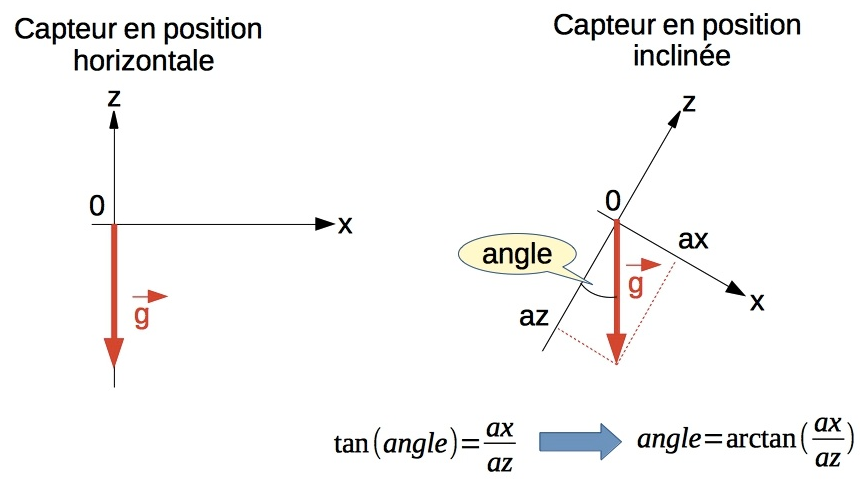
Contrôleur de vol de drone DIY

[1. Les modes de vol 2](#_Toc486693384)

[2. Filtre complémentaire 4](#_Toc486693385)

[1.1 Roulis (calculé avec l’accélération) 4](#_Toc486693386)

[1.1 Tangage (calculé avec l’accélération) 4](#_Toc486693387)

[ 5](#_Toc486693388)

[3. Stabilisation gyroscopique 5](#_Toc486693389)

[1.2 Réglage PID 6](#_Toc486693390)

[4. Fusion de données 7](#_Toc486693391)

[1.3 Filtre complémentaire 7](#_Toc486693392)

[1.4 Filtre de Kalman 8](#_Toc486693393)

[5. Configuration en “Quad +” 8](#_Toc486693394)

[6. Configuration en “Quad X” 9](#_Toc486693395)

[7. Asservissement, mode “ANGLE” 9](#_Toc486693396)

[1.5 Roulis tangage 10](#_Toc486693397)

[8. Arduino UNO rev 3 10](#_Toc486693398)

[9. ESC 10](#_Toc486693399)

[1.6 Génération PWM à 400Hz 10](#_Toc486693400)

[10. Piloter avec une télécommande RC 2.4Ghz 11](#_Toc486693401)

[1.7 Failsafe 11](#_Toc486693402)

[11. Code « CodeDroneDIY » 11](#_Toc486693403)

[1.8 Connections 11](#_Toc486693404)

[1.9 Machine à états 11](#_Toc486693405)

[12. Châssis 12](#_Toc486693406)

[13. Configuration retenue 13](#_Toc486693407)

[Diatone Q450 Quad 450 V3 13](#_Toc486693408)

[14. Bibliographie 13](#_Toc486693409)

[1.10 Arduino 13](#_Toc486693410)

[1.11 PWM 13](#_Toc486693411)

[1.12 PPM 13](#_Toc486693412)

[1.13 PID 13](#_Toc486693413)

[1.14 Drone 13](#_Toc486693414)

[1.15 Traitement de données 13](#_Toc486693415)

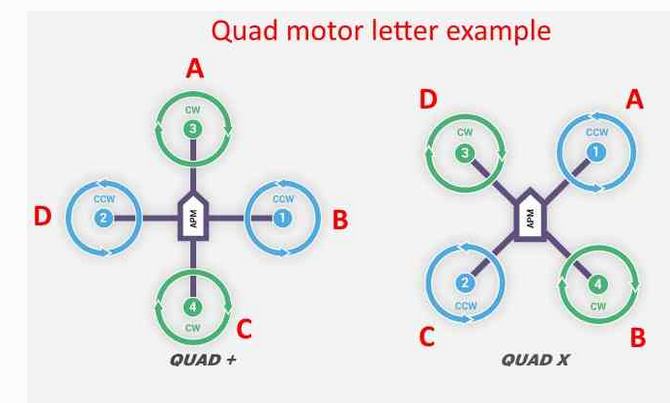
[1.16 Fusion de données 13](#_Toc486693416)

[1.17 Programme python 13](#_Toc486693417)

[1.18 Vibrations 13](#_Toc486693418)

# Les modes de vol

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Mode** | Gyro | Accé | Baro | Bouss | GPS | Rqes |
| ACRO/Gyroscope uniquement | X |  |  |  |  | Un mode généralement par défaut et son vol plus « acrobatique » (le [drone](http://www.robotshop.com/eu/fr/kits-uav-drone.html) ne peut faire de la mise à niveau automatique) |
| ANGLE (Stable/Niveau/Accéléromètre) | X | X |  |  |  | Mode stable ; va essayer de maintenir le niveau du modèle par rapport au sol (mais pas à une position fixe). |
| HORIZON | X |  |  |  |  | Combine l’effet stable avec des commandes et des acrobaties RC lentes et avec des commandes RC rapides. |
| BARO (Maintien de l’altitude) | X | X | X |  |  | Le baromètre est utilisé afin de conserver une certaine hauteur (fixée) lorsqu’aucune autre commande n’est reçue. |
| MAG (Tenue du cap) | X | X |  | X |  | Mode verrouillage de cap (direction à la boussole), pour essayer de maintenir son orientation en lacet. |
| HEADFREE (CareFree/ orientation indépendante du déplacement) | X | X |  | X |  | Maintient l’orientation (lacet) du drone et se déplace toujours dans la même direction 2D pour le même mouvement du manche en ROULIS/TANGAGE. |
| GPS/ Retour à la base |  | X |  | X | X | Utilise automatiquement une boussole et un GPS pour rentrer à la base, au point de départ GPS. |
| GPS/ Points de passage |  | X |  | X | X | Suit automatiquement les points de cheminement GPS pré-configurés de manière autonome. |
| GPS/ Maintien de position |  | X |  | X | X | Maintient la position actuelle en utilisant le GPS et le baromètre (si disponible). |

Connaître la position

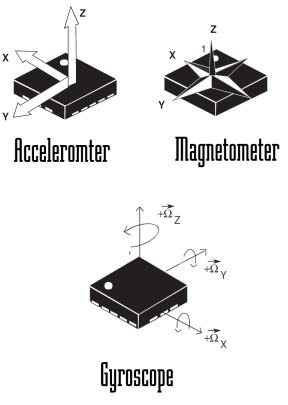
Il faut un gyroscope 3 axes et un accéléromètre 3 axes. Le gyroscope détecte la vitesse angulaire, l’accéléromètre mesure l’accélération sur chaque axe.

Une IMU est une centrale à inertie qui combine un accéléromètre et un gyroscope.

Exemples :

* IMU 6050 invensense
* Le MPU 9250 combine le MPU 6500 (gyroscope + accéléromètre) et l’AK8963 (compas 3 axes)

Le gyroscope dérive dans le temps, et l’accéléromètre est sensible aux vibrations (bruité). L’orientation ne peut pas être calculée à partir des données du gyroscope seul ou de l’accéléromètre seul. La solution est de combiner les données des 2 capteurs avec un filtre de Kalman.



Etape 1 : lire via l’I2C les données brutes d’accélération et de vitesse de rotation.

Etape 2 : Calculer l’orientation

# Filtre complémentaire

## Roulis (calculé avec l’accélération)

Lorsque le drone est immobile, l’accélération mesurée est la gravité.

x

y

z

g

g

y

ay

az

z

## Tangage (calculé avec l’accélération)

Lorsque le drone est immobile, l’accélération mesurée est la gravité.

x

y

z

g

g

x

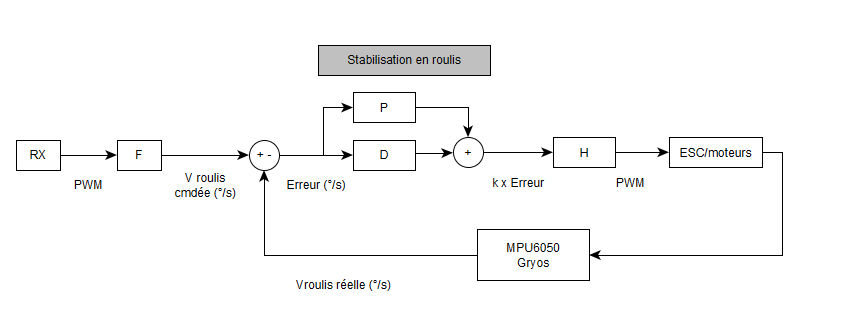
ax

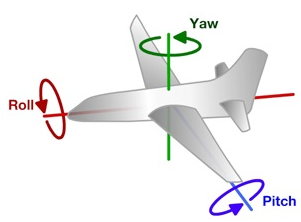
az

z

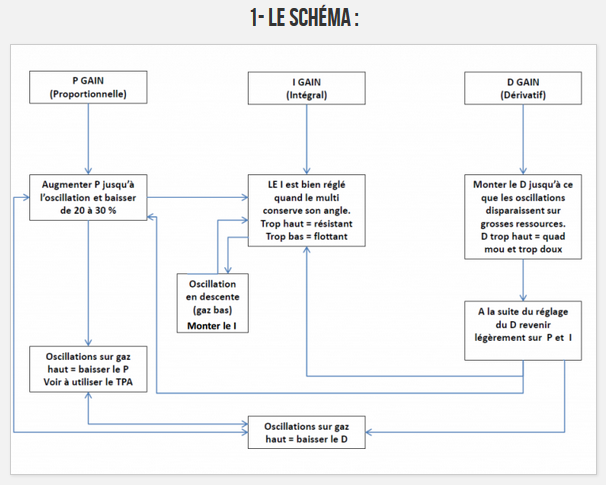
# 

# Stabilisation gyroscopique





## Réglage PID



* Le P

C’est le **P** qui va résoudre les problèmes de **vibrations**. Le I quant à lui, joue sur l’inertie de la machine et sur sa réactivité.

1. Montez le P jusqu’à obtenir une machine qui vibre / oscille. Baissez le I également, ça aide.
2. Descendez le P peu à peu jusqu’à ce que les vibrations disparaissent totalement, même à fond de gaz !

* Le I

Il agît sur la **dérive** du multi est est lié au P. En acro, voltige ou FPV, il sera plus bas que pour une machine dédiée à la vidéo. Dans la vidéo en Français, vous apprendrez comment le régler à l’aide d’un truc connu : placer un poids sur un bras

1. Montez le I jusqu’à obtenir des oscillations faibles en montée et ou en descente. Vous devriez en avoir aussi à fond de gaz.
2. Si vous avez des oscillations en descente : montez le I
3. Si vous avez des oscillations en montée : baissez le I
4. Trouvez une valeur qui vous débarrasse des deux

Si des vibrations réapparaissent, c’est normal. Retouchez le **P**. Baissez le légèrement, c’est selon.

* Le D

Ce paramètre est le plus  » personnel  » des trois. Il influence la réactivité de la machine.

1. Faites de grands mouvements de gauche à droite ou d’avant en arrière pour observer les réactions de la machine. Appréciez et réglez selon vos préférences.
2. Des vibrations peuvent revenir : corrigez le **P**.

* Le Yaw

Si votre machine continue à dériver après un ordre ou si elle dérive seule sur l’axe du lacet, changez la valeur. Elle est très souvent sur **8.5** et est correcte ainsi.

* tpa breakpoint

Ce paramètre joue sur le ratio des PID. En effet, la tension et le niveau de gaz sont des variables qui agissent sur le comportement. Le TPA va faire varier vos PID selon ces facteurs.Si vous n’en mettez pas, il se peut que vous ayez des vibrations lorsque vous êtes à fond de gaz avec une lipo chargée à bloc. Pour être précis, les TPA ( Throtlle PID Attenuation ) jouent sur le P. ( Merci **XKin Ai** pour la précision )

# Fusion de données

Un algorithme de fusion filtre et prédit les données de plusieurs capteurs.

Le gyroscope dérive.

L’accéléromètre n’est pas assez rapide, il est utilisable au repos.

On applique un filtre passe-bas à l’accéléromètre et un filtre passe-haut au gyroscope.

## Filtre complémentaire

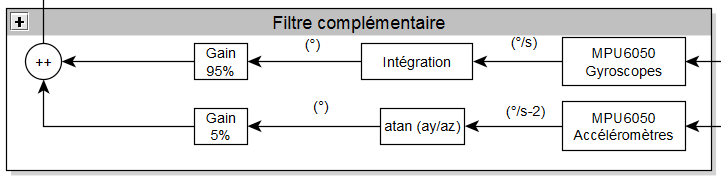
Le filtre complémentaire consiste:

* à appliquer un filtre passe bas sur les données de l’accéléromètre car ses données sont exploitables sur la durée, il faut éliminer les variation brusques.

Le filtre passe-haut consiste en un gain de 5%.

* à appliquer un filtre passe haut sur les données du gyroscope car ses données sont fiable sur le court terme mais prennent de l’erreur dans le temps à cause de sa dérive

Le filtre passe-bas consiste en un gain de 95%.



## Filtre de Kalman

Extended Kalman Filter EKF

# Configuration en “Quad +”

x

y

ESC2 PORTB2

ESC3 PORTB3

ESC1 PORTB1

ESC0 PORTB0

# Configuration en “Quad X”

x

y

ESC2 PORTB2

ESC3 PORTB3

ESC1 PORTB1

ESC0 PORTB0

+

+

roll >0

pitch >0

roll <0

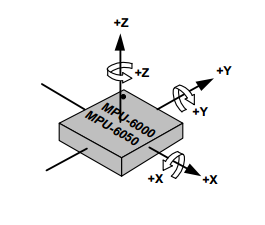
pitch <0

roll < 0

pitch > 0

roll > 0

pitch < 0

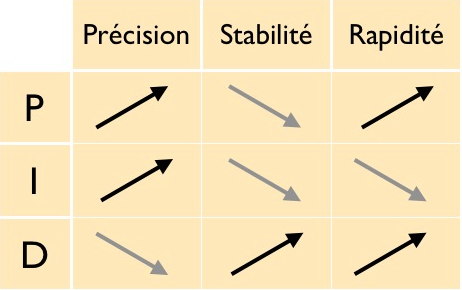


Banggood IMU:

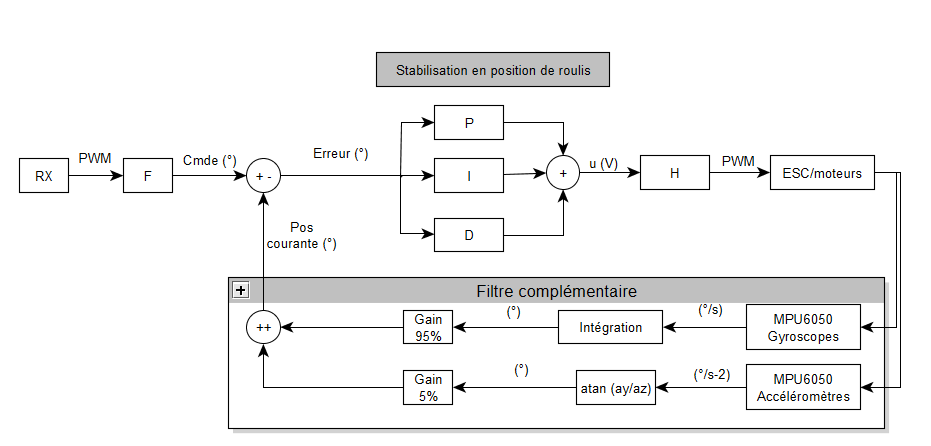
X

Y

# Asservissement, mode “ANGLE”



## Roulis tangage



# Arduino UNO rev 3

Microcontrôleur ATmega328

Architecture 8 bits RISC

16Mhz => T = 0.0625us

1MIPS par MHz

Dimensions : 68.6 mm x 53.4 mm

Poids : 25 g

# ESC

Simonk ESC : 500Hz, 1060us à 1860us de largeur d’impulsion

ESC classique : 490Hz, 1 à 2ms de largeur d’impulsion

## Génération PWM à 400Hz

1. Classer les ESC par ordre croissant de largeur d’impulsion.
2. Utiliser le « Timer1 ».
3. Mettre toutes les sorties au niveau haut en une seule fois à l’aide de la commande « PORTB=0b00001111 ».
4. Mettre à au niveau bas les sortie au fur et à mesure.

|  |
| --- |
| \* For every 2.5 ms period the first 1.5 is unused and can be used for main program logic.  \* From 1.5 to 2.5 ms every interrupt will disturb timing, but without communications most servo controllers are uninteresting.   \* We can use a loop to wait for the next "set servo low" time or we can use a timer interrupt  \* Best precision is if all the servo output pins are on the same AVR port and we directly write to the PORTx register. Then servos with equal timing can be handled with one port write.  \* The situation with the most difficult timing is when two servos are separated by 2-8 uS, less than an interrupt period.  \* After every 2.5 ms servo period we wait 7\*2.5 ms for the next servo control period. We can use these periods to address  other groups of servos if we use demuxes to distribute the servo pulses to the groups. |

# Piloter avec une télécommande RC 2.4Ghz

<http://tiptopboards.free.fr/arduino_forum/viewtopic.php?f=2&t=24>:

La commande est un signal numérique de période constante 20 ms (50 Hz), codé en largeur d'impulsions (PPM).

* 1 ms high correspond à 0% (full brake, reverse)
* 1.5 ms correspond au point neutre (neutral throttle).
* 2 ms high correspond à 100% de commande (full throttle).

Le signal utile (high) est codé en durée, sur 1 à 2 ms, pendant un cycle de 20 ms.la fonction pulsin() permet de détecter le passage à l'état high ou low d'une broche et de chronométrer cette durée.  
Toutefois cette méthode n'est pas optimale, avec 20ms de temps mort des milliers de cycles d'horloge du microcontrôleur sont perdus en temps d'attente. Régler un timeout (en microsecondes) assez court pour éviter de gaspiller les ressources du calculateur. Exemple en anglais <https://www.sparkfun.com/tutorials/348>

Solution retenue : récepteur CPPM.

## Failsafe

Pour la sécurité, définir le « failsafe » pour couper les gaz.

Pour programmer le « failsafe », mettre les commandes de la télécommande dans la configuration souhaitée lors de la perte de réception radio, et « binder » la télécommande. La configuration utilisée pendant le « bind » défini le « failsafe. »

# Code « CodeDroneDIY »

## Connections

PB0 ESC0

PB1 ESC1

PB2 ESC2

PB3 ESC3

PD2 receiver

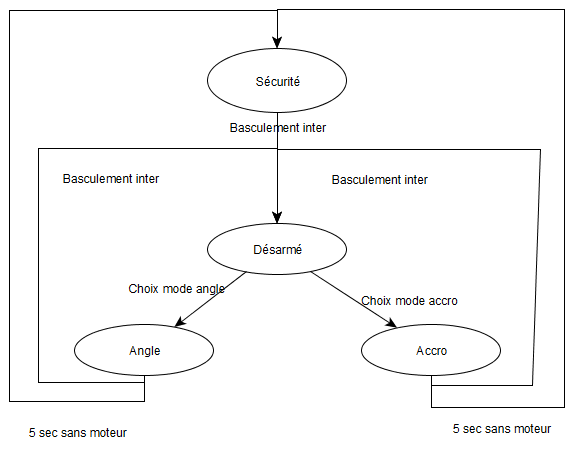
PC2 potentiometer

PC4 SDA MPU6050

PC5 SCL MPU6050

## Machine à états

|  |  |  |
| --- | --- | --- |
| **Etats** | **Evènements** | **Transitions** |
| * Désarmé * Angle * Accro | * Désarmement * Choix du mode * 5 sec sans gaz | * Désarmé vers angle * Désarmé vers accro * Angle vers désarmé * Accro vers désarmé |



# Châssis

250 mm : trop nerveux : pour voler vite et bas, entre les obstacles. Faible autonomie.

Grand châssis pour privilégier la stabilité : 450mm

|  |  |
| --- | --- |
| **Diatone Q450 V3 à 13.42€(stabilité)** | |
| 4 x 22 ~ 28mm 800 ~ 1200kv (ou taille similaire / kv)  4 x 15 ~ 30 ampères esc (30a recommander opto)  4 x 8x4 ~ 10x4.5 prop (2CW & 2ccw) |  |

Hobbyking:

Turnigy D2822/17 Brushless Outrunner 1100kv 102W 9.17€

Turnigy D2830-11 1000kv Brushless Motor 210W 10.40€

Turnigy Plush 30amp speed controller 14.18€

# Configuration retenue

|  |  |
| --- | --- |
| **ESC** | Afro 20A |
| **Moteurs** | Multistar 2216-800Kv |
| **Hélices** | 10x4.5 SF Props 2pc CW 2 pc CCW Rotation (Orange) |
| **Batterie** | Zippy Flightmax 3000mAh 4S |
| **Récepteur** | OrangeRx R617XL CPPM DSM2/DSMX 6 ch |
| **Chassis** | Diatone Q450 Quad 450 V3 |

# Bibliographie

## Arduino

[file:///C:/Program%20Files/Arduino/reference/www.arduino.cc/en/Reference/Libraries.html](file:///C:\Program%20Files\Arduino\reference\www.arduino.cc\en\Reference\Libraries.html)

[file:///C:/Program%20Files/Arduino/reference/www.arduino.cc/en/Reference/HomePage.html](file:///C:\Program%20Files\Arduino\reference\www.arduino.cc\en\Reference\HomePage.html)

## PWM

<https://librepilot.atlassian.net/wiki/display/LPDOC/PWM,+PWMSync,+Oneshot+Output>

<http://forum.arduino.cc/index.php?topic=46487.0>

## PPM

<http://frskytaranis.forumactif.org/t4426-tuto-pwm-cppm-ccpm-ppm-s-bus-s-port-kesako>

## PID

<http://www.fpv-passion.fr/docteur-pid/>

## Drone

<https://www.mondrone.net/fabriquer-quadricoptere-la-propulsion/>

## Traitement de données

<https://ericjformanteaching.wordpress.com/2013/10/08/smoothing-sensor-input/>

## Fusion de données

<http://www.mouser.fr/applications/sensor_solutions_mems/>

<http://www.pieter-jan.com/node/11>

## Programme python

Andy BAKER

<http://pythonfiddle.com/andy-baker-quadcopter/>

## Vibrations

<http://ardupilot.org/plane/docs/common-vibration-damping.html>