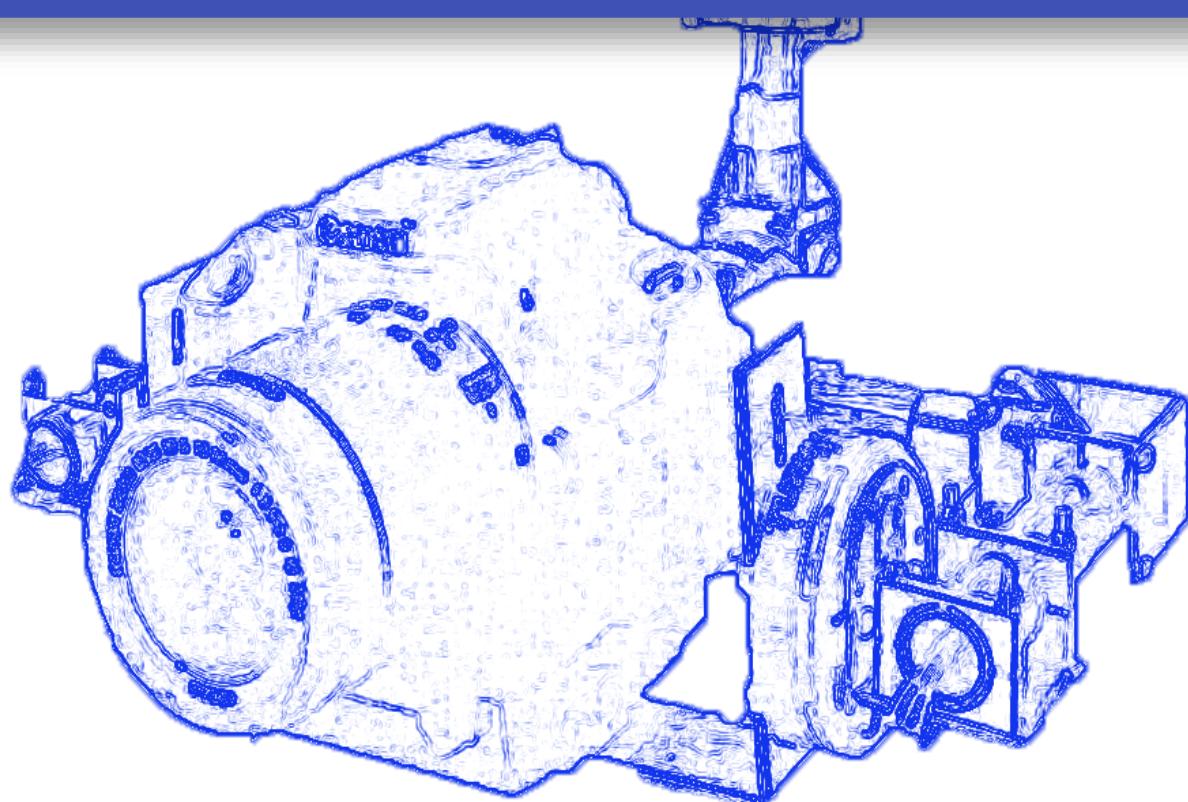


# Stabilisation de Caméra

---

Les enjeux de la réalisation d'un stabilisateur



# Table des Matières

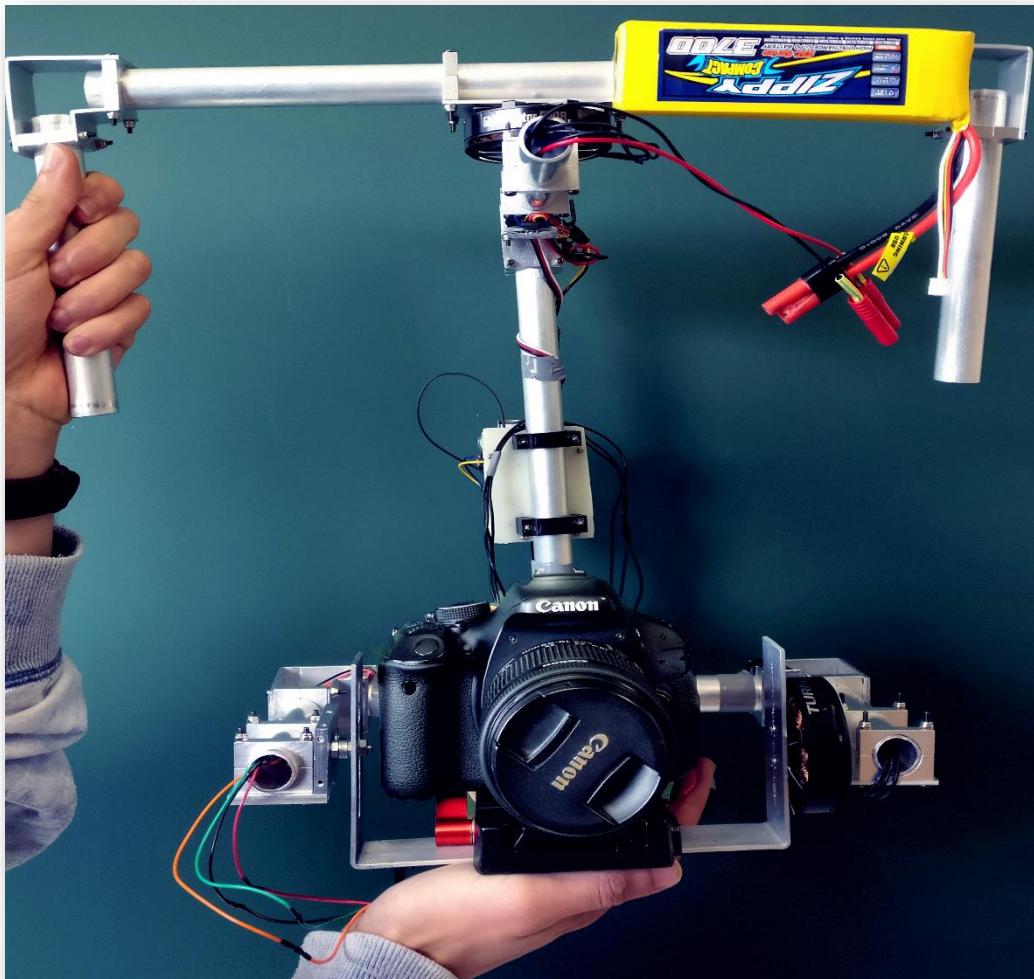
## Table des matières

|  |    |
|--|----|
| Résumé du TIPE .....   | 4  |
| I. Introduction .....  | 5  |
| 1. Utilisation dans la vie courante .....  | 5  |
| 2. Principe de fonctionnement.....   | 5  |
| II. Usinage et construction.....   | 7  |
| 1. Une construction exigeante capable de tenir une caméra de 1,2 kg.....                     | 7  |
| 2. Assemblage.....   | 7  |
| 3. Un design modulable permettant de régler le centre de masse par rapport à la caméra ..... | 11 |
| 4. Des composants de haute qualité .....   | 12 |
| 5. Schéma Électronique .....   | 15 |
| III. Fonctionnement de l'Accéléromètre : .....   | 17 |
| 1. Fonctionnement théorique .....  | 17 |
| 2. Communication avec le MPU6050.....  | 18 |
| a. Configuration initiale.....   | 18 |
| b. Lecture des données .....   | 19 |
| c. Calibration.....  | 19 |
| 3. Calcul des angles $\phi$ et $\theta$ à partir de l'accéléromètre.....                     | 20 |
| 4. Implémentation dans le code :.....  | 22 |
| IV. Stabilisation.....   | 23 |
| 1. Asservissement en vitesse angulaire et position .....                                     | 23 |
| 2. Implémentation d'un algorithme PID.....   | 23 |
| 3. Étallonnage des valeurs du PID.....   | 25 |
| V. Le filtre de Kalman.....  | 26 |
| 1. Présentation .....  | 26 |
| 2. Le principe de fonctionnement du filtre .....   | 27 |
| 3. Étude sur plusieurs variables. ....   | 29 |
| 4. Interprétation générale .....   | 32 |
| 5. Implémentation dans le code.....  | 33 |
| 6. Obtenir les valeurs à partir de notre librairie .....                                     | 33 |
| 7. Précision concernant le calcul de dt .....  | 34 |

|       |  |    |
|-------|--|----|
| VI.   | Le gimbal lock .....   | 34 |
| 1.    | Modèle géométrique des angles d'Euler .....                          | 34 |
| 2.    | Le problème du gimbal lock.....                                      | 35 |
| 3.    | La correction du problème.....                                       | 37 |
| VII.  | Commande des moteurs.....  | 37 |
| VIII. | Pour aller plus loin.....  | 39 |
| 1.    | Mode de stabilisation sur deux axes uniquement .....                 | 39 |
| 2.    | Différents taux de réactivité : méthode d'émulation de réponse ..... | 39 |
| 3.    | Contrôle à distance du stabilisateur.....                            | 39 |
|       | Conclusion .....   | 40 |
|       | Abstract .....   | 41 |
|       | Immenses Remerciements .....   | 42 |
|       | Annexe : Sitographie .....   | 43 |
|       | Annexe : Code.....   | 46 |
|       | Annexe : Code Kalman.....  | 56 |

## Résumé du TIPE

Notre TIPE devait répondre à la problématique suivante : optimalité. Nous avons donc choisi de construire un stabilisateur de caméra, afin d'optimiser les prises d'images.



Afin de stabiliser la caméra, notre système détecte les perturbations à l'aide de différents capteurs. Les informations collectées sont envoyées vers une Arduino. Un programme traite ces données avec un correcteurs PID et un filtre de Kalman. Ce filtre nécessite une longue étude théorique qui sera détaillée au cours de notre rapport, voici son principe : il calcule une estimation de la position théorique du système, prend en argument la mesure de la position effective donnée par les capteurs, renvoie une position angulaire précise sans bruit avec une erreur minime. Ensuite cette position est renvoyée dans le programme principal, qui la renvoie dans le correcteur PID. Celui-ci se charge d'envoyer les consignes aux moteurs permettant de garder la caméra stable.

Notre structure a aussi été étudiée minutieusement. Il a fallu déterminer la longueur de la moindre barre au millimètre près afin de s'assurer de la stabilité du système, et la construction devait aussi être faite avec rigueur. Une fois celle-ci terminée, un calibrage a été nécessaire (rendu

possible grâce à plusieurs pièces mobiles) afin d'équilibrer parfaitement notre structure même lorsque le programme ne tourne pas, nous disposons donc déjà d'une stabilisateur mécanique fonctionnel.

## I. Introduction

Lors de ce TIPE nous avons réalisé un stabilisateur de caméra. Mais quelle est l'utilité d'un tel objet ?

### 1. Utilisation dans la vie courante

Quiconque s'est déjà rendu dans une salle de cinéma sait que lorsqu'un film est tourné, un soin particulier est donné à la qualité des images. Il s'agit certes des retouches faites en « post production », mais pas seulement : cette recherche de qualité commence lors de la prise des plans avec les caméras, et le fait que l'image ne tremble pas est bien le minimum. Pour ce faire on dispose plusieurs moyens, en voici trois principaux :

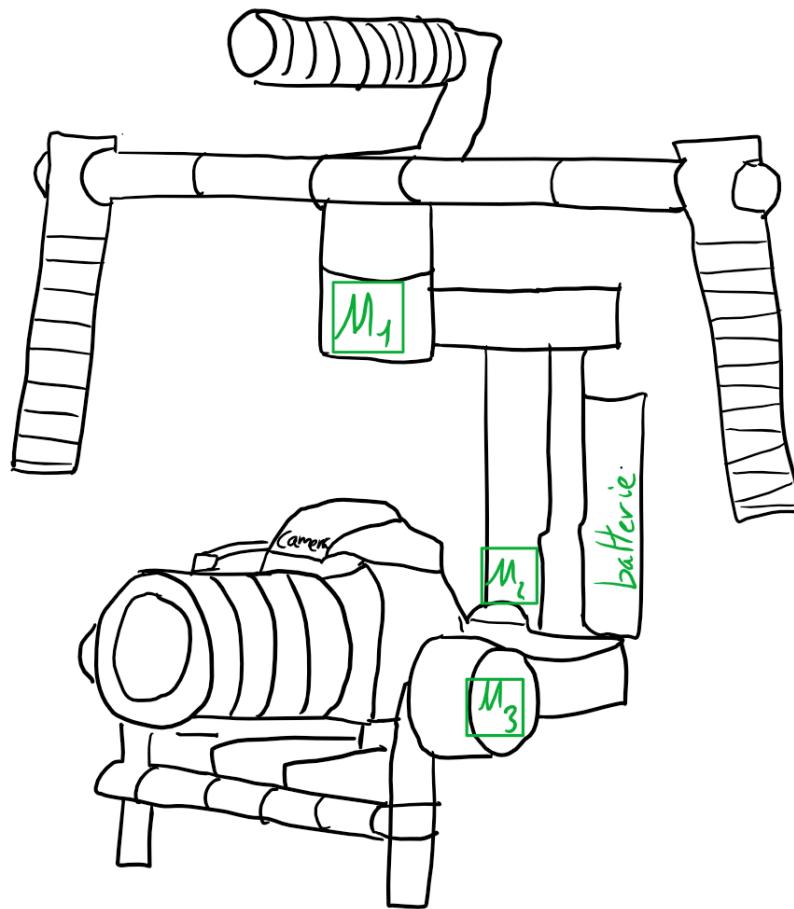
- La stabilisation numérique, qui consiste à prendre une série d'image et ne garder que celles qui sont nettes, il s'agit du moyen le plus économique et le moins efficace
- La stabilisation optique, qui fait appel à une lentille supplémentaire. Celle-ci est contrôlée par des électroaimants, pilotés par des accéléromètres. Cette lentille compense les mouvements du photographe ou du caméraman. Un mode dit « actif » existe sur les systèmes très performants, qui compense aussi les mouvements du sujet. Il s'agit d'un mode de stabilisation très efficace, mais chère, et elle a un défaut : l'ajout d'une lentille ajoute un intermédiaire supplémentaire lors du passage de la lumière qui peut nuire à la qualité de l'image
- La stabilisation mécanique, qui consiste à déplacer le capteur lui-même en réaction aux mouvements du système afin de les contrecarrer. C'est ce principe que nous développerons dans le cadre de ce TIPE.

### 2. Principe de fonctionnement.

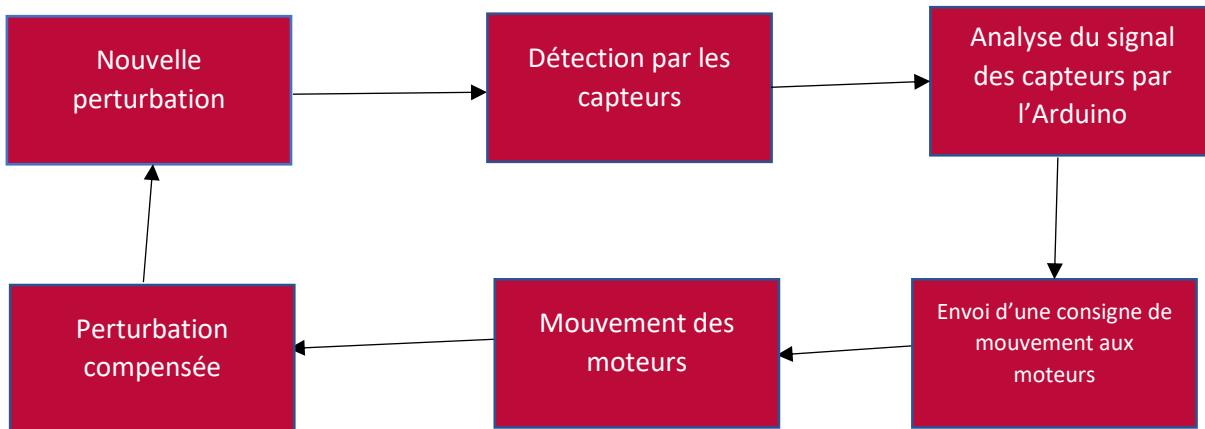
Notre système devra répondre aux contraintes suivantes

- Compenser tout mouvement non désiré de la caméra
- Eviter tout à coup lors de la compensation d'une perturbation
- Ne pas déplacer la caméra en l'absence de perturbation
- Être facilement transportable
- S'adapter à différents modèles de caméra
- Avoir un coût concurrentiel

Pour ce faire nous avons utilisé un support de tubes creux en aluminium. La caméra est placée de manière à ce qu'elle soit en équilibre et stable en l'absence de perturbations. Il y a 3 moteurs liant différents tubes, cela permet de se déplacer selon les 3 axes du monde en 3 dimensions dans lequel nous vivons (roulis, lacet et tangage).



La détection de perturbation se fait par l'intermédiaire d'un gyroscope et d'un accéléromètre. Les signaux qu'ils envoient sont analysés par une Arduino, qui commande ensuite les moteurs de manière à compenser le mouvement. Voici un schéma superficiel du fonctionnement global de notre système.



## II. Usinage et construction

### 1. Une construction exigente capable de tenir une caméra de 1,2 kg.



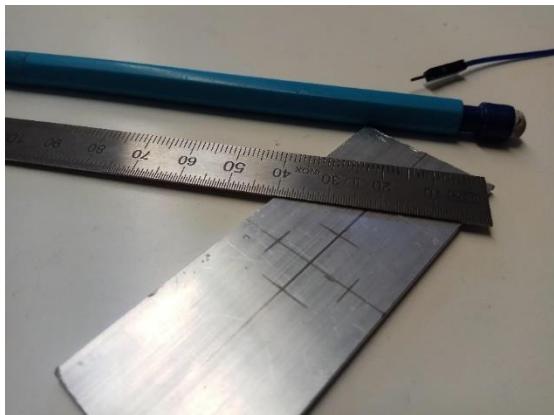
Nous avons énoncé, parmi les contraintes à respecter, que notre système devra être aisément transportable. En effet on souhaite stabiliser une caméra à main, d'un poids inférieur ou égal à 1,2kg. Pour ce faire nous avons utilisé des tubes d'aluminium d'une largeur de 2cm et d'1mm d'épaisseur.

| Nom            | Nbr | Taille |
|----------------|-----|--------|
| Barre haut 1   | 1   | 37     |
| Poignée        | 3   | 15     |
| Barre 2        | 1   | 31     |
| Barre 3        | 3   | 21     |
| Petite barre 1 | 2   | 9,27   |
| Petite barre 2 | 1   | 8,31   |

Pour définir ces valeurs nous avons étudié les caractéristiques des porte-caméra déjà existant. L'aluminium nous est apparu comme un bon choix car ce matériau offre un bon équilibre entre robustesse et légèreté. De plus il est facile à découper ce qui nous a permis d'obtenir rapidement et facilement des tubes de tailles variées et découpés très précisément. Après les découpes nous disposons des tubes suivants

### 2. Assemblage

Nous avons commencé par découper les tubes selon les tailles énoncées précédemment. Une grande précision était essentielle car toute erreur aurait nui gravement à l'équilibre de notre système. Nous considérons nos découpes précises au millimètre près.

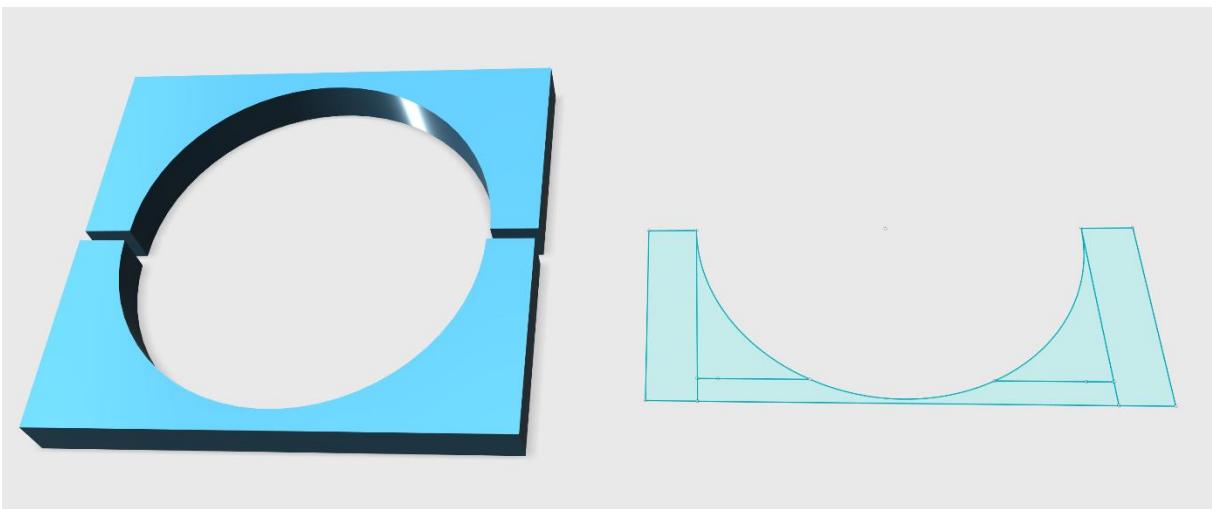


Il nous fallait ensuite trouver un moyen de fixer les différents composants sur les barres, et les barres les unes sur les autres. Il est plus commode de les attacher sur des surfaces planes nous avons donc utilisé des plaquettes d'aluminium.

Sur ces plaquettes d'aluminium, il a été nécessaire de percer plus de 62 trous.

Ici on peut voir une fixation de moteur, ici la tolérance n'est que de  $\frac{1}{4}$  de mm. Un long travail de mesure et d'ajustement a donc été nécessaire pour que l'ensemble de ces composants s'assemblent correctement.

Pour fixer les plaquettes aux tubes nous avions d'abord décidé de fabriquer nos propres pièces à l'imprimante 3D.



Voici le design de ces pièces, il est très simple et plutôt efficace. Nous avions tout d'abord imprimé deux à trois exemplaires. Ils étaient convenables mais n'étaient pas optimaux. Le plastique était suffisamment solide, mais finalement nous avons trouvé sur internet des pièces en aluminium qui correspondaient exactement au design précédemment créé.



Vissées sur une plaquette, ces pièces permettent d' « attraper » un barre en l'enserrant entre les deux attaches. Cela est très commode car on peut facilement

- Faire coulisser la pièce le long de la barre
  - Tourner la pièce autour de la barre
- Ce qui facilite l'équilibrage de notre système, ce qui est capital.

Certaines de ces plaques ont subit un traitement plutôt particulier. En effet 12 plaques avec un angle de 90 degrés étaient nécessaires.



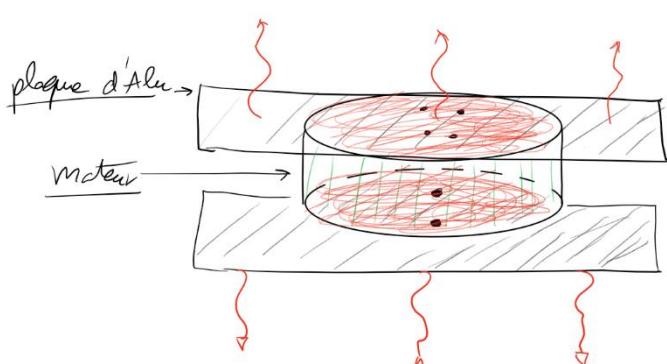
Nous les avons réalisés en créant après mesure une ligne de faille sur l'aluminium à l'aide d'une lime. Nous avons ensuite placé les plaques dans un étau pour les tenir fermement. Nous avons ensuite chauffé un peu l'aluminium à l'aide d'un fer à souder le long de cette ligne de faille. Et nous avons ainsi plié les 12 plaques nécessaires à un angle de 90°.



Une fois les plaques fixées aux barres, nous avons fixé l'électronique aux plaquettes. En utilisant des vis pour les moteurs. Notons ici que le moteur est en contact direct avec l'aluminium. Ce détail semble normal, mais il est tout autant capital. En effet nous avons relevé de hautes températures lors d'usage intensif des moteurs. Celle-ci montait au-dessus des 50°C.

Ce problème est résolu grâce aux propriétés de conductivité thermique de l'aluminium.

En effet, on peut le constater directement sur l'appareil, les tubes et plaques en aluminium proches des moteurs sont souvent plus chauds.



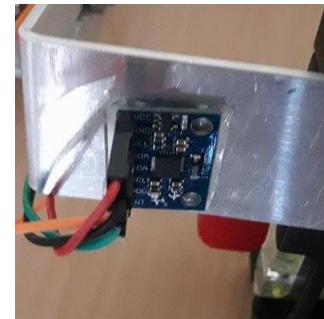
La structure agit ainsi comme dissipateur thermique. Ceci est essentiel. Lors de notre précédent TIPE, un incendie de moteur a eu lieu. Il a menacé grandement notre sécurité et il a détruit une partie de notre électronique.



Pour fixer la batterie nous avons utilisé du velcro car on doit pouvoir la retirer facilement, et que le fait qu'elle ait un peu de jeu n'était en rien un problème. Celle-ci est placée complètement en haut de la structure pour ne pas rajouter d'effort supplémentaire sur les moteurs.

Nous avons utilisé du scotch double face caoutchouteux pour fixer les capteurs car il s'agissait d'un composant :

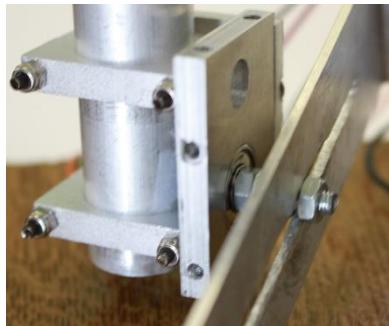
- Léger
- Devant rester stable
- Ne devrait pas subir l'ensemble des micro-vibrations du moteur.



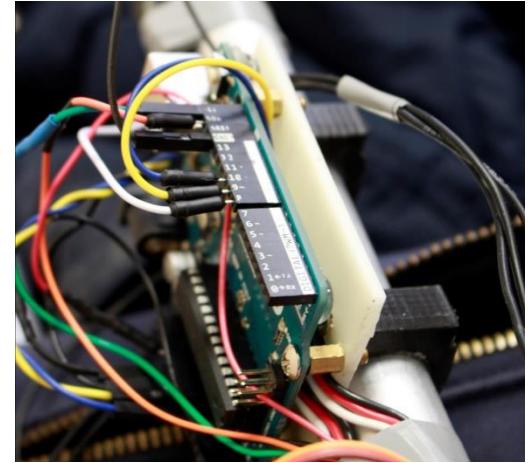
Le support caméra, doit permettre d'attacher fermement la caméra, mais aussi de la faire effectuer un mouvement de translation d'avant en arrière. C'est donc une liaison glissière. Il est composé de deux pièces un plateau, et le support. On note aussi un niveau qui permettra un ajustement final parfait. Il est fixé sur une pièce en U.



Pour cette pièce en U, nous avons tout d'abord découper la pièce à plat. Nous avons découpé à la scie sauteuse un rail de chaque coté pour ajuster précisément la hauteur. Ensuite nous avons plié à 90° au niveau des traits rouge.



Enfin nous avons attaché ce U avec un roulement à bille. On peut voir ici comment faire coulisser l'axe principal dans le rail.



La carte Arduino devait être fixée solidement et à un endroit qui faciliterait les branchements. Nous l'avons donc disposée au niveau de la barre verticale, soit au centre du dispositif.

### 3. Un design modulable permettant de régler le centre de masse par rapport à la caméra

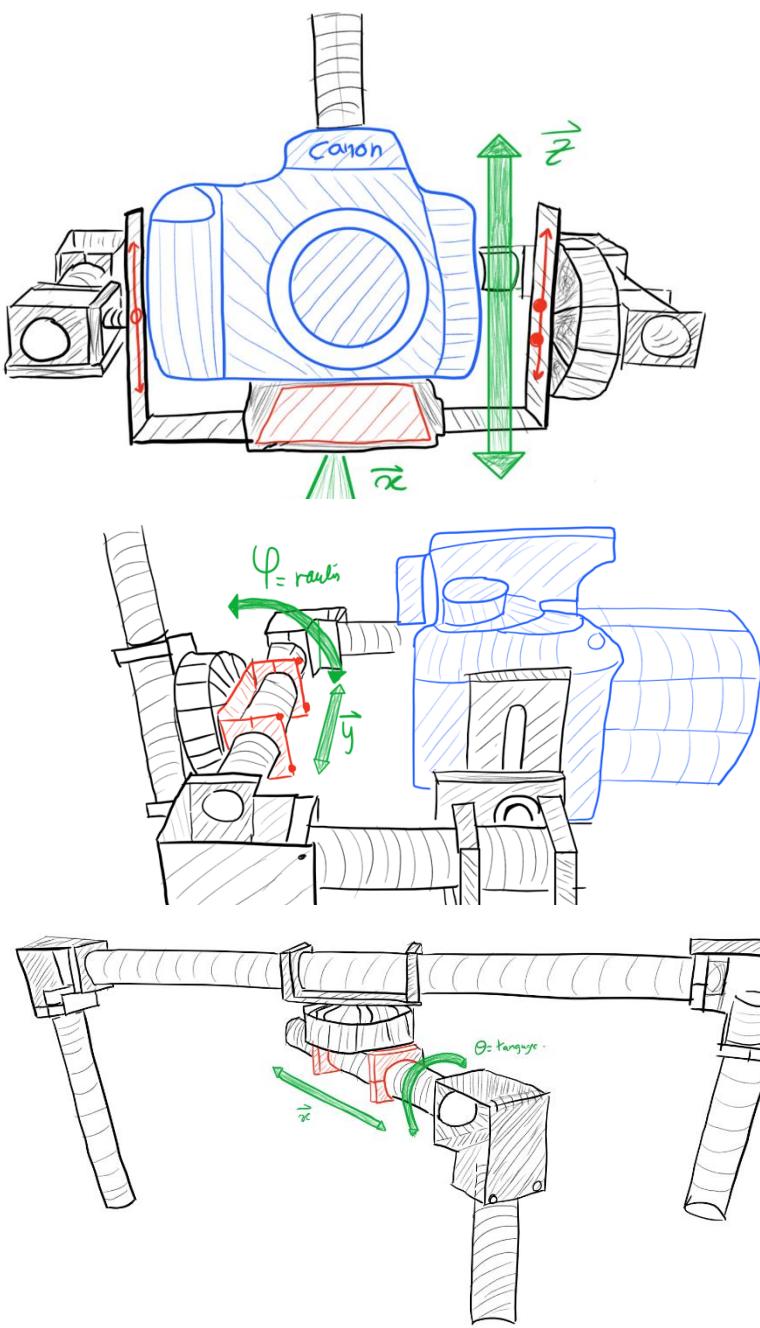
Nous traitons ici le 5eme point de notre liste de contraintes : avoir un système s'adaptant à différents modèles de caméra.

- Quel lien avec un centre de masse ?

Afin d'obtenir une stabilisation optimale, il faut limiter autant que possible l'action des moteurs, qui ne sont pas parfaits. Imaginons un instant que pour avoir une situation d'équilibre (caméra parallèle au sol et perpendiculairement à l'axe des épaules de l'utilisateur) les moteurs doivent être actifs. Dans ce cas, la stabilisation n'étant pas parfaite, la caméra tremblera en permanence. De plus, notre stabilisateur fonctionnant sur batterie, il y aurait un gaspillage d'énergie diminuant la durée d'autonomie. On en conclut donc que la caméra doit pouvoir être placée de manière à ce qu'elle soit en situation d'équilibre en absence de toute perturbation.

- Comment régler la position de ce centre de masse pour une caméra quelconque ?

La caméra est placée sur un plateau, en « bas » du stabilisateur. Pour obtenir l'équilibre recherché on doit pouvoir déplacer la caméra relativement au stabilisateur, ce que l'on fait en autorisant des mouvements sur tous les angles et tous les axes. (obtenu en utilisant des attaches mobiles entre différentes barres). Cela permet de régler selon les trois axes nécessaires. Autrement dit, il n'existe aucun moment sur ces axes.



1- On commence par le moteur effectuant le mouvement roulis de la camera (roll).

On règle l'axe  $\vec{x}$  puis  $\vec{z}$  ici pour n'avoir aucun moment sur l'axe du moteur . La caméra doit rester immobile peu importe l'angle imposé.

2- On règle à présent le moteur correspondant à l'axe du tanguage ( pitch ).

On déplace alors la structure selon l'axe  $\vec{y}$  puis selon l'angle  $\varphi$  et de même la caméra doit rester immobile peu importe l'angle imposé

3- Pour terminer le troisième axe celui du lacet (yaw).

On commence par translatser selon l'axe  $\vec{x}$  pour obtenir un alignement du centre de masse et de l'axe du lacet.

Et enfin on termine avec un rotation selon l'angle  $\theta$ .

#### 4. Des composants de haute qualité

Une fois que nous disposons d'un structure opérationnelle, nous devons choisir les composants électroniques que nous utiliserons.

| Qt | Composant  | Lien  | Specs  | Prix  |
|----|--|---|--|-------|
| 3  | Moteur Gimbal<br> | <a href="https://hobbyking.com/en_us/turnigy-hd-5208-brushless-gimbal-motor-bldc.html">https://hobbyking.com/en_us/turnigy-hd-5208-brushless-gimbal-motor-bldc.html</a> | Poles: 12N14P<br>KV(RPM/V): 31<br>Resistance: 10 ohm<br>Weight: 180g<br>Wire: 0.27mm<br>Termination method: Star | 38.40 |

|   |  |  |   |  |
|---|--|--|---|--|
|   |  |  | Top holes center to center: <b>M2.5 15mm</b><br>Bottom mount: <b>M3 25mm x 25mm</b>   |  |
| 3 | ESC (contrôleur électronique de vitesse) | <br><a href="http://www.rovertec.com/products-bgmc.html">http://www.rovertec.com/products-bgmc.html</a>   | Current Draw: 30A<br>Continuous<br>Voltage Range: 2-4s<br>Lipoly<br>BEC: 0.5A Linear<br>Input Freq: 1KHz<br>Firmware: afro_nfet.hex   | 49.9<br>9  |
| 1 | Programmeur à ESC                        | <br><a href="https://hobbyking.com/en_us/afro-esc-usb-programming-tool.html">https://hobbyking.com/en_us/afro-esc-usb-programming-tool.html</a>   |   | 7.95   |
| 1 | Arduino Méga                             | <br><a href="https://store.arduino.cc/product/GBX00067">https://store.arduino.cc/product/GBX00067</a>   | The Mega 2560 is based on the <b>ATmega2560</b> . It has 54 digital input/output pins a 16 MHz crystal oscillator,  | 0<br>Déjà  |
| 1 | MPU6050 Accéléromètre et gyroscope       | <br><a href="http://www.hobbyking.com/hobbyking/store/_26860_Kingduino_MPU6050_3_Axis_Gyroscope_3_Axis_Accelerometer.html">http://www.hobbyking.com/hobbyking/store/_26860_Kingduino_MPU6050_3_Axis_Gyroscope_3_Axis_Accelerometer.html</a>                         | Arduino MPU6050 3-Axis Gyroscope 3-Axis Accelerometer   | 0<br>Déjà  |
| 1 | Batterie                                 | <br><a href="http://www.hobbyking.com/hobbyking/store/_36195_ZIPPY_Compact_3700mAh_3S_25C_Lipo_Pack_EU_warehouse.html">http://www.hobbyking.com/hobbyking/store/_36195_ZIPPY_Compact_3700mAh_3S_25C_Lipo_Pack_EU_warehouse.html</a>                                 | <b>Spec.</b><br><b>Capacity:</b> 3700mAh<br><b>Voltage:</b> 3S1P / 3 Cell / 11.1V<br><b>Discharge:</b> 25C Constant / 35C Burst<br><b>Weight:</b> 264g (including wire, plug & case)<br><b>Dimensions:</b> 146x19x43 mm | 0<br>Déjà  |
| 3 | Tube Aluminium                           | <br><a href="http://www.castorama.fr/store/Tube-rond-aluminium-brut--16-mm-1-m-PRDm175689.html?navAction=jump&amp;isSearchResult=true">http://www.castorama.fr/store/Tube-rond-aluminium-brut--16-mm-1-m-PRDm175689.html?navAction=jump&amp;isSearchResult=true</a> |   | 5,25   |
| 6 | Attaches Alu                             |  | <a href="https://hobbyking.com/en_us/silver-anodized-cnc-6mm-aluminum-tube-clamp-20mm-diameter.html">https://hobbyking.com/en_us/silver-anodized-cnc-6mm-aluminum-tube-clamp-20mm-diameter.html</a>                     | Weight: 34g<br>Size: 30 x 24 x 6mm<br>Tube Diameter: 20mm<br>Mounting: 25mm center to center |

|   |                  |   |   |    |
|---|------------------|---|---|----|
|   |                  |   |   |    |
| 7 | Attache camera : | <a href="https://www.amazon.fr/Andoer-Release-Adaptateur-Compatible-Manfrotto/dp/B016D6AM16/ref=sr_1_fkmr1_1?ie=UTF8&amp;qid=1494484949&amp;sr=8-1-fkmr1&amp;keywords=quick+release+camera">https://www.amazon.fr/Andoer-Release-Adaptateur-Compatible-Manfrotto/dp/B016D6AM16/ref=sr_1_fkmr1_1?ie=UTF8&amp;qid=1494484949&amp;sr=8-1-fkmr1&amp;keywords=quick+release+camera</a> | Peut être utilisé avec trépied, rail, stabilisateur et différents appareils, pour la photographie | 15 |

- Les moteurs

Notre choix s'est porté sur 3 moteurs brushless, un pour chaque axe (caractéristiques en annexe). Une différence majeure entre un moteur à courant continu et un brushless est que le premier est bruyant, ce qui aurait posé un problème évident si l'on considère la faible distance séparant les moteurs et le micro de la caméra. Les moteurs que nous avons choisis disposaient d'un couple suffisant pour notre système tout en ayant un poids assez faible pour ne pas trop déséquilibrer la structure. En effet, la façon dont ils sont placés en fait un facteur modifiant la position du centre de masse de la structure

- Les ESC (Electronic Speed Controllers)

Il est nécessaire de faire passer le signal sortant de l'arduino par des ESC, afin d'envoyer aux moteurs brushless le signal triphasé nécessaire à leur fonctionnement. De plus, les moteurs doivent pouvoir être commandés dans les deux sens de rotation (horaire et anti horaire) afin de compenser tout type de perturbation. Dans un premier temps nous avions l'intention d'utiliser des ESC standards, et de les flasher.

Flasher un composant électronique consiste à remplacer le firmware (programme pré implanté) de ce composant par un programme nous arrangeant. Cela s'est avéré plus difficile que prévu : nous avions l'intention de nous fonder sur un travail de rétro-ingénierie (la rétro ingénierie est le fait de déduire la programmation d'un composant en analysant son fonctionnement) réalisé par un ingénieur allemand Olliver W. connu sur Internet pour son expertise dans le domaine. Mais il s'est avéré qu'il n'avait pas réussi à obtenir des résultats concluants, nous avons donc dû rechercher un ESC nous permettant de commander nos moteurs dans les deux sens de rotation. Nous avons eu de la chance, un tel ESC venait d'être mis au point par une startup américaine : notre TIPE n'aurait donc pas pu être réalisé il y a quelques années.

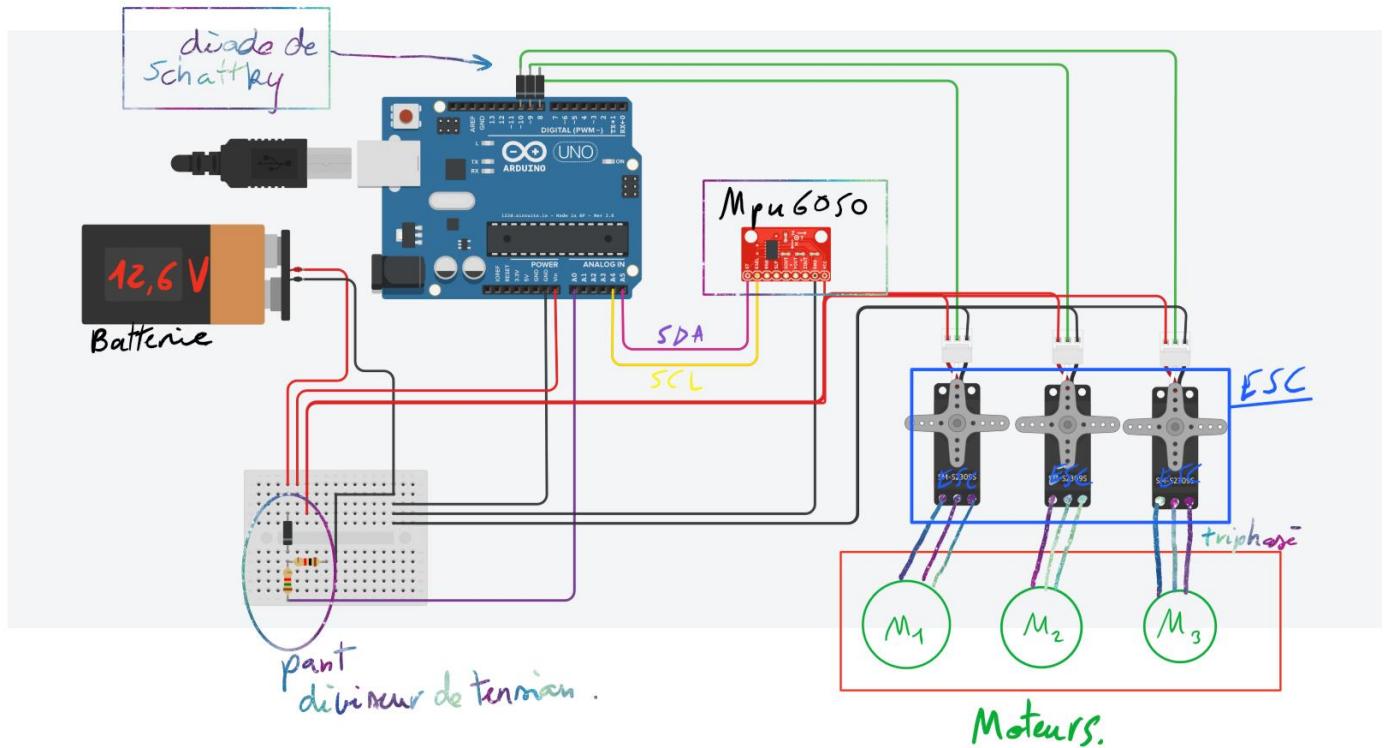
- Le processeur

Notre système n'exigeant pas de grandes capacités de calculs, la puissance d'une carte arduino uno est suffisante. Le prix bas et la simplicité d'utilisation de cette carte ont fini de nous convaincre qu'il s'agissait du composant à choisir

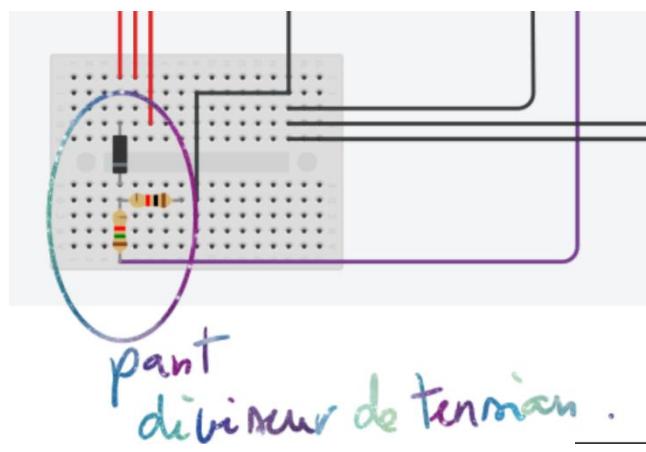
- Les capteurs

Nous avons utilisé deux capteurs différents afin de collecter les données nécessaires au fonctionnement de notre système.

## 5. Schéma Électronique



La partie électronique de ce TIPE est plus ou moins simple, n'étant pas le sujet principal. Cependant certaines subtilités sont à relever.



Une diode est branchée en dérivation sur la cathode de la batterie, puis celle-ci est relié à un simple diviseur de tension couplé à deux résistances l'une de  $1,5\text{ K}\Omega$  et l'autre de  $1\text{ K}\Omega$ .

Cela permet de lire la tension de la batterie. La carte Arduino ne peut lire sur ses ports analogiques qu'une tension de 0 à 5 V, ainsi brancher la batterie de 12V directement sur la carte celle-ci risquerait de la griller. On divise

donc la tension par deux et l'on met les valeurs de résistance nécessaires pour ne jamais dépasser 5 V sur le pin de l'Arduino.

On multiplie ensuite la valeur de la tension pour avoir le voltage original. La diode empêche quant à elle le courant d'aller dans le sens inverse.

Plus la batterie est utilisée, plus la tension de la batterie diminue puisqu'elle s'épuise. Nous devons donc nous assurer que l'usager soit informé de l'épuisement de celle-ci afin de limiter les risques d'accident (une batterie LIPO déchargée chauffe, s'abime et peut exploser)

Il faut tout d'abord calculer le voltage de la batterie. Celle-ci délivre une tension de 12.6V lorsqu'elle est pleine (La batterie est reliée au Pin A0). La valeur retournée par `analogRead()` est de 1023 pour 12.6V, nous devons donc faire le rapport  $1260/1023 = 1.2317$  et multiplier ce ratio à la valeur d'`analogRead()` pour ramener la valeur de celle-ci à la bonne échelle. Il ne faut pas oublier que la diode fait perdre 0.7V aussi.

Le calcul de la tension de la batterie est donc :

```
tension_batterie= (analogRead(0) + 65) * 1.2317;
```

Le connecteur de la batterie étant relié au pin analogique 0 de l'Arduino.

Il faut maintenant prévenir le pilote lorsque la tension de la batterie est trop faible grâce à un buzzer :

```
if(tension_batterie < 1050 && tension_batterie > 600) digitalWrite(12, HIGH);
```

Pour faire fonctionner le buzzer, il suffit de lui envoyer une tension de 5 V. Ainsi, si la tension de la batterie est comprise entre 10,5 V et 9 V, c'est-à-dire lorsque la batterie est faible on effectue un buzz.

On note aussi la présence de 3 diodes de Schottky . Elles ont pour rôle de limiter le courant que prennent les ESCs. En effet nous avons remarqué un défaut sur ces ESCs. Ils peuvent fonctionner directement en utilisant le canal PWM de l'Arduino. Ceci résulte en une quantité de courant trop important pour l'Arduino. Chaque pin étant limité à 0.40mA. Ces ESCs étaient largement au-dessus et empêcher l'Arduino de fonctionner. Nous avons informé le constructeur, il nous a confirmé le problème et a affirmé que sur la prochaine version il n'y aura plus un tel problème.

### III. Fonctionnement de l'Accéléromètre :

#### 1. Fonctionnement théorique

Un accéléromètre est un capteur capable de mesurer une accélération. Sur terre ce capteur mesure donc en permanence l'accélération de la pesanteur  $\vec{g}$ . Cette mesure s'effectue selon les trois axes du repère cartésien ( $\vec{x}, \vec{y}, \vec{z}$ ).

Son fonctionnement est plus ou moins complexe. Il est composé d'une nappe de plaque sensitive.

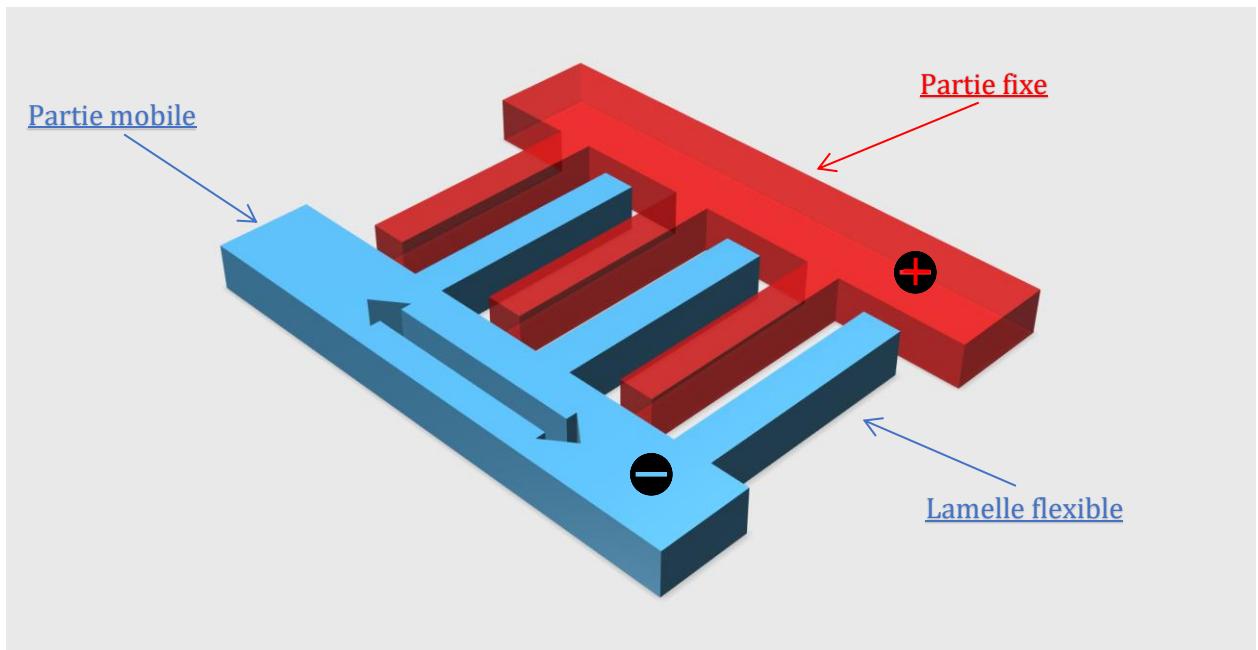


Figure 1 - Représentation simplifié de l'accéléromètre.

Le circuit à l'intérieur de l'accéléromètre est composé de petites lamelles flexibles. Ces lamelles en se translatant agissent comme des petits condensateurs variables. Le reste du circuit ne fait que convertir ces différences de capacités proportionnellement en accélération subit. Un schéma simplifié serait alors (Figure 2) :

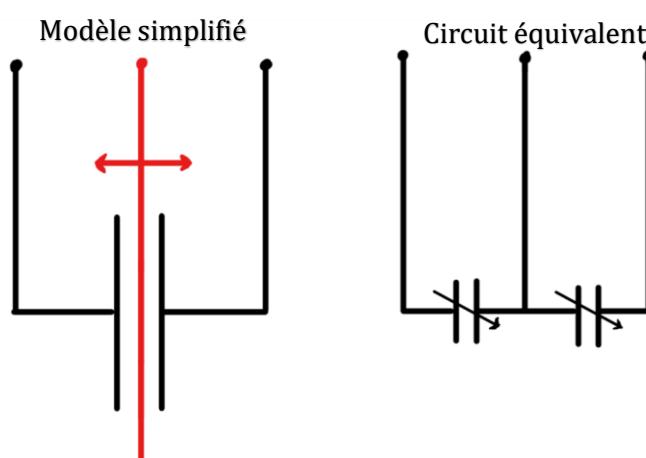


Figure 2. Modèle simplifié et circuit électrique équivalent.

Le gyroscope a un fonctionnement assez similaire. Cependant les lamelles sont disposées de manière circulaire.

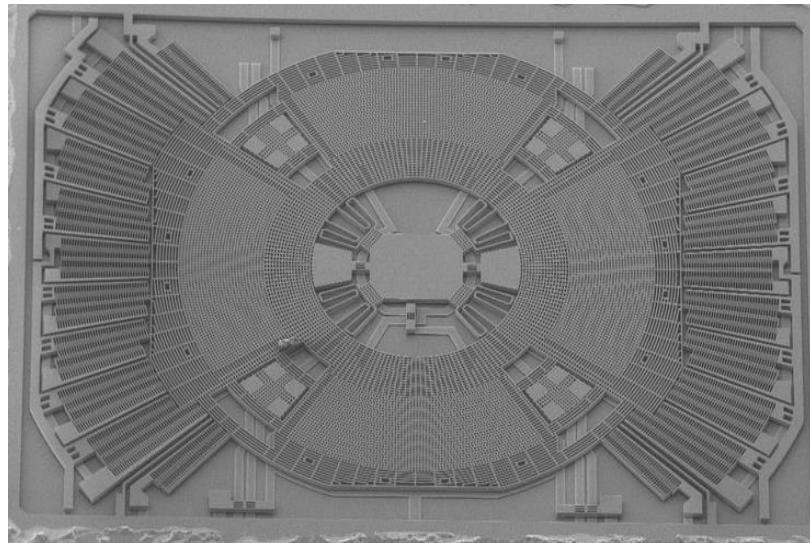
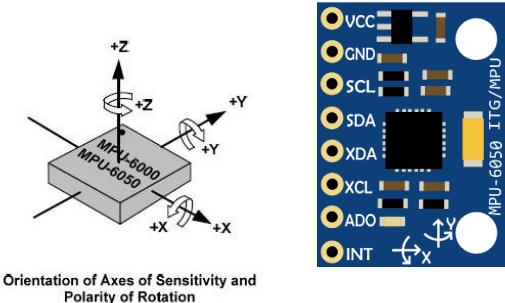


Figure 3 : Vu au microscope électronique d'un gyroscope.

Pour ce TIPE on a utilisé un module (MPU6050). Intégrant à la fois un gyroscope et un accéléromètre.

## 2. Communication avec le MPU6050



Ce capteur communique avec la carte Arduino grâce au port SDA et SCL. C'est le protocole de communication  $I^2C$ . Le MPU6050 possède ainsi un cache (une petite mémoire où il stocke les informations). La réelle difficulté ici, c'est de savoir comment lire correctement ce cache pour obtenir les informations. Le code étant inclus en C ou C++ dans la datasheet du constructeur, on ne s'est pas trop intéressé à la compréhension en profondeur de celui-ci. Par contre, il convient évidemment de le comprendre en substance et de l'expliquer.

### a. Configuration initiale.

```
void MPU6050_init(){
    Wire.begin();
    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(0x1C);
    Wire.write(0x00);
    Wire.endTransmission();
}
```

//Start communication with the address.  
//We want to write to the PWR\_MGMT\_1 register (6B hex)  
//Set the register bits as 00000000 to activate the gyro  
//End the transmission with the gyro.

//Start communication with the address.  
//We want to write to the GYRO\_CONFIG register (1B hex)  
//Set the register bits as 00010000 (500dps full scale)  
//End the transmission with the gyro

//Start communication with the address.  
//We want to write to the ACCEL\_CONFIG register (1A hex)  
//Set the register bits as 00010000 (+/- 8g full scale range)  
//End the transmission with the gyro

Il y a deux enjeux ici : trouver le registre d'écriture correct pour effectuer les commandes suivantes : réveiller le capteur et configurer le nombre de degrés par seconde que l'on veut obtenir. Ainsi systématiquement, nous initialisons la communication avec le capteur, nous écrivons sur le bit correspondant à la commande que nous voulons réaliser.

### b. Lecture des données

Une fois initialisé, le gyroscope peut transmettre les données. Ensuite comme la lecture des valeurs doit être effectuée de manière périodique, on met le code de lecture des informations dans une fonction que l'on appelle MPU\_signal().

```
void MPU6050_signal()
{
    Wire.beginTransmission(gyro_address);
    Wire.write(0x3B);
    Wire.endTransmission();
    Wire.requestFrom(gyro_address, 14);

    while(Wire.available() < 14);
    acc_x_raw = Wire.read()<<8|Wire.read();
    acc_y_raw = Wire.read()<<8|Wire.read();
    acc_z_raw = Wire.read()<<8|Wire.read();
    temperature = Wire.read()<<8|Wire.read();
    gyro_roll_raw = Wire.read()<<8|Wire.read();
    gyro_pitch_raw = Wire.read()<<8|Wire.read();
    gyro_yaw_raw = Wire.read()<<8|Wire.read();

    //Start communication with the gyro.
    //Start reading @ register 43h and auto increment with every read.
    //End the transmission.
    //Request 14 bytes from the gyro.

    //Wait until the 14 bytes are received.
    //Add the low and high byte to the acc_x_raw variable.
    //Add the low and high byte to the acc_y_raw variable.
    //Add the low and high byte to the acc_z_raw variable.
    //Add the low and high byte to the temperature variable.
    //Read high and low part of the angular data.
    //Read high and low part of the angular data.
    //Read high and low part of the angular data.
}
```

### c. Calibration.

Comme l'ensemble des capteurs, notre module gyroscope- accéléromètre est bruitée. C'est bien pour cela que ce TIPE comporte une grande partie sur la filtration du bruit à l'aide de Kalman. Cependant avant ce procédé complexe. Une calibration bien plus simple est nécessaire. Rien que pour supprimer ce que l'on appelle le offset. Par exemple lorsque l'on regarde les valeurs immobiles de l'axe x de l'accéléromètre celui-ci indique toujours une accélération de  $3 \text{ m.s}^{-2}$ . C'est pour cela qu'on effectue une calibration préliminaire c'est-à-dire de relever à peu près 3000 valeurs du gyroscope, en faire la somme et ensuite de diviser le tout par 3000. Ensuite, on va soustraire cette valeur aux valeurs du gyroscope.

```
void calibration_MP6050(){
    digitalWrite(13,HIGH);
    delay(5);
    //Calibration
    for (iteratif_1 = 0; iteratif_1 < 3000 ; iteratif_1 ++){           //Accumule 3000 valeurs pour la calibration
        if(iteratif_1 % 15 == 0)digitalWrite(13, !digitalRead(13));      //Fait clignoter la LED pour indiquer la calibration.
        gyro_signal();
        //Appelle la fonction gyroscope
        gyro_roll_cal +=gyro_roll_raw;
        gyro_pitch_cal +=gyro_pitch_raw;                                // Fait la somme sur 3000 valeurs sur tous les axes
        gyro_yaw_cal += gyro_yaw_raw;
        acc_x_cal +=acc_x_raw;
        acc_y_cal +=acc_y_raw;
        acc_z_cal +=acc_z_raw;

        //Maintenant on a obtenu 3000 valeurs on peut faire la moyenne
        gyro_roll_cal /= 3000;                                         //Divise le total par 3000
        gyro_pitch_cal /= 3000;
        gyro_yaw_cal /= 3000;
        acc_x_cal /=3000;
        acc_y_cal/= 3000;
        acc_z_cal /= 3000;
```

A cette étape nous pouvons lire les valeurs en vitesse angulaire du gyroscope et celles en accélération de l'accéléromètre.

Grâce à l'accéléromètre nous allons pouvoir déduire l'angle du roulis et du tangage. Tout d'abord nos angles seront orientés avec la convention classique  $YPR$  ( $Yaw(\psi)$ ,  $Pitch(\theta)$ ,  $Roll(\phi)$ ) (Figure 4.) :

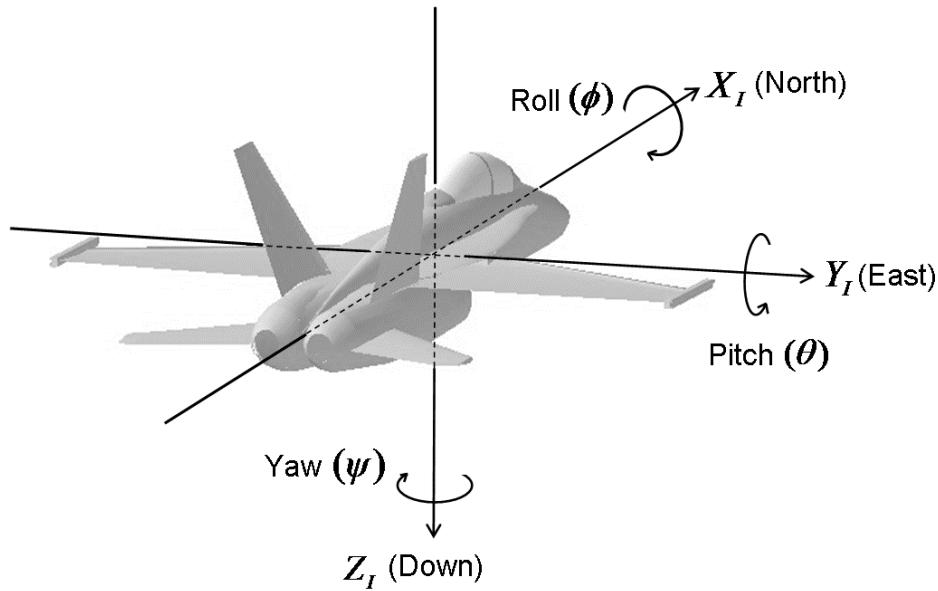


Figure 4. Définition des axes de rotation et du système de coordonnées.

### 3. Calcul des angles $\phi$ et $\theta$ à partir de l'accéléromètre.

Supposons d'abord que l'accéléromètre soit immobile et qu'il subit le champ gravitationnel  $\vec{g}$ .

Le vecteur rendu par l'accéléromètre serait alors

$$\vec{G} = \begin{pmatrix} G_x \\ G_y \\ G_z \end{pmatrix} = \mathbf{R}\vec{g}$$

avec  $\mathbf{R}$  une matrice de rotation décrivant l'orientation de l'accéléromètre relativement au système de coordonnées d'un référentiel terrestre.

Supposons à présent que l'on place à l'origine l'accéléromètre en alignant l'axe Z de celui avec l'accélération  $\vec{g}$ .

$$\vec{G} = \begin{pmatrix} G_x \\ G_y \\ G_z \end{pmatrix} = \mathbf{R}\vec{g} = \mathbf{R} \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}$$

Les matrices de rotations qui transforment un vecteur dans les coordonnées  $\vec{x}, \vec{y}, \vec{z}$  à une rotation d'angle  $\phi, \theta, \psi$  sont :

$$R_x(\phi) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & \sin\phi \\ 0 & -\sin\phi & \cos\phi \end{pmatrix}$$

$$R_y(\theta) = \begin{pmatrix} \cos\theta & 0 & -\sin\theta \\ 0 & 1 & 0 \\ \sin\theta & 0 & \cos\theta \end{pmatrix}$$

$$R_z(\psi) = \begin{pmatrix} \cos\psi & \sin\psi & 0 \\ -\sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

On remarque que les déterminants de ces matrices est bien égal à 1 et respectivement  $M \times {}^t M = I_n$ . Ce sont donc bien des matrices de rotation. On démontre facilement que ces matrices ne sont pas commutatives. On est donc obligé de choisir une séquence de rotation bien précise.

Dans le cadre de notre projet nous utiliserons la séquence de rotation de l'aérospatial.  $R_{xyz}$

$$\begin{aligned} Gg &= R_{xyz} \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} = R_x(\phi)R_y(\theta)R_z(\psi) \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \\ &= \begin{pmatrix} \cos\theta\cos\psi & \cos\theta\sin\psi & -\sin\theta \\ \cos\psi\sin\theta\sin\phi - \cos\phi\sin\psi & \cos\phi\cos\psi + \sin\theta\sin\phi\sin\psi & \cos\theta\sin\phi \\ \cos\phi\cos\psi\sin\theta + \sin\phi\sin\psi & \cos\phi\sin\theta\sin\psi - \cos\psi\sin\phi & \cos\theta\cos\phi \end{pmatrix} \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \\ &= \begin{pmatrix} -\sin\theta \\ \cos\theta\sin\phi \\ \cos\theta\cos\phi \end{pmatrix} \end{aligned}$$

On remarque ici qu'en utilisant la séquence  $x, y, z$  qu'on a isolé les angles  $\theta$  et  $\phi$ .

$$\frac{G}{\|G\|} = \begin{pmatrix} -\sin\theta \\ \cos\theta\sin\phi \\ \cos\theta\cos\phi \end{pmatrix} \Rightarrow \frac{1}{\sqrt{G_x^2 + G_y^2 + G_z^2}} \begin{pmatrix} G_x \\ G_y \\ G_z \end{pmatrix} = \begin{pmatrix} -\sin\theta \\ \cos\theta\sin\phi \\ \cos\theta\cos\phi \end{pmatrix}$$

*On veut trouver l'angle  $\phi$ .*

$$\tan \phi = \frac{\sin\phi}{\cos\phi} = \frac{\cos\theta\sin\phi}{\cos\theta\cos\phi} = \frac{G_y}{G_z} \Rightarrow \phi = \arctan\left(\frac{G_y}{G_z}\right) \text{ Attention à ici au signe.}$$

*de même*

$$\tan \theta = \frac{\sin\theta}{\cos\theta} = \frac{-G_x}{\sqrt{\cos\theta^2(\sin\phi^2 + \cos\theta^2)}} = \frac{-G_x}{\sqrt{(\cos\theta\sin\phi)^2 + (\cos\theta\cos\phi)^2}} = \frac{-G_z}{\sqrt{G_z^2 + G_y^2}}$$

$$\text{Donc } \theta = \arctan\left(\frac{-G_z}{\sqrt{G_z^2 + G_y^2}}\right) \text{ Attention à ici au signe ( pris en compte par le suite).}$$

#### 4. Implémentation dans le code :

```
double roll_angle = atan2(acc_y_raw, acc_z_raw) * 57.2951;
double pitch_angle = atan(-acc_x_raw / sqrt(acc_y_raw * acc_y_raw + acc_z_raw * acc_z_raw)) * 57.2951;
```

On remarque alors la fonction atan2 : <https://fr.wikipedia.org/wiki/Atan2> qui élimine déjà les problèmes de signes. Ne prenant que deux arguments elle ne peut pas être utilisée pour le calcul de  $\theta$ .

Un autre problème pouvant survenir est la multiplicité des solutions aux multiples de  $2\pi$ . On peut donc restreindre l'angle à varier entre  $[0, \pi]$ . Cependant il reste encore des solutions doubles. En effet :

$$\begin{pmatrix} -\sin\theta \\ \cos\theta\sin\phi \\ \cos\theta\cos\phi \end{pmatrix} = \begin{pmatrix} -\sin(\pi - \theta) \\ \cos(\pi - \theta)\sin(\phi + \pi) \\ \cos(\pi - \theta)\cos(\phi + \pi) \end{pmatrix}$$

Une solution est donc de restreindre un des deux angles à un intervalle  $[-\frac{\pi}{2}, \frac{\pi}{2}]$ . Ceci élimine en même temps tous problèmes de Gimbal lock. Voici donc l'implémentation du code.

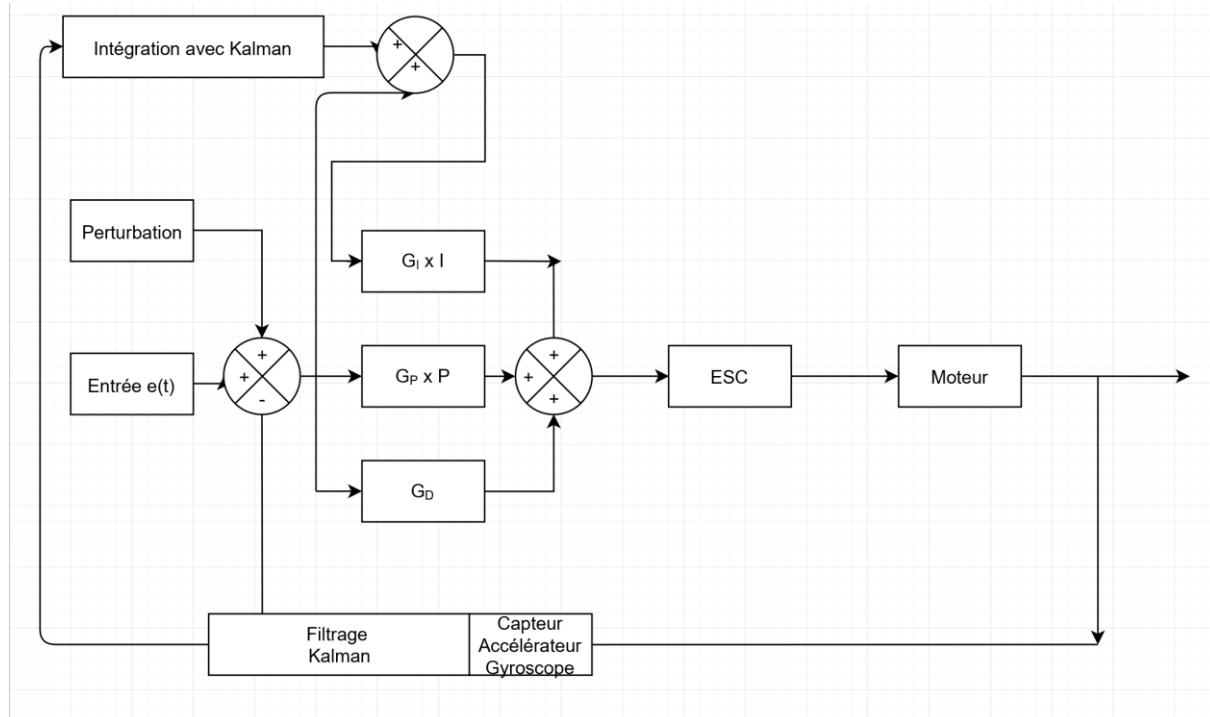
```
// Ceci résout le problème de la transition de l'angle de l'accéléromètre entre -pi et pi
if ((roll_angle < -90 && roll_angle_kal > 90) || (roll_angle > 90 && roll_angle_kal < -90)) {
    kalmanX.setAngle(roll_angle);
    roll_angle_kal = roll_angle;
}
else    roll_angle_kal = kalmanX.getAngle(roll_angle, gyro_roll_d, dt); // Calculate the angle using a Kalman filter

if (abs(roll_angle_kal) > 90) gyro_pitch_d = -gyro_pitch_d; // Inverse le signal pour avoir rester dans l'intervalle -pi/2 \pi/2
pitch_angle_kal = kalmanY.getAngle(pitch_angle, gyro_pitch_d, dt);
```

## IV. Stabilisation

### 1. Asservissement en vitesse angulaire et position

Le fonctionnement détaillé notre système peut être modélisé par le diagramme en blocs suivant.



On note qu'il s'agit d'un système asservi. Sur la boucle de retour on voit la présence d'un gyroscope et d'un accéléromètre. Le premier composant permet de connaître la position angulaire de notre objet, le second de connaître sa vitesse angulaire. Ces données seront exploitées par le filtre de kalman situé sur cette même boucle retour, et dont le fonctionnement précis sera détaillé dans une partie ultérieure de ce rapport.

Sur la chaîne directe on note la présence d'un régulateur PID dont l'implémentation sera détaillée plus tard. On note cependant que la correction angulaire ne s'effectue que sur l'intégrateur du PID. Le produit et la dérivée ne font que rendre le mouvement des moteurs fluide.

L'asservissement se fait donc grâce au mouvements des moteurs, qui sont commandés par une consigne dépendant de calculs réalisés à l'aide de mesures de position et de vitesse angulaire angulaire.

### 2. Implémentation d'un algorithme PID

Introduction au correcteur PID:

Le correcteur PID (Produit Intégrale Dérivée) a pour fonction de corriger les erreurs d'asservissement du stabilisateur. Comme son nom l'indique, ce correcteur renvoie une valeur qui correspond à la somme d'un produit, d'une intégrale, et d'une dérivée sur la différence entre la valeur demandée et la valeur mesurée par le gyroscope pour chaque axe:

$$P_{output} = (val_{gyro} - val_{demandée}) \times P_{gain}$$

$$I_{output} = I_{output_{pcdt}} + ((val_{gyro} - val_{demandée}) \times I_{gain}) + \text{Angle ajusté}$$

$$D_{output} = (val_{gyro} - val_{demandée} - (val_{gyro_{pcdt}} - val_{demandée_{pcdt}})) \times D_{gain}$$

$$PID_{output} = P_{output} + I_{output} + D_{output}$$

La caméra peut avoir trois mouvements différents : le roulis (roll), le tangage (pitch), et le lacet (yaw). Il y a donc 3 valeurs à calculer pour que celle-ci soit parfaitement stabilisée. Voici les rôles des différentes composantes de la valeur renvoyée par notre fonction PID:

### Produit :

Il permet une première stabilisation de la caméra, mais il y aura des oscillations qui ne pourront être annulé par le produit seul. Augmenter le gain permet un meilleur temps de réponse.

### Intégrale :

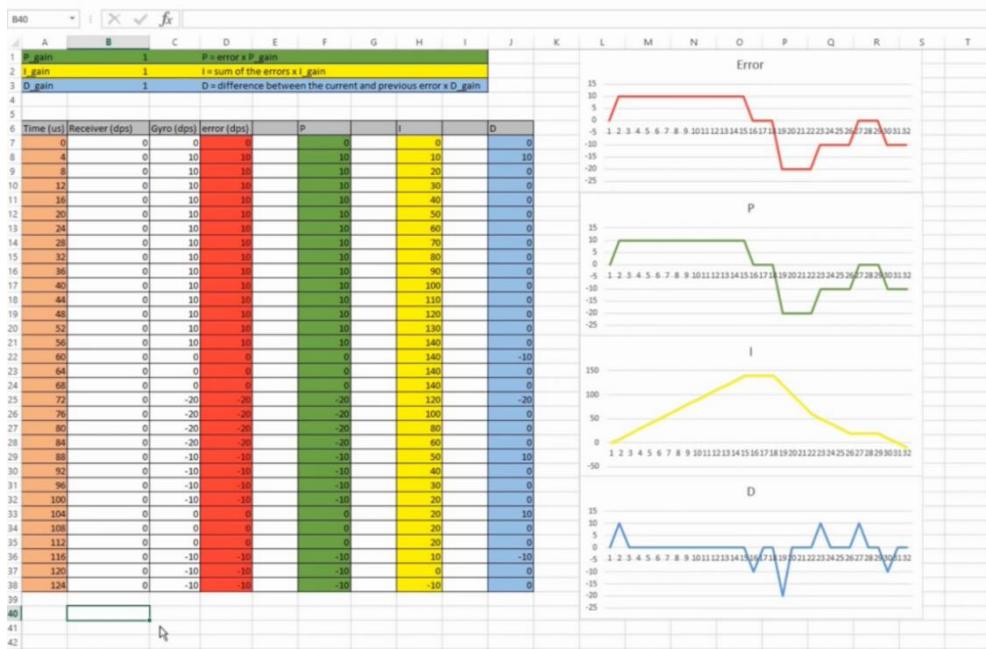
Etant la somme des erreurs, elle permet un redressement rapide : par exemple, si la caméra se penche sur la gauche violement, l'intégrale va immédiatement prendre une très grande valeur, et donc corriger l'angle de celle-ci rapidement, c'est un complément du produit, elle augmente la réactivité.

### Dérivée :

C'est la dernière valeur à calculer, elle permet d'amortir la compensation du mouvement non-voulu, pour éviter une sur-correction. Plus la dérivée de la vitesse

angulaire est proche de 0, moins la dérivée entre en compte, et moins le mouvement de compensation est rapide.

Ci-dessous, un graphe des fonctions produit (en vert), intégrale (en jaune), et dérivée (en bleu) avec des gains de 1 en fonction de l'erreur (en rouge).



Ici, on voit nettement que le produit compense exactement l'erreur, l'intégrale augmente la vitesse de compensation si l'erreur continue d'augmenter, et la dérivée intervient pour amortir la compensation en fin de correction.

### 3. Étalonage des valeurs du PID.

Nous avons parlé plus haut d'une variable particulière qui intervient dans chaque équation : le gain. Nous allons voir dans cette partie comment déterminer ces valeurs : Ici il est nécessaire de traiter chaque axe indépendamment. Il n'y aucune raison que le moteur du roulis est les mêmes gains que le moteur du tangage.

- 1) S'occuper du gain P en premier. Augmenter progressivement la valeur du gain jusqu'à obtenir une sur-réaction (oscillation). Baisser à nouveau la valeur jusqu'à ce que l'axe réagisse de manière douce. Baisser cette valeur de 25% : c'est la valeur à garder pour ce gain.
- 2) Réglage du gain de la dérivée à présent. Augmenter par pas de 0.2 ce gain. Jusqu'à obtenir des oscillations et une surcompensation. Baisser cette valeur de 50% : c'est notre valeur pour ce gain.
- 3) Réglage des gains sur l'intégrale : augmenter la valeur par tranche de 0.01 jusqu'à ce que l'axe oscille doucement. Baisser cette valeur de 50% : c'est notre valeur pour ce gain.

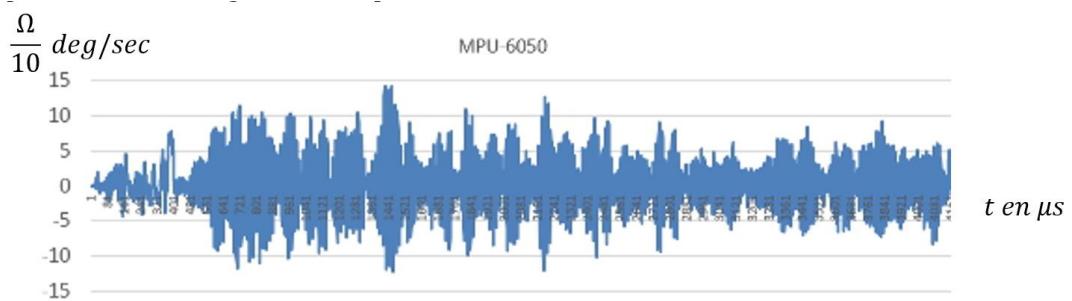
On remarque plusieurs caractéristiques. Tout d'abord la correction en vitesse angulaire n'est effectué que dans le I du PID. On remarque que la boucle d'asservissement principale est un asservissement en vitesse angulaire. Ceci est logique car nos moteurs se commandent uniquement en vitesse angulaire.

## V. Le filtre de Kalman

### 1. Présentation

Pourquoi un filtre de Kalman ?

Voici un Graphique montrant les valeurs obtenues lorsque l'on laisse notre capteur de vitesse angulaire au repos :



En ordonnée des vitesses angulaires en dixième de degrés par seconde et en abscisse le temps en  $\mu s$

C'est chaotique, le bruit est manifestement très important. Si l'on veut intégrer cette vitesse angulaire pour obtenir une position angulaire par exemple on se rend bien vite compte que ce signal ne peut pas convenir. La position va bien vite diverger vers des valeurs très incorrectes.

C'est bien pour cela qu'il faut filtrer ces données. Faisons une moyenne me direz-vous ? On appelle ceci le filtre complémentaire. Lors de notre TIPE en Sup nous avions utilisé cette méthode pour un drone. Cependant on se rend bien vite compte que celle-ci ne correspond pas à la réalité et sa précision ne nous convient pas. Elle adoucit la courbe certes mais justement cache des grandes perturbation qui peuvent arriver (vent, choc etc... ) Un stabilisateur faisant face à des perturbations en permanence ne peut fonctionner correctement avec cette méthode de filtrage.

C'est bien pour cela que nous allons utiliser une méthode de filtrage dite de KALMAN :

Le filtre de Kalman est un algorithme qui peut avoir plusieurs applications, dans le cadre de notre étude nous nous intéresseront particulièrement à ses applications suivantes:

- Filtrer le bruit d'un signal
- Prévoir de futurs états

En effet, si il est parfaitement possible de stabiliser un système sans utiliser ce filtre, par exemple en utilisant ce que l'on appelle un filtre complémentaire, mais ce filtre n'est pas parfait il y a des imprécisions, des « bruits ». L'avantage du filtre de Kalman est qu'il peut calculer directement la position, et même donner une estimation de l'erreur sur la vitesse à venir afin de donner une réponse plus adaptée dans le but de stabiliser le système sans donner d'à coups.

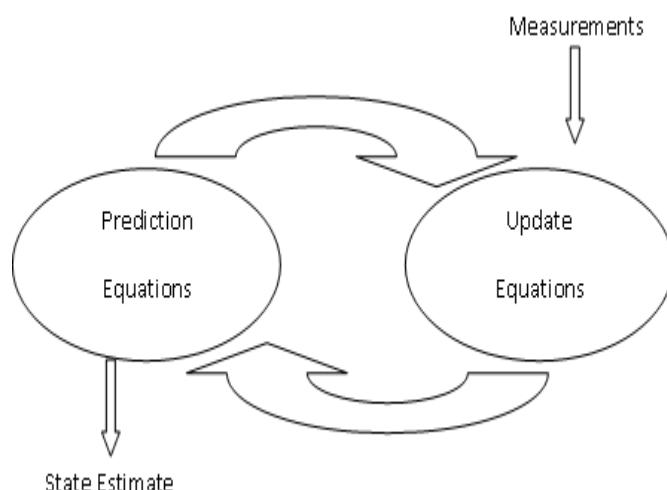
Ce filtre ne fonctionne de manière optimale que s'il est utilisé dans le cadre d'un système linéaire, ce qui est à priori notre cas.

Mathématiquement, le principe repose sur un rapport entre la dernière mesure et une matrice définissant la probabilité pour cette mesure d'être pertinente

## 2. Le principe de fonctionnement du filtre

Le fonctionnement du filtre repose sur deux types d'équations.

- Les premières sont les équations de prédiction : se basant sur les positions précédentes et la commande, elles calculent la position actuelle logique
- Les secondes sont les équations de mise à jour, basées sur les informations fournies par les capteurs, considérant la fiabilité de chacun.



Le filtre fonctionne ensuite en prédisant la position théorique avec les équations de prédiction, comparant cette position avec celle effective mesurée par les capteurs et en corrigeant l'erreur. Ce principe tourne ensuite en boucle indéfiniment.

### Les équations

On expliquera ici le fonctionnement calculatoire en dimension 1. Plusieurs variables entrent en jeu lors du calcul. On les pose de la façon suivante :

$EST_t$  = l'estimation de l'état futur « calculée à t ». C'est ce que l'on cherche à déterminer

$EST_{t-1}$  = l'estimation de l'état actuel, « calculée à t-1 »

$M$  = la mesure par les capteurs de l'état actuel du système

$E_E$  = l'erreur d'estimation

$E_M$  = l'erreur de mesure (due aux capteurs)

$KG$  = gain de kalman

Le gain de kalman est une variable comprise entre 0 et 1, définie de la manière suivante.

$$KG = \frac{E_E}{E_E + E_M}$$

$$EST_t = EST_{t-1} + KG (M - EST_{t-1})$$

Mais que signifie la valeur adoptée par KG ?

Les mesures sont précises

$E_M=0$

KG

1

Les estimations sont mauvaises



0.9

0.8

0.7

0.6

0.5

0.4

0.3

0.2

0.1

0

Les mesures sont imprécises

Les estimations sont parfaites

Dans les faits les valeurs 1 et 0 ne seront jamais atteinte, et heureusement. Si  $E_E$  est nul on a  $KG = 0$  quelle que soit la valeur de  $E_M$  donc on ne pourrait rien déduire sur celle ci

### 3. Étude sur plusieurs variables.

L'étude précédente ne considérait qu'une seule variable. Généralisons cette étude pour un nombre supérieur de variable.

Dans un premier temps nous devons mettre en place les équations de prédiction, qui permettent d'estimer à priori l'état du système à un instant  $t$  en se basant sur son état à l'état précédent  $t-1$

Définition des variables :

$$x_t = \begin{pmatrix} \theta \\ \dot{\theta}_b \end{pmatrix}_t : \text{état du système à un instant } t$$

$\tilde{x}_{t-1|t-1}$ : estimation de l'état précédent basée sur les estimations encore antérieures (previous estimate state)

$\tilde{x}_{(t|t-1)}$ : estimation de l'état actuel basée sur les estimations précédentes (a priori state)

$\tilde{x}_{(t|t)}$  : état actuel observé par les capteurs (a posteriori)

$F = \begin{pmatrix} 1 & -\Delta t \\ 0 & 1 \end{pmatrix}$  il s'agit de la matrice de transition d'un état

$B = \begin{pmatrix} \Delta t \\ 0 \end{pmatrix}$  ne sert que d'un point de vue

$\omega_t \sim N(0, Q_t)$  suit une loi normale centrée en 0 , paramètre  $Q_t$

$Q_t = \begin{pmatrix} Q_\theta & 0 \\ 0 & Q_{\dot{\theta}_b} \end{pmatrix}$  : coefficients constants issus de l'expérimentation.

$$x_t = Fx_{t-1} + B\dot{\theta}_t + \omega_t$$

L'état du système est modélisé par un vecteur ayant pour coordonnées une position angulaire et une vitesse angulaire, car

- Notre système est conçu pour détecter les mouvements angulaires et la correction s'effectue avec les rotations des moteurs, qui sont aussi des vitesses angulaires
- Le filtre de kalman, après calcul, nous rend une valeur qui est un angle

On peut donc effectuer les calculs :

$$\begin{aligned} x_{(t|t)} &= \begin{pmatrix} 1 & -\Delta t \\ 0 & 1 \end{pmatrix} \begin{pmatrix} \theta_{t-1} \\ \dot{\theta}_b \end{pmatrix} + \begin{pmatrix} \Delta t \\ 0 \end{pmatrix} \dot{\theta}_t + \omega_t \\ &= \begin{pmatrix} \theta_{t-1} - \Delta t \dot{\theta}_b + \Delta t \dot{\theta}_t \\ \dot{\theta}_b \end{pmatrix} + \omega_t \\ &= \begin{pmatrix} \theta_{t-1} + \Delta t(\theta_t - \theta_b) \\ \dot{\theta}_b \end{pmatrix} + \omega_t \end{aligned}$$

$$\begin{pmatrix} \theta_t \\ \dot{\theta}_b \end{pmatrix} = \begin{pmatrix} \theta_{t-1} + \Delta t(\theta_t - \theta_b) \\ \dot{\theta}_b \end{pmatrix} + \omega_t$$

Mais l'état actuel observé peut être défini comme étant l'état actuel à priori (celui calculé au moment précédent) auquel s'ajoutent les imprécisions de mesure et les perturbations subies par le système, qui sont regroupées dans la variable  $\omega_t$ . On en déduit donc

$$x_{t|t-1} = Fx_{t-1} + B\dot{\theta}_t$$

Où  $x_{t|t-1}$  est estimation de l'état actuel basée sur les estimation précédentes.

On obtient donc la relation suivante  $\tilde{x}_{t|t-1} = \begin{pmatrix} \theta_t + \Delta t(\dot{\theta}_t - \dot{\theta}_b) \\ \dot{\theta}_b \end{pmatrix}$

Une fois ces calculs effectués, l'étape suivante consiste à calculer la covariance de l'erreur calculée à l'instant précédent (t-1). Cette matrice est utilisée pour estimer le taux de confiance que l'on accorde à nos estimations de l'état à venir du système (on peut faire une analogie avec le E<sub>E</sub> du paragraphe précédent. Dans les faits, il s'agit d'une matrice 2x2 variable souvent initialisée comme une matrice diagonale ou la matrice nulle. Elle aura une utilité lors de calculs qui seront détaillés plus tard ( cf covariance d'innovation). Pour la calculer on part de l'expression suivante

$$P_{t|t-1} = FP_{t-1|t-1}^T F + Q_t$$

$$\begin{aligned} P_{t|t-1} &= FP_{t-1|t-1}^T F + Q_t \\ &= \begin{pmatrix} 1 & -\Delta t \\ 0 & 1 \end{pmatrix} \begin{pmatrix} P_{00} & P_{01} \\ P_{10} & P_{11} \end{pmatrix} \begin{pmatrix} 1 & 0 \\ -\Delta t & 1 \end{pmatrix} + \begin{pmatrix} Q_\theta & 0 \\ 0 & Q_{\dot{\theta}_b} \end{pmatrix} \Delta t \\ &= \begin{pmatrix} 1 & -\Delta t \\ 0 & 1 \end{pmatrix} \begin{pmatrix} P_{00} - \Delta t P_{01} & P_{01} \\ P_{10} - \Delta t P_{11} & P_{11} \end{pmatrix} + \begin{pmatrix} Q_\theta & 0 \\ 0 & Q_{\dot{\theta}_b} \end{pmatrix} \Delta t \\ &= \begin{pmatrix} P_{00} - \Delta t P_{01} - \Delta t P_{10} + \Delta t^2 P_{11} & P_{01} - \Delta t P_{11} \\ P_{10} - \Delta t P_{11} & P_{11} \end{pmatrix} + \begin{pmatrix} Q_\theta & 0 \\ 0 & Q_{\dot{\theta}_b} \end{pmatrix} \end{aligned}$$

Nous en avons fini avec les équations de prédiction, il va maintenant être nécessaire de mettre en place les équations d'actualisation du système.

Nous devons d'abord définir mathématiquement ce que sont nos mesures

$z_t$  : la mesure de l'angle

$H = [1 \ 0]$  matrice nous permettant de ne garder que la composante angulaire de  $x_t$ .

$v_t \rightsquigarrow N(\mathbf{0}, R)$  loi normale centrée en zéro , de paramètre  $R = V(v)$  ;

$V(v)$ : variance du capteur donc la moyenne des carrés des écarts à la moyenne de nos

*mesures expérimentales.*

$$\begin{aligned}\mathbf{z}_t &= \mathbf{Hx}_t + \boldsymbol{\nu}_t \\ &= \begin{pmatrix} \mathbf{1} \\ \mathbf{0} \end{pmatrix} \begin{pmatrix} \boldsymbol{\theta}_t \\ \dot{\boldsymbol{\theta}}_b \end{pmatrix} + \boldsymbol{\nu}_t = \begin{pmatrix} \boldsymbol{\theta}_t \\ \mathbf{0} \end{pmatrix}\end{aligned}$$

Interprétation de l'équation:

La mesure de l'angle est donc l'angle à l'instant t « parfait » auquel s'ajoutent les facteurs d'imprécision de la mesure comme le fait que les capteurs ne soient pas fiables à 100%, modélisés par une loi normale. On note que l'on obtient un vecteur donc la seule composante est un angle donc l'homogénéité est respectée.

Cette définition nous permettra de calculer ce que l'on appelle la variable d'innovation. On la définit de la manière suivante

$$\begin{aligned}\tilde{\mathbf{y}}_t &= \mathbf{z}_t - \mathbf{H}\tilde{\mathbf{x}}_{t|t-1} \\ &= \begin{pmatrix} \boldsymbol{\theta}_t \\ \mathbf{0} \end{pmatrix} + \boldsymbol{\nu}_t - \begin{pmatrix} \mathbf{1} \\ \mathbf{0} \end{pmatrix} \begin{pmatrix} \boldsymbol{\theta}_t + \Delta t(\dot{\boldsymbol{\theta}}_t - \dot{\boldsymbol{\theta}}_b) \\ \dot{\boldsymbol{\theta}}_b \end{pmatrix}\end{aligned}$$

Interprétation

L'innovation est définie comme étant la différence entre la valeur mesurée qui représente l'état effectif du système et l'état du système à priori, calculé en fonction de l'état à l'insant précédent. Cela permet de faire ressortir les modifications subies par le système. Dans notre cas, ce sera par exemple un déplacement dû à un faux mouvement de l'opérateur de la caméra. On notera que même s'il n'y a aucune modification,  $\tilde{\mathbf{y}}_t$  n'est pas nul pour autant à cause de  $\boldsymbol{\nu}_t$ . Les imprécisions de capteurs peuvent amener le système à « croire qu'il y a des innovations alors qu'il est resté stable »

Calcul de la covariance de l'innovation.

Ce scalaire représente la confiance que l'on accorde aux mesures faites par les capteurs, et est basée sur la matrice de covariance de l'erreur à priori et la covariance de la mesure notée R. plus la valeur de S est élevée, moins mesures sont fiables. On peut faire une analogie entre S et le E<sub>M</sub> du paragraphe précédent.

$$\begin{aligned}\mathbf{S}_t &= \mathbf{H} \mathbf{P}_{t|t-1}^T \mathbf{H} + \mathbf{R} \\ &= (\mathbf{1} \ \mathbf{0}) \begin{pmatrix} \mathbf{P}_{00} & \mathbf{P}_{01} \\ \mathbf{P}_{10} & \mathbf{P}_{11} \end{pmatrix}_{t|t-1} - (\mathbf{1}) + \mathbb{V}(\boldsymbol{\nu}) \\ &= \mathbf{P}_{00 \ t|t-1} + \mathbb{V}(\boldsymbol{\nu})\end{aligned}$$

On en vient au calcul de la variable de Kalman (Gain de Kalman ) proprement dire. Elle est indiquée de t car elle est variable en fonction du temps

$$\mathbf{K}_t = \mathbf{P}_{t|t-1}^T \mathbf{H} \mathbf{S}_k^{-1}$$

$$= \begin{pmatrix} \mathbf{P}_{00} & \mathbf{P}_{01} \\ \mathbf{P}_{10} & \mathbf{P}_{11} \end{pmatrix}_{t|t-1} \begin{pmatrix} \mathbf{1} \\ \mathbf{0} \end{pmatrix} \frac{\mathbf{1}}{\mathbf{P}_{00\ t|t-1} + \mathbb{V}(\boldsymbol{\nu})} = \frac{\begin{pmatrix} \mathbf{P}_{00} \\ \mathbf{P}_{10} \end{pmatrix}_{t|t-1}}{\mathbf{P}_{00\ t|t-1} + \mathbb{V}(\boldsymbol{\nu})}$$

$$\mathbf{K}_t = \begin{pmatrix} \mathbf{K}_0 \\ \mathbf{K}_1 \end{pmatrix}_t$$

Avoir trouvé l'expression du gain de kalman nous permet donc de mettre à jour l'estimation de l'état actuel à posteriori. Il se définit comme étant la somme de l'état à priori  $\tilde{x}_{t|t-1}$  et de l'innovation multipliée par le gain de Kalman.

$$\begin{aligned} \tilde{x}_{t|t} &= \tilde{x}_{t|t-1} + \mathbf{K}_t \tilde{y}_t \\ \begin{pmatrix} \theta_t \\ \dot{\theta}_b \end{pmatrix} &= \begin{pmatrix} \theta_{t-1} \\ \dot{\theta}_b \end{pmatrix} + \begin{pmatrix} \mathbf{K}_0 \\ \mathbf{K}_1 \end{pmatrix}_t \tilde{y}_t \\ &= \begin{pmatrix} \theta_{t-1} \\ \dot{\theta}_b \end{pmatrix} + \begin{pmatrix} \mathbf{K}_0 + \tilde{y}_t \\ \mathbf{K}_1 + \tilde{y}_t \end{pmatrix}_t \end{aligned}$$

La dernière étape de cette série de calcul consiste à mettre à jour la covariance de l'erreur à posteriori

$$\mathbf{P}_{t|t} = (\mathbf{I} - \mathbf{K}_t \mathbf{H}) \mathbf{P}_{t|t-1} \text{ avec I la matrice identité de dimension 2}$$

#### 4. Interprétation générale

A la fin de la boucle, l'effet du filtre aura été de corriger la matrice de covariance de l'erreur, en fonction de la correction apportée par rapport à l'état théorique dans lequel le système aurait dû se trouver.

Par exemple, si l'on considère que l'état parfait est un ange de 0 radians et qu'on ne donne aucune consigne de déplacement (comme c'est le cas dans le cas de notre stabilisation), et qu'à l'instant t-1 on l'angle est effectivement de 0 radians. L'angle estimé sera donc de 0 radians, mais une perturbation pourra venir modifier le système, la valeur de l'innovation  $\tilde{y}_t$  ne sera donc pas nulle.

Cela aura pour conséquence que le gain de Kalman ne sera pas nul non plus, donc la matrice  $\mathbf{P}_{t|t}$  sera mise à jour. Connaissant la covariance de l'erreur, on pourra donner une consigne permettant de corriger cette dernière.

L'erreur aura diminué à chaque mise à jour de la matrice P, jusqu'à avoir été entièrement compensée. Si d'autres perturbations s'ajoutent à la première, la matrice prendra des valeurs en conséquence, et une consigne sera donnée pour corriger toutes

les erreurs en même temps. Les mises à jours fréquentes des matrices permettent en effet au filtre de s'adapter rapidement aux changements.

## 5. Implémentation dans le code.

Une fois les équations et le principe compris. Il est alors nécessaire de coder ces équations. Pour cela nous avons décidé de créer une librairie (voir code en annexe). Cette librairie se compose de deux fichiers. Un fichier (.h) que l'on appelle le constructeur(header file) . Et le programme principal (.cpp). Dans le fichier (.h) on crée une classe et on définit l'ensemble des variables. Celles qui sont modifiables (public) et celle qui ne le sont pas (private).

Dans le fichier (.cpp) : ici sont effectués l'ensemble des calculs détaillés précédemment (voir fichier en annexe). Nous n'allons pas tout redétailler, mais voici par exemple comment nous mettons à jour la matrice de covariance.

```
//Étape-2 // Mise à jour de la matrice de covariance

P[0][0] += dt*(dt*(P[1][1]) - P[1][0] - P[0][1] + Q_angle);
P[0][1] -= dt*P[1][1];
P[1][0] -= dt*P[1][1];
P[1][1] += Q_error_s * dt;
```

## 6. Obtenir les valeurs à partir de notre librairie

- Étape 1 : On importe la librairie.

```
4 #include "Kalman.Lib.h"
```

- Étape 2 : On crée les instances de la classe Kalman.

```
//Création des instances de Kalman//
Kalman kalmanX ;
Kalman kalmanY ;
Kalman kalmanZ ;
```

- Étape 3 : On donne au filtre l'angle de départ à partir de l'angle de accéléromètre. Ici on utilise la fonction setAngle.

```
kalmanX.setAngle(roll_angle); // définition des angles de départ
kalmanY.setAngle(pitch_angle);
kalmanZ.setAngle(yaw_angle);
```

- Étape 4 : Calcul de angle : On donne au filtre de Kalman l'angle de l'accéléromètre, la vitesse angulaire et le temps d'exécution de boucle dt.

```
pitch_angle_kal = kalmanY.getAngle(pitch_angle, gyro_pitch_d, dt);
roll_angle_kal = kalmanX.getAngle(roll_angle, gyro_roll_d, dt);
yaw_angle_kal = kalmanZ.getAngle(yaw_angle, gyro_yaw_d, dt) ;
```

## 7. Précision concernant le calcul de dt.

Lors de l'intégration de la vitesse angulaire ce temps dt est essentiel. Le temps infinitésimal dt est proportionnel au temps d'exécution de la boucle. De cette manière ce temps varie selon la difficulté de calcul à un moment t. On obtient donc une valeur plus cohérente que si dt était une constante.

```
void loop() {
    double dt = (double) (micros() - timer) / 1000000; // Calcul de delta t
    timer = micros();
```

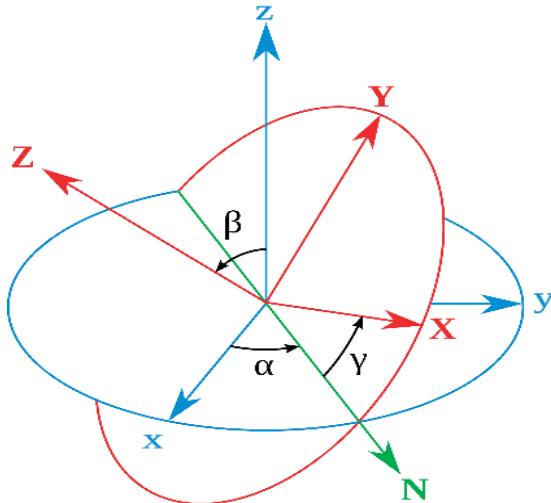
Ici à chaque tour de boucle dt se met à jour. La fonction micros(); étant le temps à un instant t. Cette fonction utilise l'horloge interne de l'Arduino.

## VI. Le gimbal lock

Dans cette partie nous étudierons un phénomène dont nous avons dû tenir compte durant notre étude : le gimbal lock. Pour comprendre ce phénomène il faut d'abord définir le modèle géométrique dans lequel nous travaillons : le modèle d'Euler

### 1. Modèle géométrique des angles d'Euler

Euler a proposé un modèle géométrique pour décrire l'orientation d'un solide par rapport à un axe. Celui-ci est défini comme tel :



Soit un système de points mobiles supposé indéformable.

On fixe un repère fixe direct  $(O; x, y, z)$  au centre de gravité du mobile.

On fixe trois points  $(X, Y, Z)$  sur le mobile. On définit alors le repère mobile par  $(O'; X, Y, Z)$ .  $O$  et  $O'$  étant ici confondus.

On définit :

- la ligne de noeud comme étant la droite de vecteur unitaire  $\vec{N}$  formée par l'intersection entre le plan  $(O; x, y)$  et le plan  $(O; X, Y)$  lorsque ceux-ci ne sont pas confondus.

- L'angle entre  $\vec{N}$  et  $\vec{x}$  noté  $\alpha$ .
- L'angle entre  $\vec{Z}$  et  $\vec{z}$  noté  $\beta$ .
- L'angle entre  $\vec{N}$  et  $\vec{X}$  noté  $\gamma$ .

Pour définir la position d'un mobile (par exemple un axe de stabilisateur) par rapport au repère fixe, on n'utilisera alors que 6 paramètres : les coordonnées de son centre de masse dans le repère  $(O;x,y,z)$  et les trois angles d'Euler définis plus haut. Pour déterminer les angles d'Euler, on supposera que  $O$  et  $O'$  sont confondus.

Ce modèle constitue une généralisation du repérage sphérique qui n'utilise que 3 paramètres pour déterminer l'emplacement d'un point dans une sphère : sa distance à l'axe et 2 angles (longitude et latitude).

## 2. Le problème du gimbal lock

Considérons un mobile orienté dans un espace d'euler comme ci contre

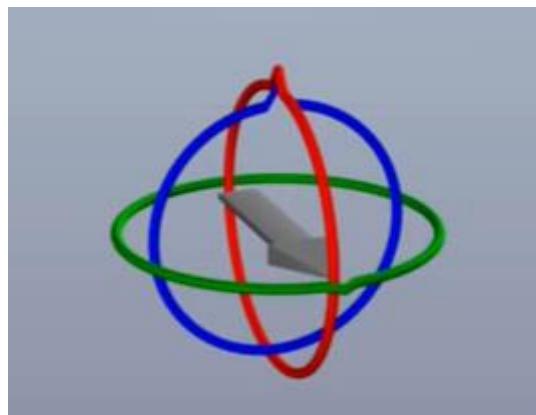


Figure 1 : Système au repos avant mouvement

Au cours de ses déplacements angulaires il sera amené à pivoter, et à adopter diverses positions

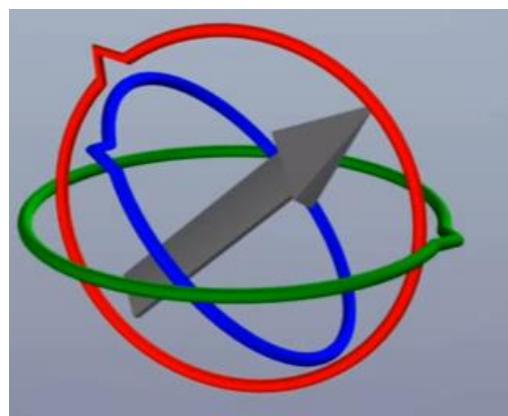


Figure2 : Système en position quelconque

A présent imaginons que la flèche continue de se redresser en suivant l'axe rouge en étant invariant selon les autres, on obtient la situation suivante

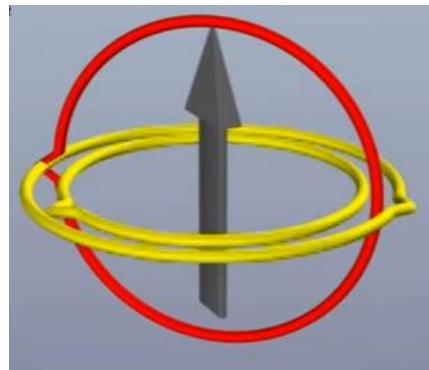


Figure 3 : Système « gimbal locké »

En quoi est-ce un problème ?

Imaginons qu'à présent, nous ayons besoin d'orienter la flèche à plat face à nous, (cf figure 4) c'est impossible en ne suivant qu'un seul axe.

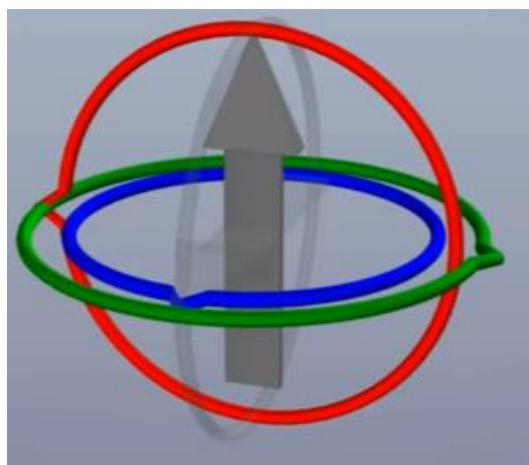


Figure 4 : axe manquant pour pivoter la flèche comme souhaité

Il est néanmoins possible de rejoindre la position recherchée en jouant sur les 3 axes simultanément ce qui donnera la trajectoire suivante

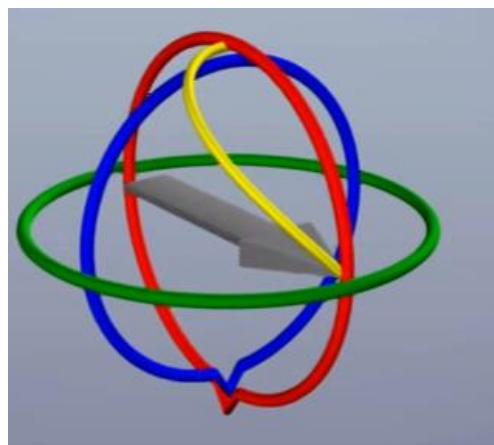


Figure 5 : Trajectoire de la flèche en réponse à une consigne rectiligne en gimbal lock

On a donc une trajectoire courbe avec une consigne rectiligne qu'on n'avait pas prévue. Ne pas contrôler précisément la trajectoire qu'emprunte un objet que l'on commande peut poser des problèmes évidents si l'on souhaite effectuer une stabilisation précise qui nécessite des déplacements minimes.

### 3. La correction du problème

Il y a plusieurs procédés permettant de résoudre ce problème

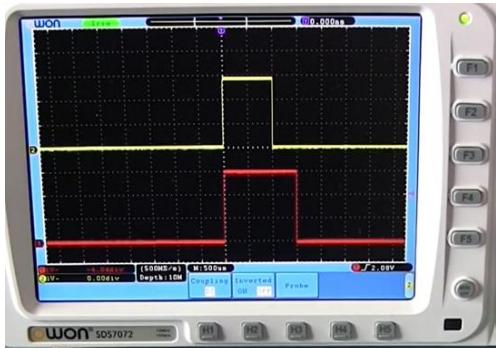
- Utiliser des quaternions

Nous avons exclu cette solution car cette solution est relativement complexe et engendre des calculs augmentant la complexité de notre programme.

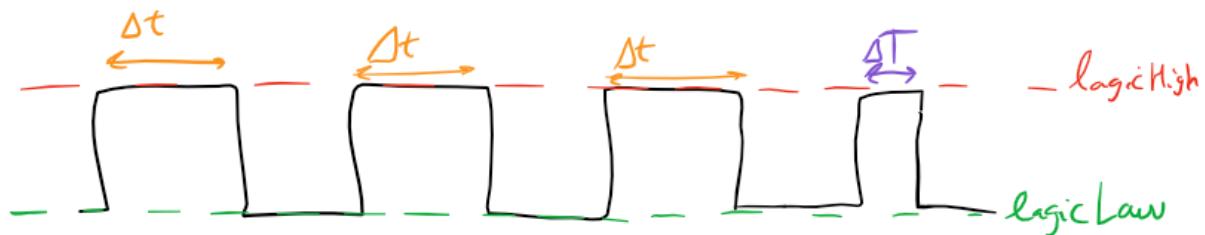
- Bloquer les angles

Considérant la structure de notre système, il est apparu que dans aucun cas d'utilisation « normale » on pourrait avoir besoin d'aller au-delà d'un angle droit. Nous avons donc construit notre programme de manière à ce qu'il limite l'angle mesuré entre  $-90^\circ$  et  $90^\circ$ . Avec une telle restriction nous ne pourrons jamais nous trouver dans le cas de la figure 3 précédente. Cela a de plus des avantages au niveau des calculs qui seront détaillés dans le prochain paragraphe

## VII. Commande des moteurs



Nos ESC (qui envoient des commandes au moteurs) fonctionnent comme la plupart des systèmes en électroniques. Grâce à des impulsions de temps variable (PWM : Pulse Width Modulation). Ces impulsions varient entre 1000-2000us. La pulsation médiane étant 1500us. C'est celle qui stoppe les moteurs.



Nous avons trouvé une librairie, très simple à utiliser, pour envoyer des pulsations aux moteurs : la librairie *Servo*.

Il faut donc inclure la librairie Servo dans notre code Arduino comme ceci :

```
#include <Servo.h>
```

Dans notre cas, chaque moteur va être un objet contenant plusieurs attributs et des fonctions pour les manipuler, son constructeur est Servo.

```
//Création des instances de SERVO  
Servo M_Yaw;  
Servo M_Roll;  
Servo M_Pitch;
```

Dans notre objet, nous avons plusieurs fonctions : celles que nous allons utiliser sont :

- Attach() // Pour renseigner le digital pin du moteur
- writeMicroseconds() // Pour envoyer une pulsation en microsecondes

Nous avons donc à mettre pour indiquer le numéro des pins des moteurs :

Et nous devons indiquer à chaque objet ou moteur leur pin respectif en utilisant objet.attach(pin) :

```
M_Yaw.attach(8); // le fil sur l'arduino est blanc  
M_Pitch.attach(9); // le fil sur l'arduino est bleu  
M_Roll.attach(10); // le fil sur l'arduino est jaune
```

Puis ils suffit d'envoyer les pulsations aux moteurs  
objet.writeMicroseconds(tempms\_microsecondes)

```
M_Yaw.writeMicroseconds(M_yaw_PID);  
M_Roll.writeMicroseconds(M_roll_PID);  
M_Pitch.writeMicroseconds(M_pitch_PID);
```

Il n'est même pas nécessaire de faire des delays pour s'assurer d'envoyer une pulsation toutes les 4ms la librairie étant faite pour contrôler les moteurs, elle sait quand il faut envoyer les pulsations. Cette librairie n'est pas seulement simple à utiliser, elle supprime les pulsations erronées dû aux interférences, c'est-à-dire que si une pulsation est trop éloignée de la précédente, elle l'annule.

## VIII. Pour aller plus loin

Il y a de nombreux ajouts possibles à notre systèmes. Nous en avons étudiés certains, mais par manque de temps nous n'avons pas pu les mener à bien. Voici les résultats de certaines de ces recherches

### 1. Mode de stabilisation sur deux axes uniquement

- Utilité : stabilisation sur tous les axes sauf le lacet (on n'autorise que les rotations autour de l'axe vertical), ce qui permet de filmer des plans panoramiques
- Comment le réaliser : il faudrait désactiver les lignes de code assurant la stabilité selon le lacet.

### 2. Différents taux de réactivité : méthode d'émulation de réponse

- Utilité : lors d'un mouvement de caméra, il est parfois nécessaire de bouger rapidement celle-ci pour souligner l'intensité de l'action. A l'inverse, il peut être nécessaire d'assurer un mouvement relativement lent et très fluide pour traduire un moment paisible. Ces subtilités de réalisation apportent un grand plus dans la mise en place d'un ambiance dans un film, il serait donc apprécié que notre système s'adapte aisément à ces styles de prise d'image.
- Comment le réaliser : dans son état actuel, notre système réagit de manière linéaire aux mouvements. Il faudrait modéliser une réponse créée à partir de différentes fonction : réponse exponentielle lorsque l'on recherche une réponse rapide ou brusque, et une réponse logarithmique lorsque l'on recherche une réponse plus lente ou plus douce.

### 3. Contrôle à distance du stabilisateur

- Utilité : cela permettra la division des tâches, un opérateur s'occupera de manier la caméra (on pourrait même la disposer sur un système automatisé) tandis qu'une seconde personne assure les mouvements corrects de celle-ci (panoramique,travelling...)
- Comment le réaliser : l'arduino pouvant recevoir des consignes d'ondes de fréquence 250Hz correspondant à une télécommande classique, il faudrait qu'elle relaie ces données aux ESC afin qu'ils commandent les moteurs.
- Léger défaut à souligner : le résultat peut voir sa qualité diminuer si les deux opérateurs se coordonnent mal.

## Conclusion

Ce TIPE a mis en lumière différentes facettes du métier d'ingénieur qui s'éloignent du milieu académique dans lequel nous travaillions jusqu'ici. Dans un premier temps il a fallu trouver un projet qui répondait au sujet posé, l'optimalité. En effet le rôle de notre stabilisateur est d'améliorer autant que possible la qualité d'une vidéo. Mais nous souhaitions aussi créer un objet qui répondrait à une problématique et serait compétitif sur le marché, c'est une façon d'aborder un sujet que nous n'avions encore jamais mise en place. Notre deuxième tâche a été de définir un cahier des charges mêlant la faisabilité technique et financière, des recherches sur les innovations de l'électronique ont été nécessaire ainsi que des choix basés sur une analyse comparative des composants que l'on a trouvés ; il s'agit ici aussi d'un aspect du travail d'ingénieur nouveau pour nous. La troisième étape a été la production du support ; il s'agissait d'un travail manuel et de conception requérant une grande rigueur, la moindre imprécision pouvant avoir de lourdes retombées de stabilité, ce qui se rapproche du travail dans un bureau d'étude. Enfin la création de notre code a été pour nous une introduction au domaine des systèmes embarqués.

Au cours de la réalisation de notre système nous avons été confronté à différents problèmes : au cours de la construction de la structure, l'organisation et la division du travail en fonction des compétences de chacun, les erreurs d'étude théorique qu'il a fallu trouver et corriger, n'en sont que trois parmi d'autres. Nous avons dû être capables de réagir rapidement face aux problèmes et de garder la tête froide pour les traiter, même s'ils pouvaient être décourageants ou sembler insurmontables. De plus, il a été régulièrement nécessaire de reconsidérer et modifier les plans de notre système ainsi que les fonctionnalités que l'on souhaitait y intégrer face à des limitations temporelles, financière matérielles ou techniques

Nous avons donc tenté autant que possible d'aborder ce TIPE comme un projet professionnel confié à un bureau d'étude, même si on est encore loin du résultat d'un projet réel : l'ergonomie et le design sont grandement perfectibles ainsi que la rentabilité. En revanche c'est une base sur lequel pourront nous fonder pour nos futurs projets de robotique d'électronique ou même d'informatique. Dans tous les cas ce projet nous a déjà permis d'acquérir des compétences nécessaires au travail d'ingénieur. Nous avons développé nos aptitudes à travailler en groupe. Nous avons mis en pratiques des connaissances fondamentales en physique et en mathématique. Finalement nous avons réussi à construire partant de rien un système abouti.

## Abstract

This project has put many aspects of the work as an engineer in the spotlight, which are different from the scholar work that we used to do so far. First of all, we had to find a project which was an answer to the subject: optimality. Indeed, the purpose of our stabilizer is to improve as much as possible the quality of a footage. Our second task was to define the specifications, mixing the financial and technical possibilities, researches about high technologies were necessary as well, and we made choices based on a comparative analysis of the electronics devices we found; this was another perspective of the engineer's work that was new to us. The third step was the construction of the structure; it was a work of conception and handcraft which required strictness, any imprecision would have harsh effects on the stability. Finally, the creation of our program introduced us to the embedded systems sector.

During the creation of our system, we had to face different problems: while we were building the structure, organization and work division based on the skills of each of us, the mistakes made in the study we had to find and correct, are just three out of many. We had to react quickly in case of problems and remain calm to deal with issues, even if it was discouraging or seemed unresolvable. Furthermore, we often had to reconsider and modify the plans of our system, and the functionalities we wanted to include in it, because of timelines, financial material or technical issues.

We tried as much as possible to manage our TIPE as a professional project led by a design office, even if we are not even close from the result of an actual project: ergonomics and design have a big room for improvement, and so does rentability. In any cases this project already made us acquire engineers skills, we learned teamwork, we have put into practice some of our fundamental physics and mathematics courses, finally starting from scratch we achieved to create a complete product

# Immenses Remerciements

*Thierry, Lucas tiennent encore à remercier vivement :*

**Monsieur Jean-Marie SAINT-JALM,**

*Professeur de Physique à l'ISEP.*

**Monsieur Camille GAUDILLIERE ,**

*Professeur de Science de l'ingénieur à l'ISEP.*

**Monsieur Hervé AFRIAT,**

*Responsable des laboratoires de l'ISEP.*

**Monsieur Vladimir BOYT,**

*Assistant Responsable des laboratoires de  
l'ISEP.*

## L'association Air Isep

*Association de robotique de l'ISEP*

Pour leur soutien et leur aide précieuse dans ce TIPE.

Ainsi que tous ceux qui ont contribué de près ou de loin à la réalisation de ce TIPE.

# Annexe : Sitographie

## Filtrage de Kalman :

Wikipédia, consulté en avril 2017 : [https://en.wikipedia.org/wiki/Kalman\\_filter](https://en.wikipedia.org/wiki/Kalman_filter)

*Kristian Sloth Lauszus : 10 septembre 2012*, consulté en avril 2017 : Guide d'implémentation pratique du filtre de Kalman :

<http://blog.tkjelectronics.dk/2012/09/a-practical-approach-to-kalman-filter-and-how-to-implement-it/>

David Kohanbash , 30 janvier 2014, consulté en Février 2017 :  
<http://robotsforroboticists.com/kalman-filtering/>

## Contrôleur de Stabilisateur Open source :

Brugi Brushless Gimbal , consulté en Novembre 2016 <https://sourceforge.net/projects/brushless-gimbal-brugi/>

## Communication $I^2C$ :

Dejan Nedelkovski : October 5, 2015, consulté en Novembre 2016

<http://howtomechatronics.com/tutorials/arduino/how-i2c-communication-works-and-how-to-use-it-with-arduino/>

## Modification du Firmware des ESC (pistes de travaux).

Forum , Flasher un ESC , OlliW , consulté en Novembre 2016 :

<https://www.rcgroups.com/forums/showthread.php?1858164-Flashing-ESC-for-Brushless-Direct-Drive-Gimbals>

## Composants :

Site d'achat : Voir Détail dans le Plan (cf compte rendu de TIPE).

## Fiche Constructeur ATMEL 2560 (Arduino-Mega) :

ISEP 2016-2017, TIPE sur la stabilisation,  
Thierry LINCOLN & LUCAS D'OLIVEIRA

Fiche constructeur +450 pages : Mais crucial, consulté en Janvier 2017 :  
[http://www.atmel.com/Images/Atmel-2549-8-bit-AVR-Microcontroller-ATmega640-1280-1281-2560-2561\\_datasheet.pdf](http://www.atmel.com/Images/Atmel-2549-8-bit-AVR-Microcontroller-ATmega640-1280-1281-2560-2561_datasheet.pdf)

## **Fiche Constructeur MPU-6050 (Gyro-Accé) :**

Fiche Constructeur +40 Pages :

Documentation officielle Invensense Gyroscope , consulté en Janvier 2017:  
<https://www.invensense.com/products/motion-tracking/6-axis/mpu-6050/>  
<https://store.invensense.com/Datasheets/invensense/RM-MPU-6000A.pdf>

NXP, document sur les calculs d'angle à partir d'un accéléromètre, consulté en Février 2017:  
<http://www.nxp.com/assets/documents/data/en/application-notes/AN3461.pdf>

Implementing a Tilt-Compensated eCompass, Talat Ozyagcilar, consulté en Avril 2017 :  
[https://cache.freescale.com/files/sensors/doc/app\\_note/AN4248.pdf](https://cache.freescale.com/files/sensors/doc/app_note/AN4248.pdf)

Matrice de rotation, Wikipédia, consulté en Février 2017 :  
[https://fr.wikipedia.org/wiki/Matrice\\_de\\_rotation](https://fr.wikipedia.org/wiki/Matrice_de_rotation)

Angle d'Euler, Wikipédia, consulté en Février 2017 :  
[https://en.wikipedia.org/wiki/Euler\\_angles](https://en.wikipedia.org/wiki/Euler_angles)

Atan2(), Wikipédia , consulté en Février 2017 :  
<https://fr.wikipedia.org/wiki/Atan2>

## **ESCs**

(contrôleur électronique de vitesse).

Logiciel pour le flash , consulté en Janvier 2017:  
<http://lazyzero.de/en/modellbau/kkmulticopterflashtool>

Firmware original des ESCs , consulté en Janvier 2017: <https://github.com/sim-/tgy>

Firmware modifié pour rendre les ESC bidirectionnel , consulté en Janvier 2017:  
<https://www.rcgroups.com/forums/showthread.php?1849952-SimonK-Reversible-ESC-Think-3D-QuadCopter>

BGMC ESC , contrôleur électronique de vitesse, Luke Whittaker, consulté en Février 2017 :  
<http://rovertec.com/products-bgmc2.html>

## **Électronique :**

Output Protection ; forum , consulté en Mai 2017 :

<http://forum.arduino.cc/index.php?topic=139558.0>

ISEP 2016-2017, TIPE sur la stabilisation,  
Thierry LINCOLN & LUCAS D'OLIVEIRA

Diode de Schottky : Wikipédia , consulté en Mai 2017 :

[https://en.wikipedia.org/wiki/Schottky\\_diode](https://en.wikipedia.org/wiki/Schottky_diode)

## Autre :

Réalisation d'une sitographie, lewebpédagogique , consulté en Mai 2017 :

[http://lewebpedagogique.com/hgplateau/files/2012/10/Realiser\\_une\\_bibliographie.pdf](http://lewebpedagogique.com/hgplateau/files/2012/10/Realiser_une_bibliographie.pdf)

01net.com , Information sur la stabilisation , consulté en Octobre 2016 :

<http://www.01net.com/astuces/stabilisation-optique-mecanique-ou-numerique-quelles-differences-634788.html>

# Annexe : Code

---

```

1 #include<Wire.h>
2 #include<Servo.h>
3 #include<EEPROM.h>
4 #include "Kalman_Lib.h"
5
6 //Création des instances de Kalman//
7 Kalman kalmanX;
8 Kalman kalmanY;
9 Kalman kalmanZ;
10
11 //Création des instances de SERVO
12 Servo M_Yaw;
13 Servo M_Roll;
14 Servo M_Pitch;
15
16 //##### All the MPU 6050 STUFF #####
17 const int gyro_address = 0x68;
18 int iteratif_1, iteratif_2;
19
20 #define LED_PIN 13//(Arduino.is.13)
21 bool blinkState = false;
22
23 float gyro_roll_cal;
24 float gyro_pitch_cal;
25 float gyro_yaw_cal;
26 float acc_x_cal, acc_y_cal, acc_z_cal;
27
28 float gyro_roll_d, gyro_pitch_d, gyro_yaw_d;
29 float gyro_roll_d_cal, gyro_pitch_d_cal, gyro_yaw_d_cal;
30 float gyro_roll_raw, gyro_pitch_raw, gyro_yaw_raw;
31 float acc_x_raw, acc_y_raw, acc_z_raw, temperature;
32
33 float yaw_angle, pitch_angle, roll_angle;
34 float pitch_angle_kal, roll_angle_kal, yaw_angle_kal;
35 float pitch_level_adjust, roll_level_adjust, yaw_level_adjust;
36
37
38 ##### gains du PID #####
39
40 //gains pour le roulis:
41 float P_gain_roll = 1.0;
42 float I_gain_roll = 0.0;
43 float D_gain_roll = 0;
44 int max_roll = 400;
45
46 //gains pour le tangage:
47 float P_gain_pitch = 1.0;
48 float I_gain_pitch = 0.0;
49 float D_gain_pitch = 0;
50 int max_pitch = 400;
51
52 //gains pour le lacet:
53 float P_gain_yaw = 1.0;
54 float I_gain_yaw = 0.0;
55 float D_gain_yaw = 0.0;
56 int max_yaw = 400;
57
58 ##### variables du PID #####
59
60
61 float roll_error, pitch_error, yaw_error;
62 //valeurs du PID pour le roulis:

```

---

---

```

63 float·val_P_roll;
64 float·val_I_roll;
65 float·val_D_roll;
66 float·val_I_roll_pcdt·=·0.0;
67 float·roll_erreur_pcdt·=·0.0;
68 float·val_correction_roll;
69 float·M_roll_PID·;·
70
71
72 //valeurs·du·PID·pour·le·tangage:
73 float·val_P_pitch;
74 float·val_I_pitch;
75 float·val_D_pitch;
76 float·val_I_pitch_pcdt·=·0.0;
77 float·pitch_erreur_pcdt·=·0.0;
78 float·val_correction_pitch;
79 float·M_pitch_PID·;··
80
81 //valeurs·du·PID·pour·le·lacet:
82 float·val_P_yaw;
83 float·val_I_yaw;
84 float·val_D_yaw;
85 float·val_I_yaw_pcdt·=·0.0;
86 float·yaw_erreur_pcdt·=·0.0;
87 float·val_correction_yaw;
88 float·M_yaw_PID·;·
89
90 #####Variables·EEPROM·mémoire#####
91 int·address·=·1;
92 int·address1·=·2·;
93 int·address2,address3,address4,address5,address6;
94
95 int·nbr_buzz,·duree_buzz,·interval_buzz,·i·;·
96
97
98 int·battery_voltage;
99 int·joy_roll_raw,joy_pitch_raw,joy_yaw_raw;
100 int·joy_roll_pulse,joy_pitch_pulse,joy_yaw_pulse;
101 int·joy_roll_pid,·joy_pitch_pid,·joy_yaw_pid·;·
102 int···commandex;
103 bool·button_1·==·true·;·
104
105 //variables·de·temps·
106 unsigned·long·timer;
107 double·dt;
108
109 //=====
110 //-----INITIAL·SETUP-----
111 //=====
112
113
114 void·setup(){
115     ·Wire.begin();
116     ·//·Vérification·de·la·bonne·fréquence·I2C·
117     ·#if·ARDUINO·>·157
118     ··Wire.setClock(400000UL);·//·Set·I2C·frequency·to·400kHz
119     ·#else
120     ··TWBR·=·((F_CPU·/·400000UL)··16)··2;··//·Set·I2C·frequency·to·400kHz
121     ·#endif
122     ···
123     ··gyro_init();··//·initialise·le·MPU6050·
124     ··magn_init();··//·initialise·le·Magnétomètre··

```

---

---

```

125 ....Serial.begin(9600);
126 ....pinMode(13,OUTPUT);///·configuration·de·la·led·informative
127 ....
128 ....M_Yaw.attach(8);·//·le·fil·sur·l'arduino·est·blanc·
129 ....M_Pitch.attach(9);·//·le·fil·sur·l'arduino·est·bleu
130 ....M_Roll.attach(10);·//·le·fil·sur·l'arduino·est·jaune
131 ....
132 ....delay(50);
133 ...//·Condition··de·démarage·de·la·calibration···
134 ....buzzer_je_buzz(2,50,30);
135
136 ....commandex==1500;·
137 ....
138 ....joy_roll_raw==analogRead(0);
139 ....delay(50);
140 ....
141
142 ....if·(EEPROM.read(address)!=1)calibration_gyro();
143 ....else·if·(joy_roll_raw>990&&EEPROM.read(address)==1)·calibration_gyro();
144 ....else{
145 .....gyro_roll_cal==EEPROM.read(address1);
146 .....gyro_pitch_cal==EEPROM.read(address2);
147 .....gyro_yaw_cal==EEPROM.read(address3);
148 .....acc_x_cal==EEPROM.read(address4);
149 .....acc_y_cal==EEPROM.read(address5);
150 .....acc_z_cal==EEPROM.read(address6);
151 ....}·
152 ...
153 .....//https://en.wikipedia.org/wiki/Atan2·pour·l'explication·des·opérateurs·utilisés·pour·la·suite
154 ....gyro_signal();
155 ....magn_signal();
156 ....double·roll_angle==atan2(acc_y_raw,acc_z_raw)*·57.2951;
157 ....double·pitch_angle==atan(-acc_x_raw/sqrt(acc_y_raw*acc_y_raw+acc_z_raw*acc_z_raw))*·57.2951;
158 ....kalmanX.setAngle(roll_angle);·//·définition·des·angles·de·départ
159 ....kalmanY.setAngle(pitch_angle);
160 ....kalmanZ.setAngle(yaw_angle);
161 ...
162 ...
163 ....timer==micros();·//Retourne·le·nombre·de·microsecondes·écoulées·depuis·que·l'arduino·a·commencé·à·lire··le·code···
164
165 ...//notre·batterie·est·à·12,6V·max,·ce·qui·correspond·à·1023·en·analogRead(0).
166 ...battery_voltage==(analogRead(0)+65)*1.2317;·//·Pour·établir·une·proportionnalité·entre·la·valeur·lue··avec·analogRead·et·la·tension·en·Volts.
167 .....
168 }
169
170 void·loop()·{
171 ...double·dt==·(double)(micros()--·timer)·/·1000000;·//·Calcul·de·delta·t
172 ...timer==micros();
173 ...gyro_signal();
174
175 ...gyro_roll_d==rad_to_deg(gyro_roll_raw);///·converti·en·deg/sec·les·valeurs·non·calibrer··
176 ...gyro_pitch_d==rad_to_deg(gyro_pitch_raw);
177 ...gyro_yaw_d==rad_to_deg(gyro_yaw_raw);
178
179 ...gyro_roll_d_cal==rad_to_deg(gyro_roll_raw);·//·convertir·les·valeurs·calibrer·en·deg/sec
180 ...gyro_pitch_d_cal==rad_to_deg(gyro_pitch_raw);
181 ...gyro_yaw_d_cal==rad_to_deg(gyro_yaw_raw)·;
182
183 ...
184 ...double·roll_angle==atan2(acc_y_raw,acc_z_raw)*·57.2951;·//on·calcule·la·valeur·de·l'angle·en·fonction··

```

---

---

```

d'informations·donnée·en·repère·cartésien
185 ..double·pitch_angle==atan(-acc_x_raw/.sqrt(acc_y_raw*acc_y_raw+acc_z_raw*acc_z_raw))*57.2951;
186 ..//·Ceci·résout·le·problème·de·la·transition·de·l'angle·de·l'accéléromètre·entre·-pi·et·pi·
187 ..if((roll_angle<-90&&roll_angle_kal>90)|| (roll_angle>90&&roll_angle_kal<-90)){{
188 ....kalmanX.setAngle(roll_angle);
189 ....roll_angle_kal=roll_angle;
190 ....}
191 ..else....roll_angle_kal=kalmanX.getAngle(roll_angle,gyro_roll_d,dt); //·Calculate·the·angle·using·a· Kalman·filter
192
193 ..if(abs(roll_angle_kal)>90)gyro_pitch_d=-gyro_pitch_d; //·Inverse·le·signal·pour·avoir·rester·dans·l' interval-·\pi/2·\pi/2·
194 ..pitch_angle_kal=kalmanY.getAngle(pitch_angle,gyro_pitch_d,dt);
195 ..
196 ..yaw_angle_kal=kalmanZ.getAngle(yaw_angle,gyro_yaw_d,dt);·
197
198 //·Détermination·des·correction·angulaire·à·apporter·sur·les·moteurs..·On·refait·la·proportion·180·degré· vers·des·pulsation·1000-1500·et·1500-2000·en·multipliant·par·15..
199 ..roll_level_adjust=roll_angle_kal*15;
200 ..pitch_level_adjust=pitch_angle_kal*15;
201 ..yaw_level_adjust=yaw_angle*15;·
202 ..
203 ..val_I_roll=0;
204 ..roll_erreur_pc dt=0;
205 ..val_I_pitch=0;
206 ..pitch_erreur_pc dt=0;
207 ..val_I_yaw=0;
208 ..yaw_erreur_pc dt=0;
209
210 ...PID();
211 ...
212 ..int start=2;
213 ..int throttle=1500;
214 ..if(start==2){
215 ....
216 ....M_yaw_PID=throttle+val_correction_yaw;
217 ....M_roll_PID=throttle+val_correction_roll;
218 ....M_pitch_PID=throttle+val_correction_pitch;
219 //·limite·la·valeur·maximal·pour·ne·pas·décalibrer·les·ESC·
220 ....if(M_yaw_PID>2000)M_yaw_PID=2000;
221 ....if(M_roll_PID>2000)M_roll_PID=2000;
222 ....if(M_pitch_PID>2000)M_pitch_PID=2000;
223 //·Limite·la·valeur·minimal·pour·ne·pas·décalibrer·les·ESC·
224 ....if(M_yaw_PID<1000)M_yaw_PID=1000;
225 ....if(M_roll_PID<1000)M_roll_PID=1000;
226 ....if(M_pitch_PID<1000)M_pitch_PID=1000;
227 ....
228 ....
229 ...
230 ...
231 ..else{
232 ....//·Si·tous·va·mal·,·on·envoi·la·pulsion·médiane·dans·ESC..
233 ....M_yaw_PID=throttle;
234 ....M_roll_PID=throttle;
235 ....M_pitch_PID=throttle;
236 ...
237 ...
238 ...
239 ...//·Envie·les·corrections·au·moteurs...
240 ...M_Yaw.writeMicroseconds(M_yaw_PID);
241 ...M_Roll.writeMicroseconds(M_roll_PID);
242 ...M_Pitch.writeMicroseconds(M_pitch_PID);

```

---

---

```

243
244
245 .../////////////////////////////•Printing section for essential values/////////////////
246
247 ...Serial.print("••PID•tangage:");Serial.print(val_correction_pitch);Serial.print("••PID•roulis:");Serial. ↵
    print(val_correction_roll);Serial.print("••PID•lacet:");Serial.println(val_correction_yaw);
248 ...//Serial.print("Roll:::"),Serial.print(gyro_roll_d),Serial.print("Pitch:::"),Serial.print(gyro_pitch_d), ↵
    Serial.print("Yaw:::"),Serial.println(gyro_yaw_d);
249 ...//Serial.print("Roll_angle_K:::"),Serial.print(roll_angle_kal),Serial.print("Pitch_angle_k:::"),Serial. ↵
    print(pitch_angle_kal),Serial.print("Yaw_angle:::"),Serial.println(yaw_angle_kal);
250 ...//Serial.print("Roll_angle:::"),Serial.print(roll_angle),Serial.print("Pitch_angle:::"),Serial.print( ↵
    pitch_angle),Serial.print("Yaw_angle:::"),Serial.println(yaw_angle);
251 ...//Serial.print("Roll_joy:::"),Serial.print(joy_roll_pid),Serial.print("Pitch_Joy:::"),Serial.println( ↵
    joy_pitch_pid);
252 }
253
254 //•Fonction qui converti en deg/s
255 float rad_to_deg(float angle){
256 ...float angle_c=((angle_c)*0.7+((angle/57.14286)*0.3);
257 ...return angle_c;
258 }
259
260 void PID()
261 {
262 ...//calcul des différentes erreurs:
263 ...roll_erreur=gyro_roll_d_cal;//
264 ...pitch_erreur=gyro_pitch_cal;//
265 ...yaw_erreur=gyro_yaw_d_cal;//
266
267
268 ...//calcul de la correction à apporter au roulis:
269 ...val_P_roll+=roll_erreur*p_gain_roll;
270 ...val_I_roll+=val_I_roll_pcdt+roll_erreur*I_gain_roll+roll_level_adjust;
271 ...val_D_roll+=(roll_erreur-roll_erreur_pcdt)*D_gain_roll;
272
273 ...if(val_I_roll>max_roll)
274 ...{
275 ...    val_I_roll=max_roll;
276 ...}
277 ...else if(val_I_roll<-max_roll*-1)
278 ...{
279 ...    val_I_roll=-max_roll*-1;
280 ...}
281
282 ...val_correction_roll+=val_P_roll+val_I_roll+val_D_roll;
283 ...if(val_correction_roll>max_roll)
284 ...{
285 ...    val_correction_roll=max_roll;
286 ...}
287 ...else if(val_correction_roll<-1*max_roll)
288 ...{
289 ...    val_correction_roll=-1*max_roll;
290 ...}
291
292 ...val_I_roll_pcdt=val_I_roll;
293 ...roll_erreur_pcdt=roll_erreur;
294
295 ...//calcul de la correction à apporter au tangage:
296 ...val_P_pitch+=pitch_erreur*p_gain_pitch;
297 ...val_I_pitch+=val_I_pitch_pcdt+pitch_erreur*I_gain_pitch+pitch_level_adjust;
298 ...val_D_pitch+=(pitch_erreur-pitch_erreur_pcdt)*D_gain_pitch;
299

```

---

---

```

300 ...if(val_I_pitch>max_pitch)
301 ...
302 ...val_I_pitch-=max_pitch;
303 ...
304 ...else if(val_I_pitch<max_pitch*-1)
305 ...
306 ...val_I_pitch=-max_pitch*;-1;
307 ...
308
309 ...val_correction_pitch-=val_P_pitch++val_I_pitch++val_D_pitch;
310 ...if(val_correction_pitch>max_pitch)
311 ...
312 ...val_correction_pitch-=max_pitch;
313 ...
314 ...else if(val_correction_pitch<-1*max_pitch)
315 ...
316 ...val_correction_pitch=-1*max_pitch;
317 ...
318 ...
319 ...val_I_pitch_pcdt-=val_I_pitch;
320 ...pitch_erreur_pcdt+=pitch_erreur;
321 ...
322 ...//calcul de la correction à apporter au lacet:
323 ...val_P_yaw-=yaw_erreur*P_gain_yaw;
324 ...val_I_yaw-=val_I_yaw_pcdt++yaw_erreur*I_gain_yaw++yaw_level_adjust;
325 ...val_D_yaw=-(yaw_erreur--yaw_erreur_pcdt)*D_gain_yaw;
326
327 ...if(val_I_yaw>max_yaw)
328 ...
329 ...val_I_yaw=-max_yaw;
330 ...
331 ...else if(val_I_yaw<-max_yaw*;-1)
332 ...
333 ...val_I_yaw=-max_yaw*;-1;
334 ...
335
336 ...val_correction_yaw-=val_P_yaw++val_I_yaw++val_D_yaw;
337 ...if(val_correction_yaw>max_yaw)
338 ...
339 ...val_correction_yaw=-max_yaw;
340 ...
341 ...else if(val_correction_yaw<-1*max_yaw)
342 ...
343 ...val_correction_yaw=-1*max_yaw;
344 ...
345
346 ...val_I_yaw_pcdt-=val_I_yaw;
347 ...yaw_erreur_pcdt+=yaw_erreur;
348 }
349 //##### Fonction simple du gyro #####
350 void gyro_init(){
351 ...Wire.begin();
352 ...Wire.beginTransmission(gyro_address);.....//Start communication with the address found during search.
353 ...Wire.write(0x6B);.....//We want to write to the PWR_MGMT_1 register (6B hex)
354 ...Wire.write(0x00);.....//Set the register bits as 00000000 to activate the gyro
355 ...Wire.endTransmission();.....//End the transmission with the gyro.
356
357 ...Wire.beginTransmission(gyro_address);.....//Start communication with

```

---

---

```

    the.address.found.during.search.
358     ...Wire.write(0x1B);.....//We want to write to the..... ↵
    GYRO_CONFIG.register.(1B.hex)
359     ...Wire.write(0x08);.....//Set the register bits as..... ↵
    00001000.(500dps.full.scale)
360     ...Wire.endTransmission();.....//End the transmission with..... ↵
    the.gyro
361
362     ...Wire.beginTransmission(gyro_address);.....//Start communication with..... ↵
    the.address.found.during.search.
363     ...Wire.write(0x1C);.....//We want to write to the..... ↵
    ACCEL_CONFIG.register.(1A.hex)
364     ...Wire.write(0x00);.....//Set the register bits as..... ↵
    00010000.(+/-8g.full.scale.range)
365     ...Wire.endTransmission();.....//End the transmission with..... ↵
    the.gyro
366 }
367 void.magn_init(){
368     ...Wire.beginTransmission(0x1E); //start talking
369     ...Wire.write(0x02); //Set the Register
370     ...Wire.write(0x00); //Tell the HMC5883 to Continuously Measure
371     ...Wire.endTransmission();
372 }
373 void.magn_signal(){
374     //Tell the HMC what register to begin writing data into
375     ...Wire.beginTransmission(0x1E);
376     ...Wire.write(0x03); //start with register 3.
377     ...Wire.endTransmission();
378     int magn_x,magn_y,magn_z;
379     ...Wire.requestFrom(0x1E,6);
380     if(6<=Wire.available()){
381         magn_x = Wire.read()<<8|Wire.read();
382         magn_z = Wire.read()<<8|Wire.read();
383         magn_y = Wire.read()<<8|Wire.read();
384     }
385     float heading = 0.8*heading + 0.2*atan2(magn_y,magn_x);
386     float yaw_angle = heading * 180/3.14159265358979323846264338327950288; //converti en deg
387 }
388 }
389 void.calibration_gyro(){
390     digitalWrite(13,HIGH);
391     delay(5);
392     //Calibration
393     for (iteratif_1=0; iteratif_1<3000; iteratif_1++){ //Accumule 3000 valeurs pour la calibration
394         if(iteratif_1%15==0)digitalWrite(13,!digitalRead(13)); //Fait clignoter la LED pour indiquer la calibration.
395         gyro_signal();
396         //Appelle la fonction gyroscope
397         gyro_roll_cal+=gyro_roll_raw; //Fait la somme sur 3000 valeurs sur tous les axes.
398         gyro_pitch_cal+=gyro_pitch_raw;
399         gyro_yaw_cal+=gyro_yaw_raw;
400         acc_x_cal+=acc_x_raw;
401         acc_y_cal+=acc_y_raw;
402         acc_z_cal+=acc_z_raw;
403         M_Yaw.writeMicroseconds(400);
404         M_Roll.writeMicroseconds(400);
405         M_Pitch.writeMicroseconds(400);
406         .....//pour eviter une decalibration des ESC on envoi encore une pulsation de 1000us.
407         delay(1);
408     }

```

---

---

```

409     ....}.*;
410     .....//EEPROM.put(address,int(1));//.met le bouléen mémoire à 1...
411     .....//Maintenant on a obtenu 3000 valeurs on peut faire la moyenne.
412     .....gyro_roll_cal/=3000;.....//Divise le total par 3000.
413     .....gyro_pitch_cal/=3000;
414     .....gyro_yaw_cal/=3000;
415     .....acc_x_cal/=3000;
416     .....acc_y_cal/=3000;
417     .....acc_z_cal/=3000;...
418     .....EEPROM.put(address1,gyro_roll_cal);
419     .....address2=address1+sizeof(float)(gyro_roll_cal));
420     .....EEPROM.put(address2,gyro_pitch_cal);
421     .....address3=address2+sizeof(float)(gyro_pitch_cal));
422     .....EEPROM.put(address3,gyro_yaw_cal);
423     .....address4=address3+sizeof(float)(gyro_yaw_cal));
424     .....EEPROM.put(address4,acc_x_cal);
425     .....address5=address4+sizeof(float)(acc_x_cal));
426     .....EEPROM.put(address5,acc_y_cal);
427     .....address6=address5+sizeof(float)(acc_y_cal));
428     .....EEPROM.put(address6,acc_z_cal);*/
429     ....
430     .....digitalWrite(13,LOW);
431     .....Serial.println("Calibration gyro sucess");
432     .....Serial.print(gyro_roll_cal),Serial.print(gyro_pitch_cal),Serial.println(gyro_yaw_cal);
433
434 }
435
436 void gyro_signal()
437 {
438     .....Wire.beginTransmission(gyro_address);.....//Start communication with the gyro.
439     .....Wire.write(0x3B);.....//Start reading @ register 43h and auto increment with every read.
440     .....Wire.endTransmission();.....//End the transmission.
441     .....Wire.requestFrom(gyro_address,14);.....//Request 14 bytes from the gyro.
442     ...
443     ...
444     .....while(Wire.available()<14);.....//Wait until the 14 bytes are received.
445     .....acc_x_raw=Wire.read()<<8|Wire.read();.....//Add the low and high byte to the acc_x variable.
446     .....acc_y_raw=Wire.read()<<8|Wire.read();.....//Add the low and high byte to the acc_y variable.
447     .....acc_z_raw=Wire.read()<<8|Wire.read();.....//Add the low and high byte to the acc_z variable.
448     .....temperature=Wire.read()<<8|Wire.read();.....//Add the low and high byte to the temperature variable.
449     .....gyro_roll_raw=Wire.read()<<8|Wire.read();.....//Read high and low part of the angular data.
450     .....gyro_pitch_raw=Wire.read()<<8|Wire.read();.....//Read high and low part of the angular data.
451     .....gyro_yaw_raw=Wire.read()<<8|Wire.read();.....//Read high and low part of the angular data.
452
453     .....if(iteratif_1==3000){
454         .....float acc_x_raw_cal=acc_x_raw--acc_x_cal;
455         .....float acc_y_raw_cal=acc_y_raw--acc_y_cal;
456         .....float acc_z_raw_cal=acc_z_raw_cal--acc_z_cal;
457
458         .....float gyro_roll_raw_cal=gyro_roll_raw--gyro_roll_cal;
459         .....float gyro_pitch_raw_cal=gyro_pitch_raw--gyro_pitch_cal;

```

---

---

```
460 .....float gyro_yaw_raw_cal=gyro_yaw_raw--gyro_yaw_cal;..
461 ....}..
462 ..
463 //Fonction buzzer prend en argument :: (nbr de buzz , longueur du buzz , temps entre buzz )..
464 int buzzer_je_buzz(int(nbr_buzz),int(duree_buzz),int(interval_buzz)) {
465 ...i=1;
466 ...while(i<=nbr_buzz){
467 ....digitalWrite(13,HIGH);
468 ....delay(duree_buzz);
469 ....digitalWrite(13,LOW);
470 ....delay(interval_buzz);
471 ....i+=1;
472 ...
473 }
474
475 int Universal_joystick(int(pulse)){
476 ....if(pulse<470){
477 ....if(commandex>1100){
478 ....commandex-=2;
479 ....}
480 ....else(commandex=1100);
481 ...
482 ...else if(pulse>550){
483 ....if(commandex<1900){
484 ....commandex+=2;
485 ....}
486 ....else(commandex=1900);
487 ...
488 ...
489 ...return commandex;
490 ...
491 ...
492 }
493 ...
494 ...
495
```

# Annexe : Code Kalman

```
...abilisateur\Code_Programmation\Kalman_filter\Kalman.Lib.h  
1 ifndef _Kalman_Lib_h_  
2 define _Kalman_Lib_h_  
3  
4 class Kalman {  
5 public:  
6     Kalman();  
7     float getAngle(float acc_angle, float gyro_speed, float dt);  
8     void setAngle(float angle);  
9  
10 private:  
11     float Q_angle; //erreur angle de l'accéléromètre  
12     float Q_error_s; // erreur vitesse angulaire gyroscope  
13     float R_measure; // erreur sur la mesure des erreurs  
14  
15     float angle_after_kalman; // donc l'angle théta final  
16     float speed_after_kalman; // vitesse kalman théta\dot (pour la dérive)  
17     float speed_after_kalman_corrected; // Vitesse de kalman théta\dot (sans  
18         dérive)  
19     float P[2][2]; // matrice de covariance \in\scriptM_2(\doubleR)  
20 };  
21  
22 endif
```

```

...ilisateur\Code_Programmation\Kalman_filter\Kalman.Lib.cpp
1 #include "Kalman.Lib.h"
2
3
4 Kalman::Kalman() {
5     Q_angle = 0.001f; // Erreur en position.
6     Q_error_s = 0.003f; // Erreur en vitesse.
7     R_measure = 0.03f; // Variance(v)
8
9     angle_after_kalman = 0.0f; // remet à zéro l'angle
10    speed_after_kalman = 0.0f; // remet à zéro l'erreur
11
12    // Ceci est la matrice de covariance d'erreur// à t=0 , matrice==0
13    P[0][0] = 0.0f;
14    P[0][1] = 0.0f;
15    P[1][0] = 0.0f;
16    P[1][1] = 0.0f;
17
18}
19
20 float Kalman::getAngle(float newAngle, float newRate, float dt) {
21
22    //Étape-1// on calcul l'angle avec (\theta avec dérive)
23    speed_after_kalman_corrected = newRate - speed_after_kalman;
24    angle_after_kalman += speed_after_kalman_corrected* dt; // ici on voit      ↵
25        l'intégration
26
27    //Étape-2 // Mise a jour de la matrice de covariance
28
29    P[0][0] += dt*(dt*(P[1][1]) - P[1][0] - P[0][1] + Q_angle);
30    P[0][1] -= dt*P[1][1];
31    P[1][0] -= dt*P[1][1];
32    P[1][1] += Q_error_s * dt;
33
34    //Étape-4// Calcul du scalaire S
35
36    float S = P[0][0] + R_measure; // erreur totale (Estimation + Mesuré)
37
38    //Étape-5// Calcul du vecteur gain de kalman
39    float K[2];
40    K[0] = (P[0][0]) / S;
41    K[1] = (P[0][1]) / S;
42
43    //Étape-3// On calcule l'angle \theta et le \theta\dot dérive.
44    float y = newAngle - angle_after_kalman; // calcul de la variable      ↵
45        d'innovation.
46
47    //Étape-6// on calcule l'angle avec une proportionnalite de kalman
48    angle_after_kalman += K[0] * y;

```

```

...ilisateur\Code_Programmation\Kalman_filter\Kalman.Lib.cpp 2
48     speed_after_kalman += K[1] * y;
49
50     //Etape-7// calcul de la matrice de covariance a posteriori.
51     float P00_temp = P[0][0];
52     float P01_temp = P[0][1];
53
54     P[0][0] -= K[0] * P00_temp;
55     P[0][1] -= K[0] * P01_temp;
56     P[1][0] -= K[1] * P00_temp;
57     P[1][1] -= K[1] * P01_temp;
58
59     return angle_after_kalman; // on retourne l'angle que l'on veut.
60
61 };
62 // pointeur qui permet de definir l'angle de depart.
63 void Kalman::setAngle(float angle_after_kalman) { this->angle_after_kalman = ↵
    angle_after_kalman; }
64
65

```