuxième partie de notre projet : la stabilisation du quadrioptère en vol. Quatre mois étant une durée assez courte pour réaliser un quadrioptère stable en vol, cette partie est encore en cours de réalisation aujourd'hui. Néanmoins, nous avons tenté d'avancer le plus possible sur ce problème.

Nous avons commencé par limiter le plus possible les vibrations du système, produites par les moteurs, à l'aide de rondelles de caoutchouc disposés autour de la carte de contrôle. Cette étape permet à la centrale inertie de renvoyer des valeurs plus précises, moins perturbées par les vibrations du système.

Ensuite, nous nous sommes attaqués au traitement des informations renvoyées par la centrale inertie. Au jour d'aujourd'hui, nous n'avons pas réussi à mettre en place un algorithme de fusion fonctionnel. Cet algorithme permet de traduire les données brutes du capteur en angles d'Euler. Ces angles sont ensuite corrigés à l'aide de correcteurs Proportionnels Intégrales Dérivés (PID) que nous avons implémentés mais étant donné que nous n'arrivons pas encore à récupérer les angles d'Euler, ces correcteurs sont pour le moment inutilisables.

Il nous reste donc encore quelques progrès à apporter à notre quadrioptère afin qu'il puisse être stable en vol. Néanmoins, nous sommes satisfaits de ce que nous avons réussi à réaliser durant ces quatre mois. Ce projet nous a permis de travailler en équipe et de mettre en pratique les enseignements dispensés à l'ESME Sudria sur un sujet qui nous passionne : le modélisme. Pour cela nous tenons tout particulièrement à remercier M. Aït Abderrahim pour la confiance qu'il nous a accordé en validant ce projet que nous savions ambitieux pour un projet de quatrième année, ainsi que pour l'aide et les conseils qu'il nous a apporté tout au long de ce projet.

# 1. Annexe 1 – Schéma de la carte de contrôle



## 2. Annexe 2 – Bill Of Material

Nom	Pins	Quantité	Prix unit	Ref. Farnell
dsPIC33FJ128MC802	28	1	6,62 €	1676256
MCP1825		1	0,83 €	1578396
MCP1702		1	0,68 €	1331487
MAX3232		1	2,73 €	9724494
BC547		4	0,09 €	2317547
IMU MP9150		1		
Quartz 20MHz		1	0,38 €	1842224
Led Rouge		2		
Led Verte		1		
Push Switch		4	0,40 €	176432
Switch ON/OFF		1	0,86 €	1634687
RS232 Connecteur Male		1		
Connecteur Male 3 pins	3	6		
Connecteur Male 6 pins	6	1		
Connecteur batterie 4 pins	4	1		
Connecteur 11 pins Fem - IMU mezzanine	11	1		

	Valeur		
Resistance	145		3
	270		4
	1k		2
	1k8		4
	2k2		4
	2k7		4
	2k		2
	10k		3
	100k		1

Condensateur	20p		2
	0.1u		10
	1u		2
	4,7u		1
	10u		2
	10n		1
	100n		1

Somme	74
-------	----

### 3. Annexe 3 – Liste des tâches

ID	Task Mode	Task Name	Duration	Start	Finish	Predecessors
1		<b>Design global</b>	<b>25 days</b>	<b>Mon 10/02/14</b>	<b>Fri 14/03/14</b>	
2		Software flow chart	5 days	Mon 10/02/14	Fri 14/02/14	9
3		Software architecture	5 days	Mon 10/02/14	Fri 14/02/14	9
4		Montage chassis + carte	1 day	Fri 14/03/14	Fri 14/03/14	12
5		Premier décollage	2 days	Wed 26/02/14	Thu 27/02/14	16
6		<b>Conception de la carte électronique</b>	<b>34 days</b>	<b>Mon 27/01/14</b>	<b>Thu 13/03/14</b>	
7		Schema synoptique	5 days	Mon 27/01/14	Fri 31/01/14	
8		Choix des composants	5 days	Mon 27/01/14	Fri 31/01/14	
9		Etude des datasheets	10 days	Mon 27/01/14	Fri 07/02/14	
10		Design de la carte : Mentor Graphic	9 days	Mon 10/02/14	Thu 20/02/14	8;9
11		Réalisation de la carte	7 days	Fri 21/02/14	Mon 03/03/14	10
12		Test de la carte	8 days	Tue 04/03/14	Thu 13/03/14	11
13		<b>Propulsion du quadcopter</b>	<b>22 days</b>	<b>Mon 27/01/14</b>	<b>Tue 25/02/14</b>	
14		Fonction PWM pour un moteur	8 days	Mon 27/01/14	Wed 05/02/14	
15		Test de la PWM sur un moteur	7 days	Thu 06/02/14	Fri 14/02/14	14
16		Extention PWM aux 4 moteurs	7 days	Mon 17/02/14	Tue 25/02/14	14;15
17		<b>IMU</b>	<b>27 days</b>	<b>Mon 27/01/14</b>	<b>Tue 04/03/14</b>	
18		Programmatation/Intégration IMU	20 days	Mon 27/01/14	Fri 21/02/14	
19		Test & Validation IMU	7 days	Mon 24/02/14	Tue 04/03/14	18
20		<b>Commande HF</b>	<b>35 days</b>	<b>Fri 14/03/14</b>	<b>Thu 01/05/14</b>	
21		Intégration du récepteur HF	8 days	Fri 14/03/14	Tue 25/03/14	12
22		Programmation des commandes	23 days	Wed 26/03/14	Fri 25/04/14	21
23		Tests	4 days	Mon 28/04/14	Thu 01/05/14	22
24		<b>Stabilisation</b>	<b>35 days</b>	<b>Mon 10/02/14</b>	<b>Fri 28/03/14</b>	
25		Modèle Simulink et choix des correcteurs	10 days	Mon 10/02/14	Fri 21/02/14	
26		PID Roulis	10 days	Mon 24/02/14	Fri 07/03/14	25
27		PID Tanguage	10 days	Mon 24/02/14	Fri 07/03/14	25
28		PID Lacets	10 days	Mon 24/02/14	Fri 07/03/14	25
29		Programmation des PID	10 days	Wed 26/02/14	Tue 11/03/14	16
30		Tests & corrections	15 days	Mon 10/03/14	Fri 28/03/14	26;27;28