Code d'inscription CNC : MH066T

Stabilisation d’un drone quadrirotor

TIPE 2021 : ENJEUX SOCIÉTAUX

Réalisé par : ZTOUTI Saad
Encadré par : AMMAR ABDERAZZAK

Plan de présentation

- Description de système
- Problématique
- Cahier de charge fonctionnel
- Composition du Quadricoptère
- Présentation des composants électronique
- Le contrôleur PID et contrôle des ESC
- L’expérience et les résultats

Description du système

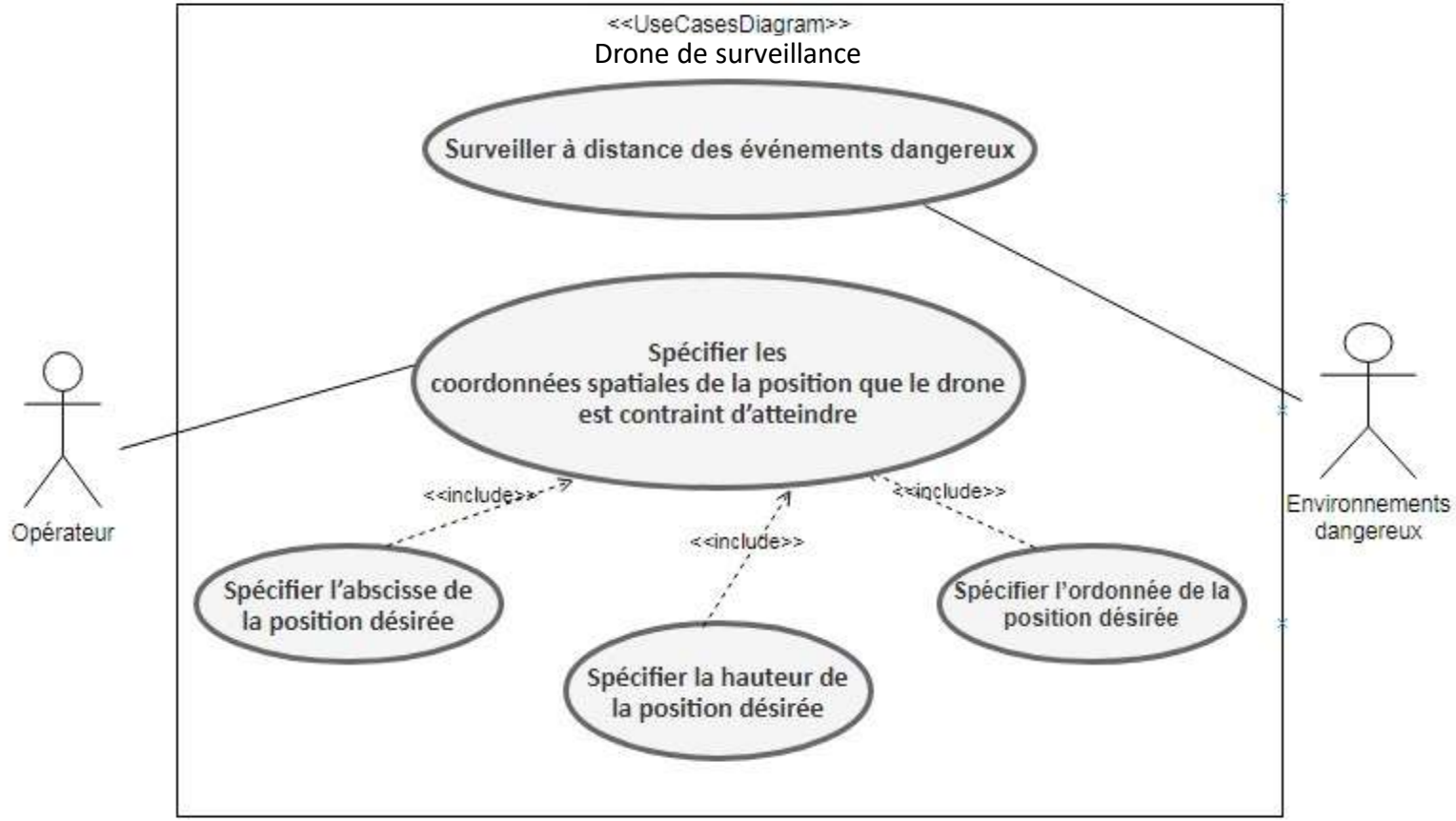
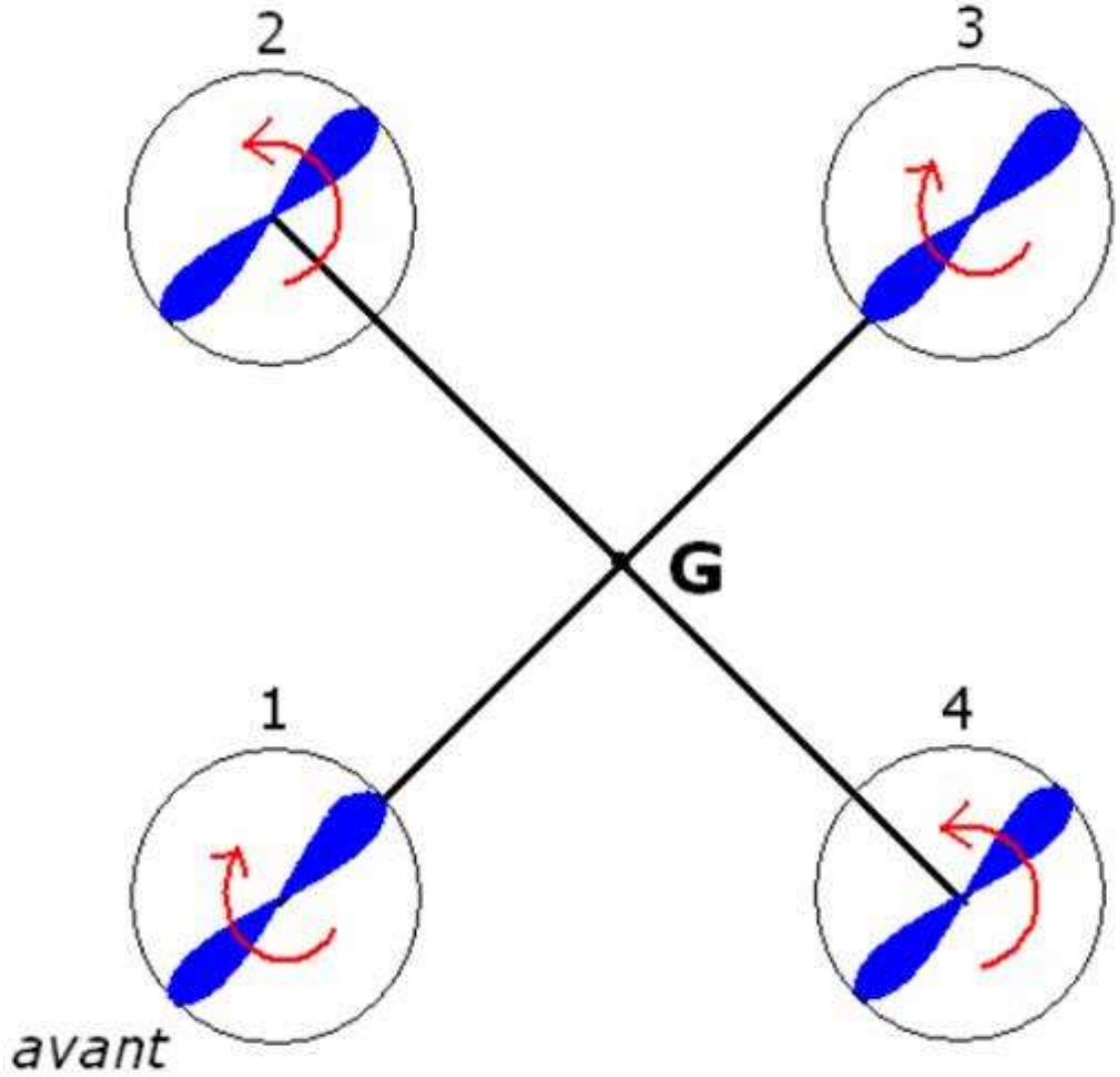


Figure 1 : Diagramme de cas d'utilisation d'un drone de surveillance

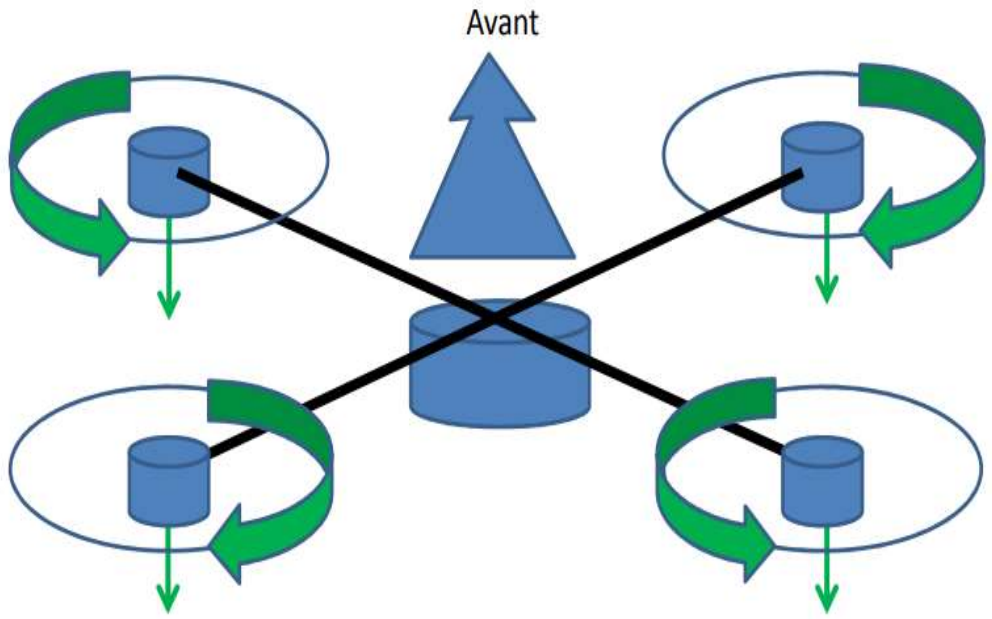
Comment fonctionne un drone ?

Sens des hélices :



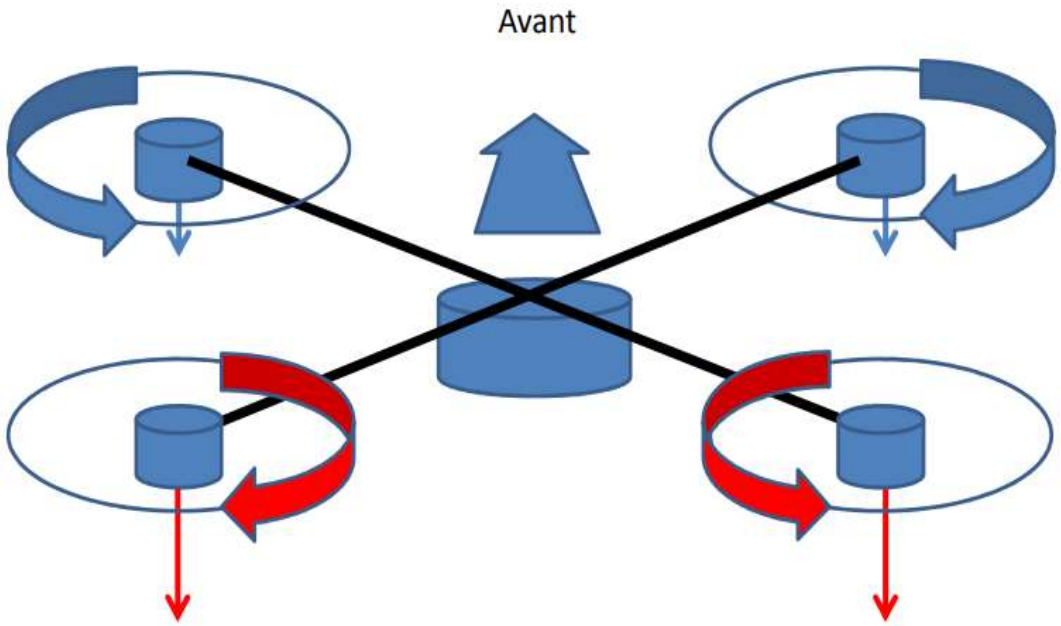
Modifier la position du drone

- Vol stationnaire



En vol stationnaire tous les moteurs tournent à la même vitesse

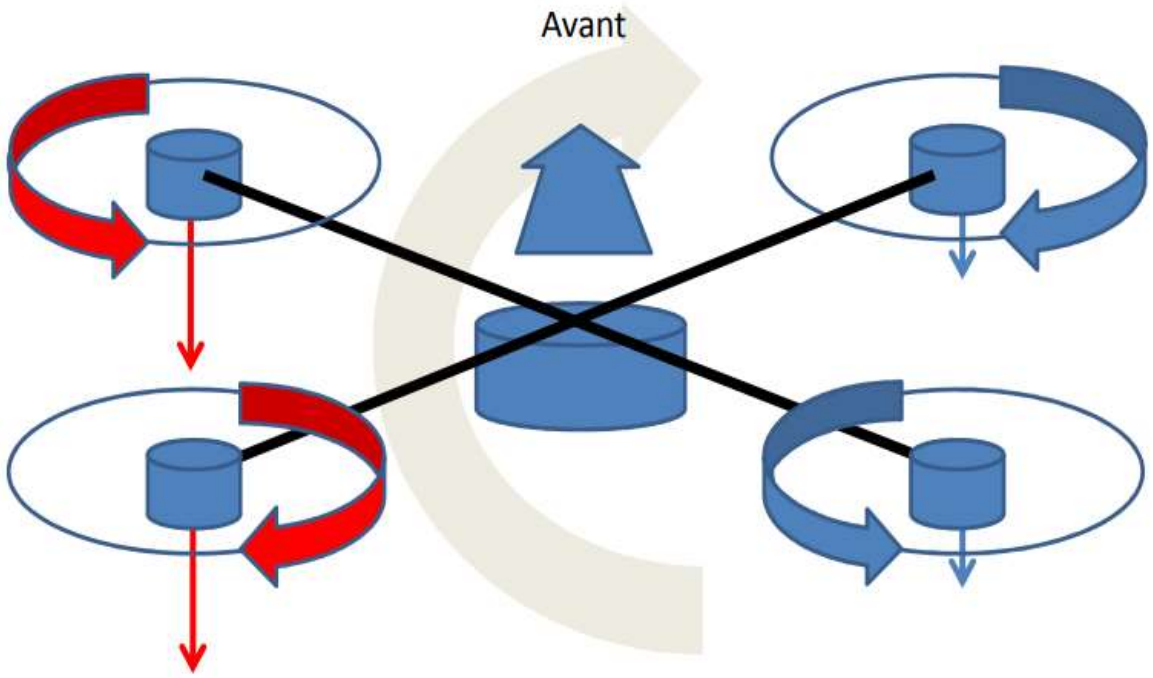
- Tangage : Avancer / Reculer



Pour avancer on ralentit les moteurs avants (cas ci-dessus)
Pour reculer on ralentit les moteurs arrières

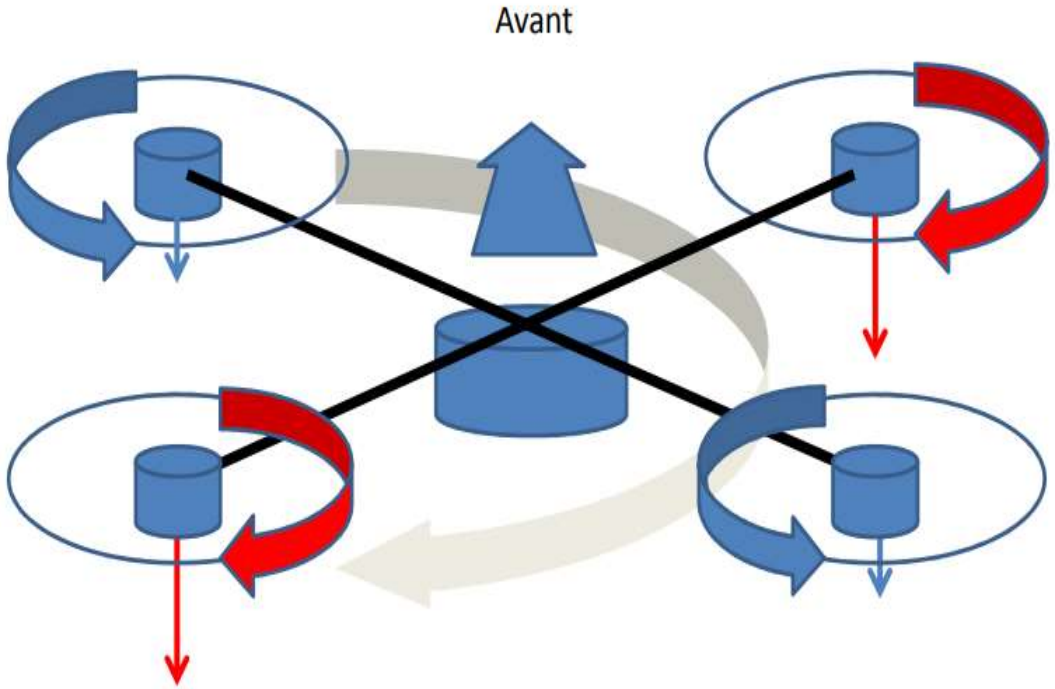
Modifier la position du drone

Roulis : Droite / Gauche



Pour aller à droite on ralenti les moteurs de droite (cas ci-dessus)
Pour aller à gauche on ralentit les moteurs de gauche

Lacet : Rotation



On augmente la vitesse d'une paires d'hélice sur un même axe

Problématique

Le drone doit être capable, en mesurant son propre angle d´inclinaison, de s´auto-équilibrer et tenir la position désiré par le pilote . Alors :

- ✓ Comment stabiliser notre quadrirotor en effectuant les choix convenables des coefficients de régulateur PID ?

Cahier de charge fonctionnel

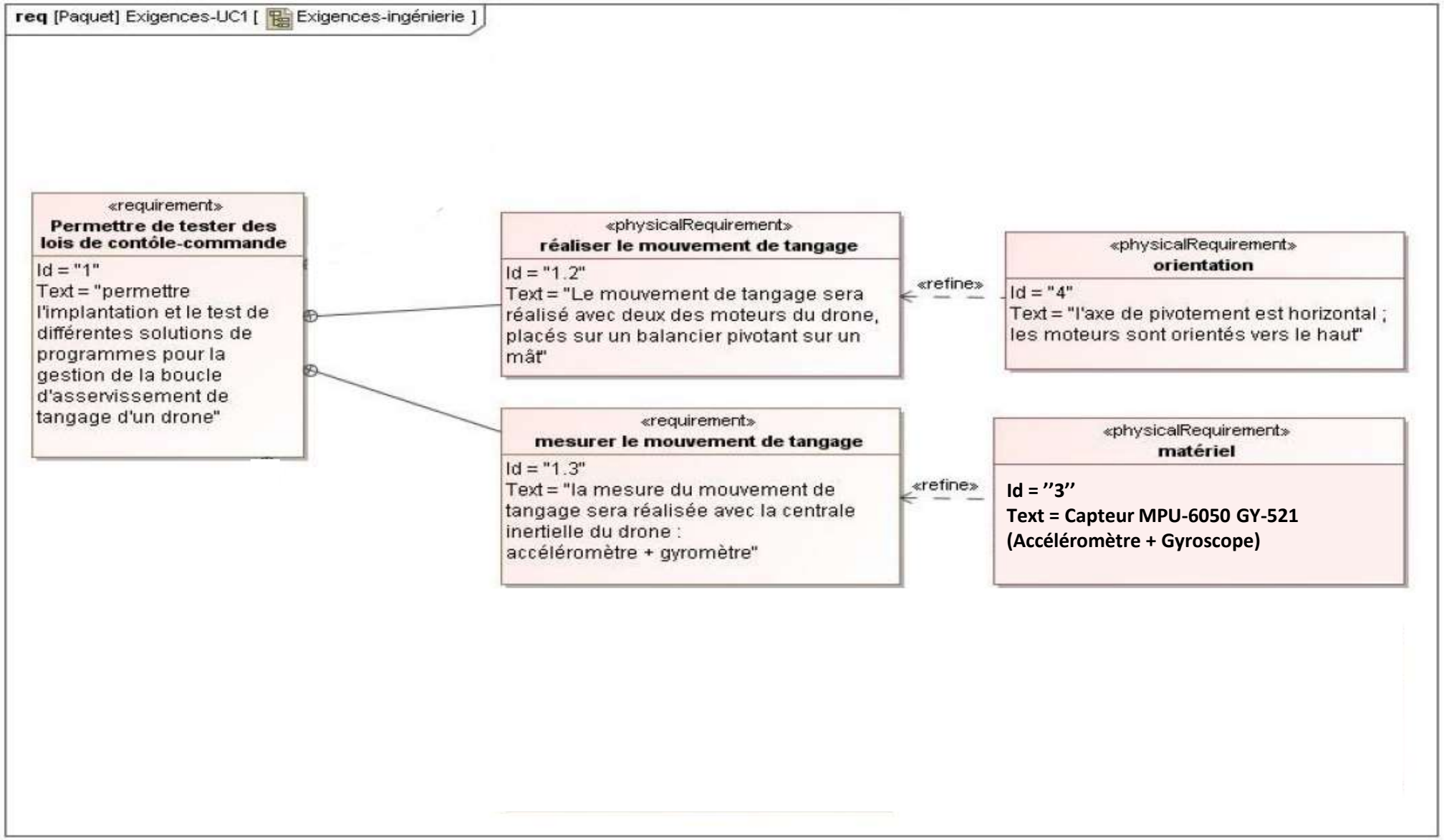


Figure 2 : Diagramme des exigences du drone de surveillance

Force de traction d'une hélice et du drone:

$$T = \rho . Ct . n^2 . D^4$$

T: La force de traction du drone (en Newton)

ρ: La masse volumique de l'air (en $kg.m^{-3}$)

Ct: Coefficient de trainée

n: la vitesse des hélices (en tr.s^-1)

D: diamètre de l'hélice (en m)

Equation de mouvement sur l'axe y :

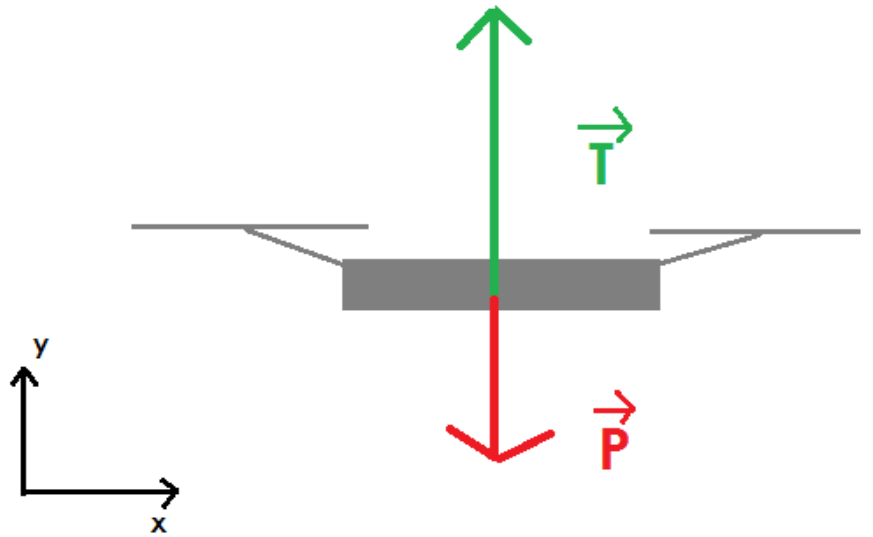
- Tout d'abord il faut déterminer l'accélération sur l'axe y, il faut utiliser la 2eme Loi de Newton:
- Système : drone
- Référentiel : terrestre dit galiléen.

$$\sum \vec{F}ext = m.\vec{a}$$

$$\vec{P} + \vec{T} = m.\vec{a}$$

$$T - P = m.a_y$$

$$\boxed{a_y = \frac{T - P}{m}}$$



$$\begin{cases} a(t) = \dfrac{T - P}{m} \\ v(t) = \left(\dfrac{T - P}{m}\right) t + v(0) \\ y(t) = \dfrac{1}{2}\left(\dfrac{T - P}{m}\right) t^2 + v(0)t + y(0) \end{cases}$$

Equation de la capacités de la capacités de charge

Pour calculer la capacités de charge il faut appliqués la 2eme loi de Newton avec 3 Forces : P le poids du drone, T la force de traction du drone, et F une force inconnue que l'on cherche:

$$\sum \vec{F} ext = 0$$

$$\vec{P} + \vec{T} + \vec{F} = 0$$

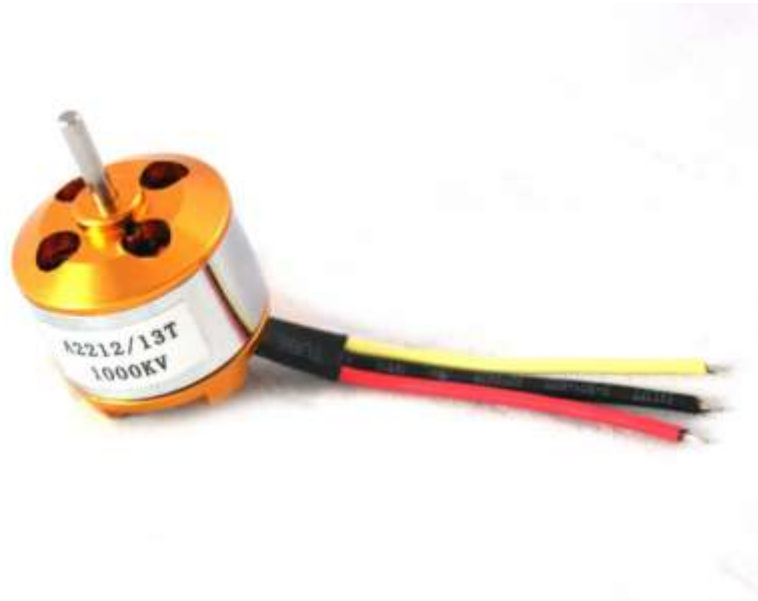
$$F = -(mg - (\rho . N^2 . D^4 . Ct))$$

Composition du quadrirotor

- Moteur BRUSHLESS « les moteurs sans balais »

Qu'est ce qu'un moteur brushless ?

- Moteur synchrone à aiment permanent
- Longue durée de fonctionnement
- Insensibilité aux parasites
- Génère la pousser pour permettre au drone de voler



Que signifié KV d'un moteur brushless?

- 1 KV est équivalent à un tour par minute par volt
- On a choisit le A221/13T (1000KV)

Les caractéristiques d'un moteur A2212/13T

No. of Cells:	2 - 3 Li-Poly 6 - 10 NiCd/NiMH
Kv:	1000 RPM/V
Max Efficiency:	80%
Max Efficiency Current:	4 - 10A (>75%)
No Load Current:	0.5A @10V
Resistance:	0.090 ohms
Max Current:	13A for 60S
Max Watts:	150W
Weight:	52.7 g / 1.86 oz
Size:	28 mm dia x 28 mm bell length
Shaft Diameter:	3.2 mm
Poles:	14
Model Weight:	300 - 800g / 10.5 - 28.2 oz

Figure 3 : Extrait du Datasheet du moteur brushless A2212/13T

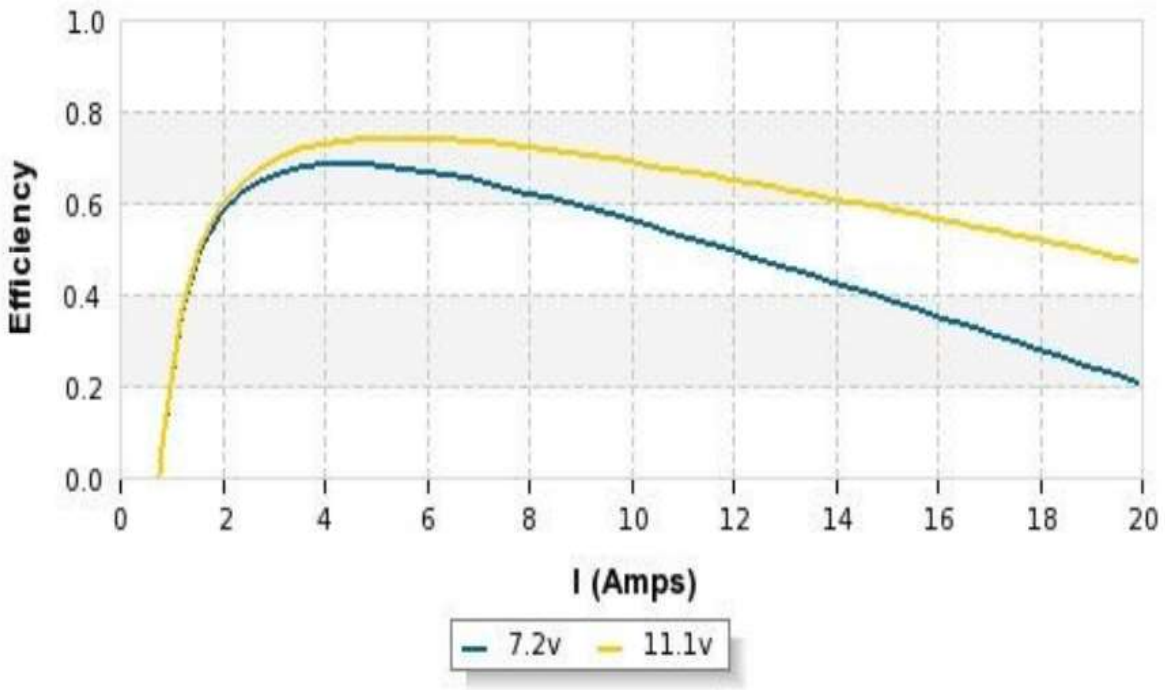


Figure 4 : Le rendement du moteur brushless A2212/13T

Contrôleur de Vitesse ESC du Moteur brushless

Batterie de Lithium-Polymère

Turnigy 2200mAh 3S/11.1v 40C LiPo Pack

Contrôleur ESC du moteur brushless avec Arduino

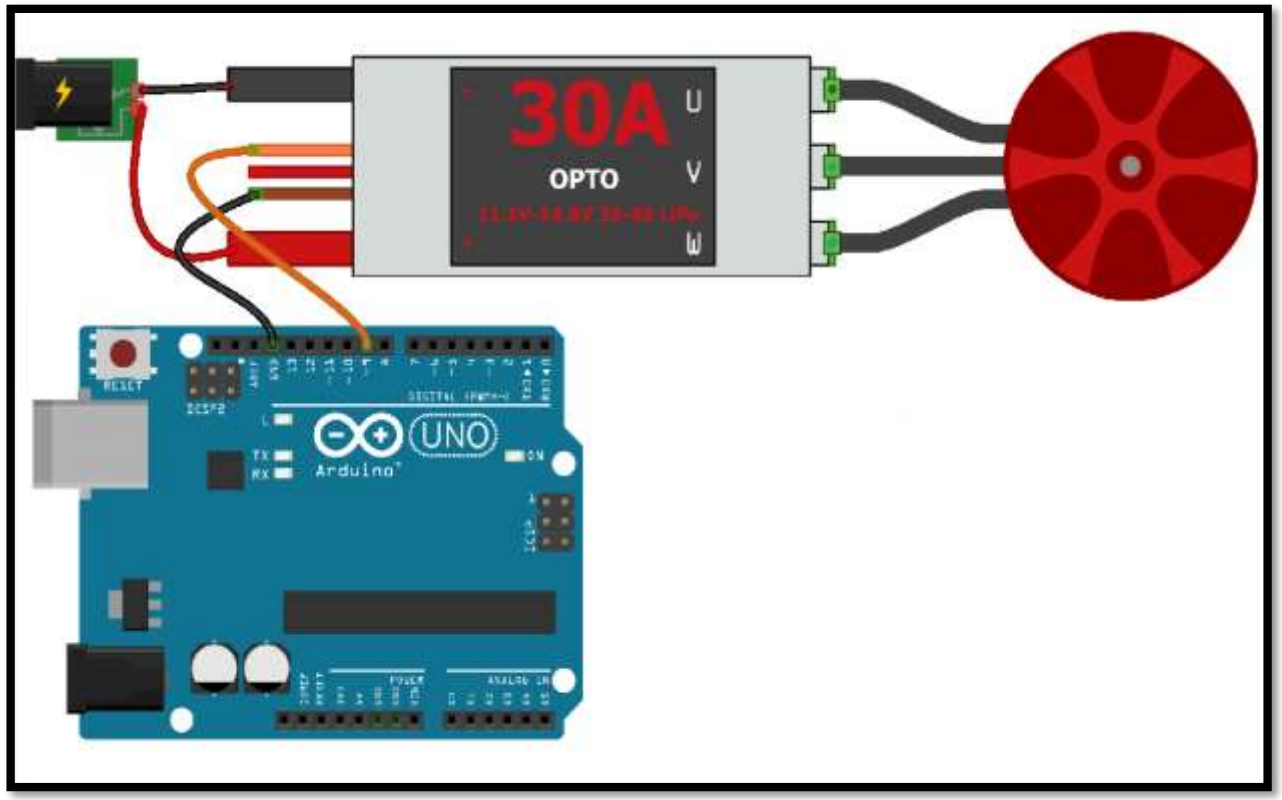


Figure 5 : Schéma éclectique du montage

sketch_jun05a | Arduino 1.8.15 (Windows Store 1.8.49.0)

File Edit Sketch Tools Help

sketch_jun05a §

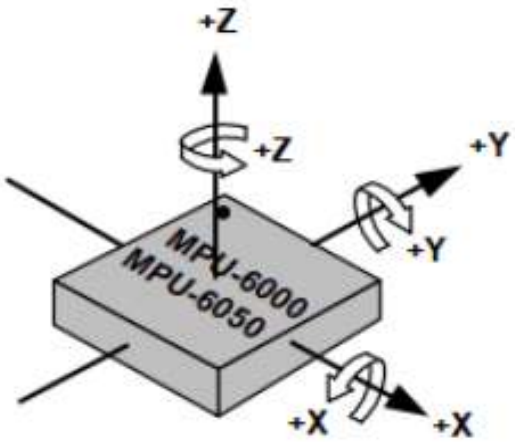
```
#define PinPWM   5

unsigned long Time_ms=0;

void setup()
{
  // Init sortie PWM
  pinMode(PinPWM,OUTPUT);

  // Init du tempo
  Time_ms=millis();
}

void loop()
{
  if(Time_ms<5000)
  {
    digitalWrite(PinPWM, LOW);
    delay(18);
    digitalWrite(PinPWM, HIGH);
    delay(2);
  }
  else
  {
    digitalWrite(PinPWM, LOW);
  }

  // Mise à jour du tempo
  Time_ms=millis();
  Time_ms=Time_ms%10000;
```

Figure 6 : code de contrôle de moteur brusless par l'ESC

Présentation des composants électronique

Capteur MPU-6050 GY-521 (Accéléromètre + Gyroscope)

Figure 7 : le module MPU 6050



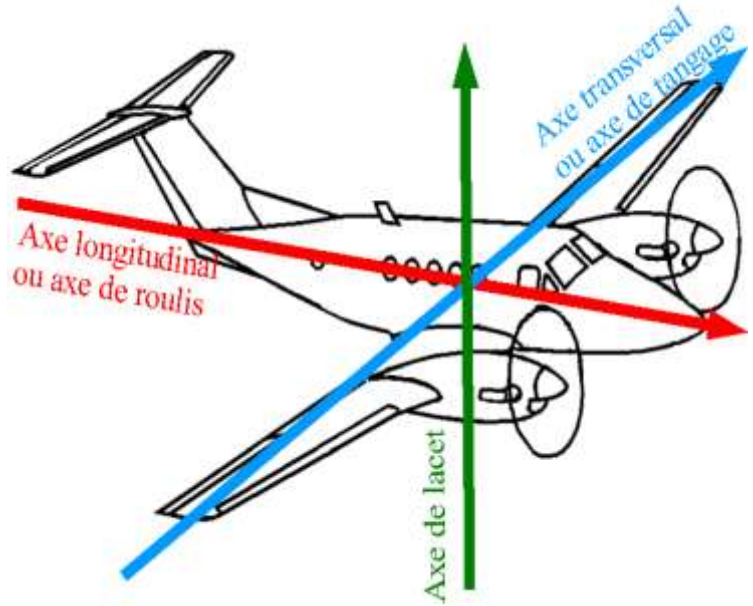


Figure 8 : les différents axes de mouvements

Caractéristiques du module MPU-6050:

- Tension d'alimentation **: 2.375V-3.46V**
- Courant de fonctionnement du gyroscope**: 3.6mA**
- Précision du gyroscope **: ± 250, ± 500, ± 1000 et ± 2000 (°/ s)**
- Courant de fonctionnement normal de l'accéléromètre**: 500µA**
- Précision de l'accéléromètre : **±2g, ±4g, ±8g et ±16g**
- Type de communication **: bus I2C**

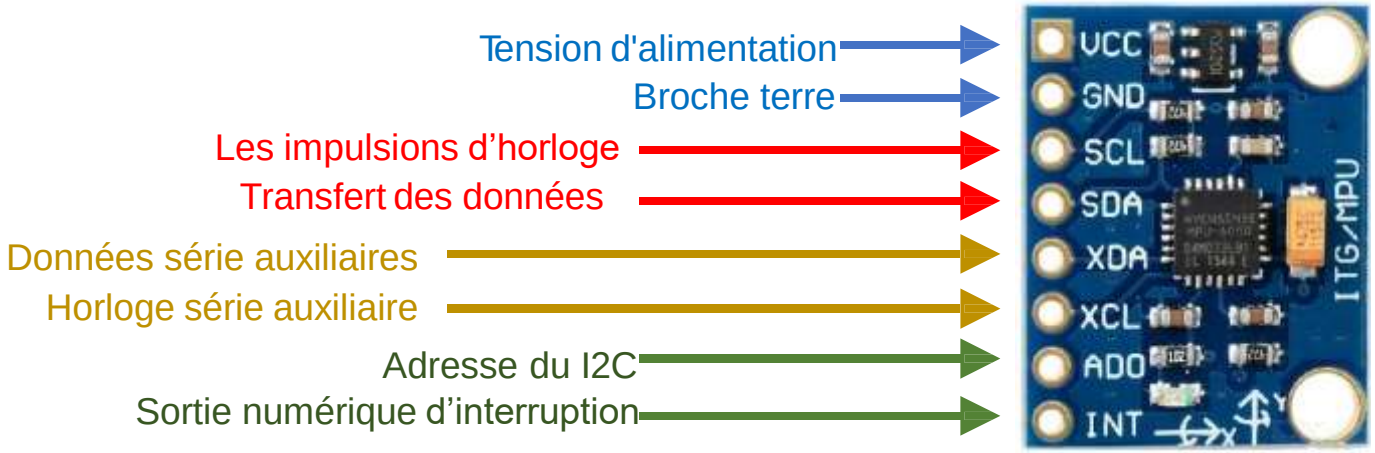


Figure 9 : Différentes broches du module MPU-6050

Accéléromètre

Calcule des angles(tangage , roulis, lacet) a partir des trois valeurs d'accélérations (Ax ,Ay , Az):

$$\theta_x = arctan(\frac{A_x}{\sqrt{A_y^2+A_z^2}})$$

$$\theta_y = arctan(\frac{A_y}{\sqrt{A_x^2+A_z^2}})$$

Gyroscope

Calcule des angles(tangage ,roulis , lacet) a partir des trois valeurs de la vitesse angulaire (Gyro_x , Gyro_y , Gyro_z):

$$\theta_x = \theta'_x + Gyro_x * t$$

$$\theta_y = \theta'_y + Gyro_y * t$$

θ':L'angle précédente

t: le temps écoulé

Le contrôleur PID et contrôle des ESC

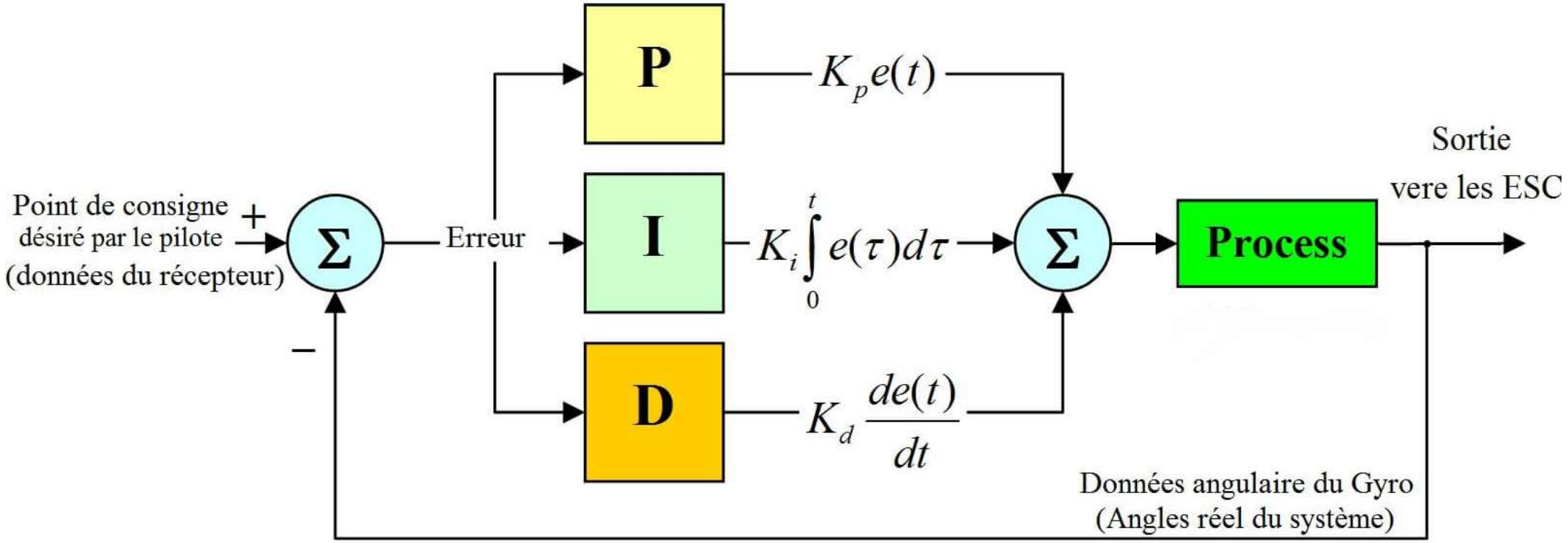
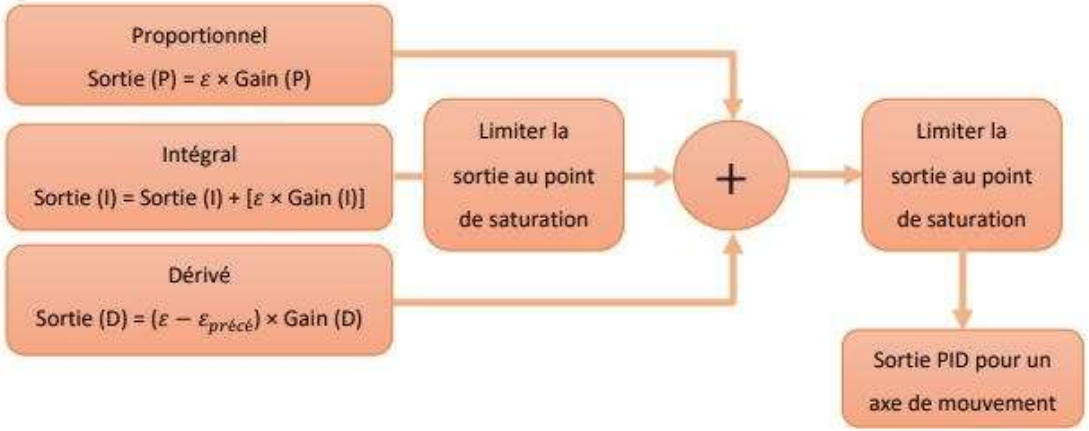


Figure 10 : Schéma de la boucle de contrôleur PID

Algorithme PID du contrôleur de vol :

L'organigramme suivant explique le calcul du sorties PID pour un axe de mouvement « Roll ».



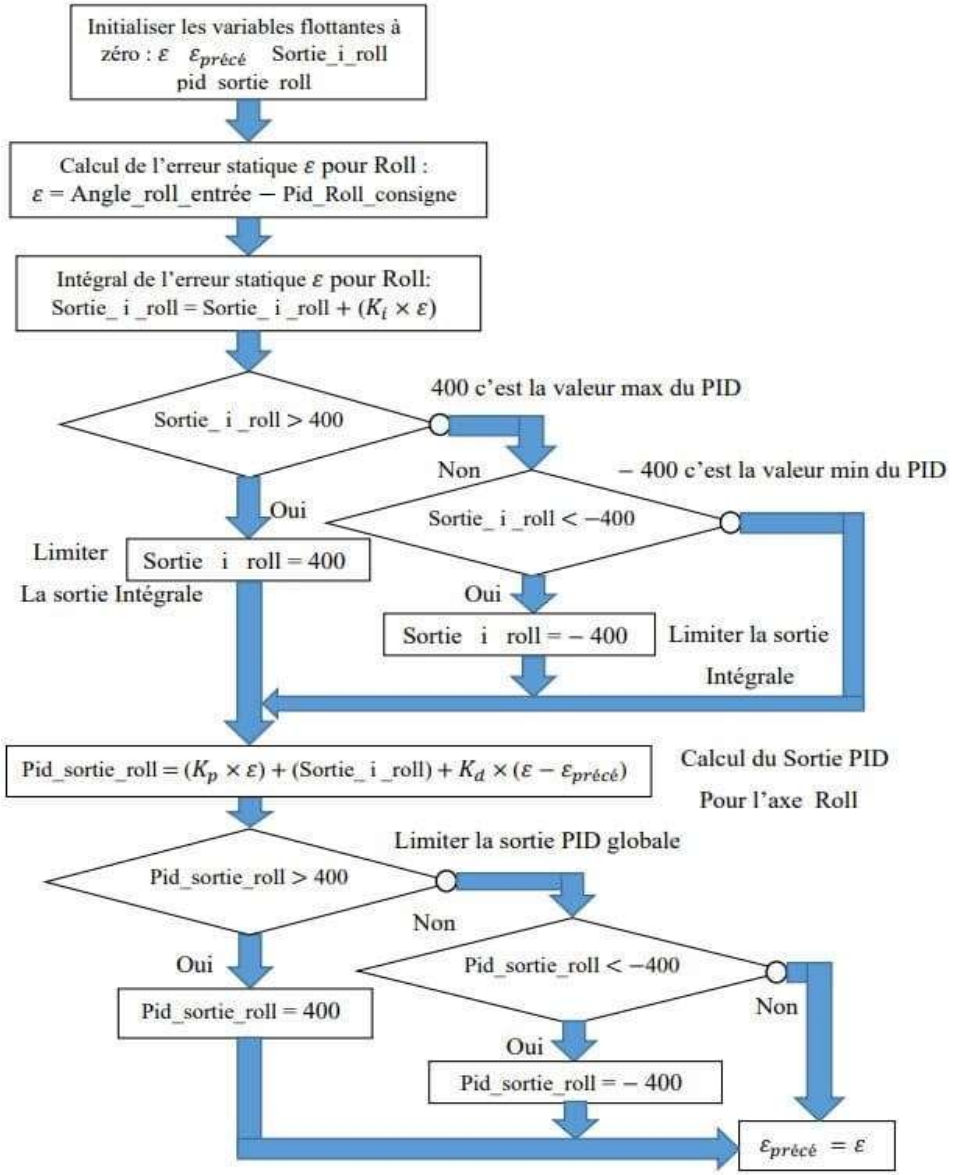


Figure 11 : L'algorithme utilisé dans le sous-programme PID du contrôleur de vol pour un axe de mouvement (roulis).

Contrôle des ESC

- Pour calculer les impulsions des ESC, il suffit de combiner les variables de sorties PID séparées (Pitch, Roll et Yaw) et de les ajouter ou les soustraire à la variable d'accélération suivant cet algorithme :

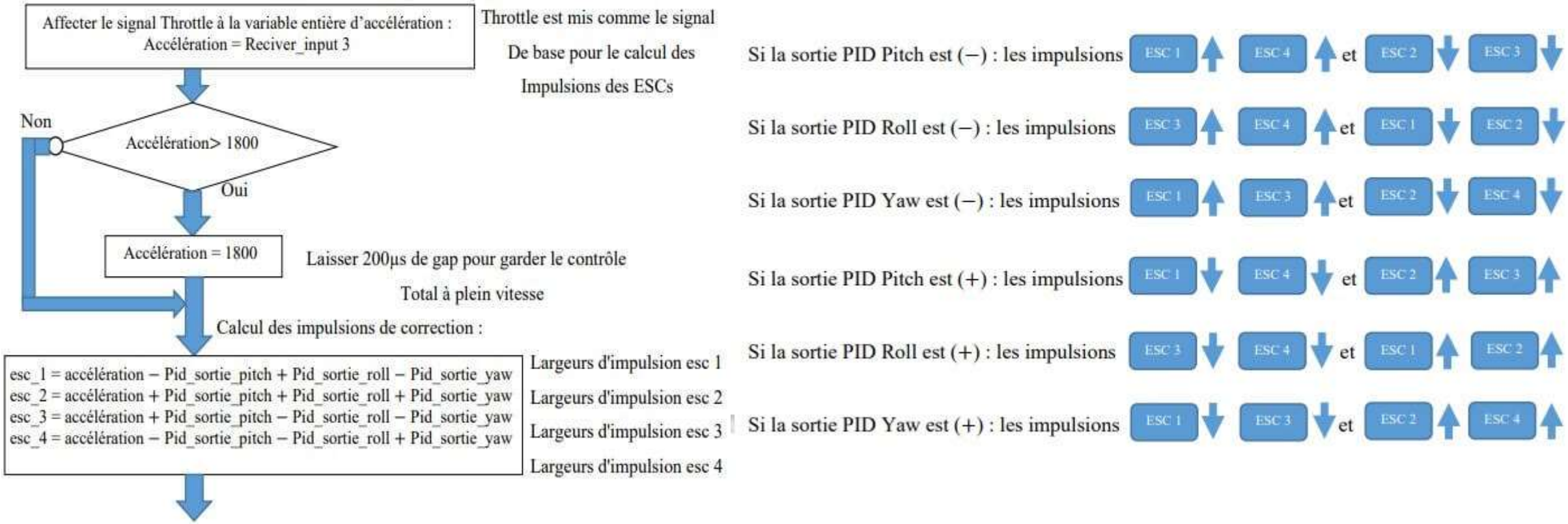


Figure 12 : L'algorithme du contrôle des ESC

L’expérience et les résultats

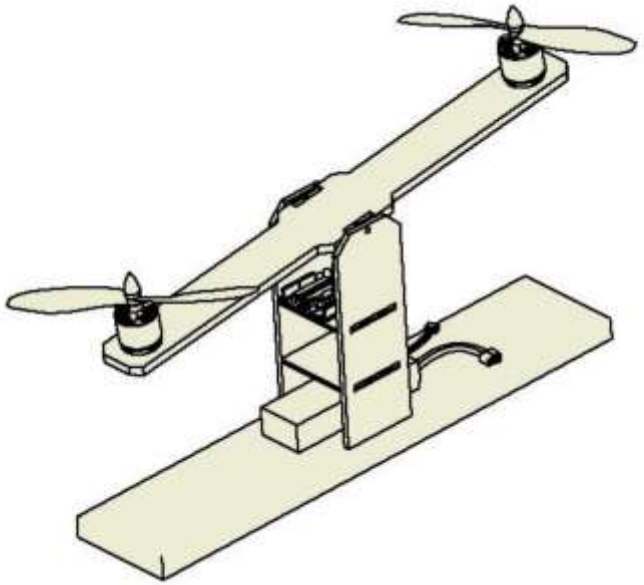


Figure 13 : La conception de la balance bicoptère

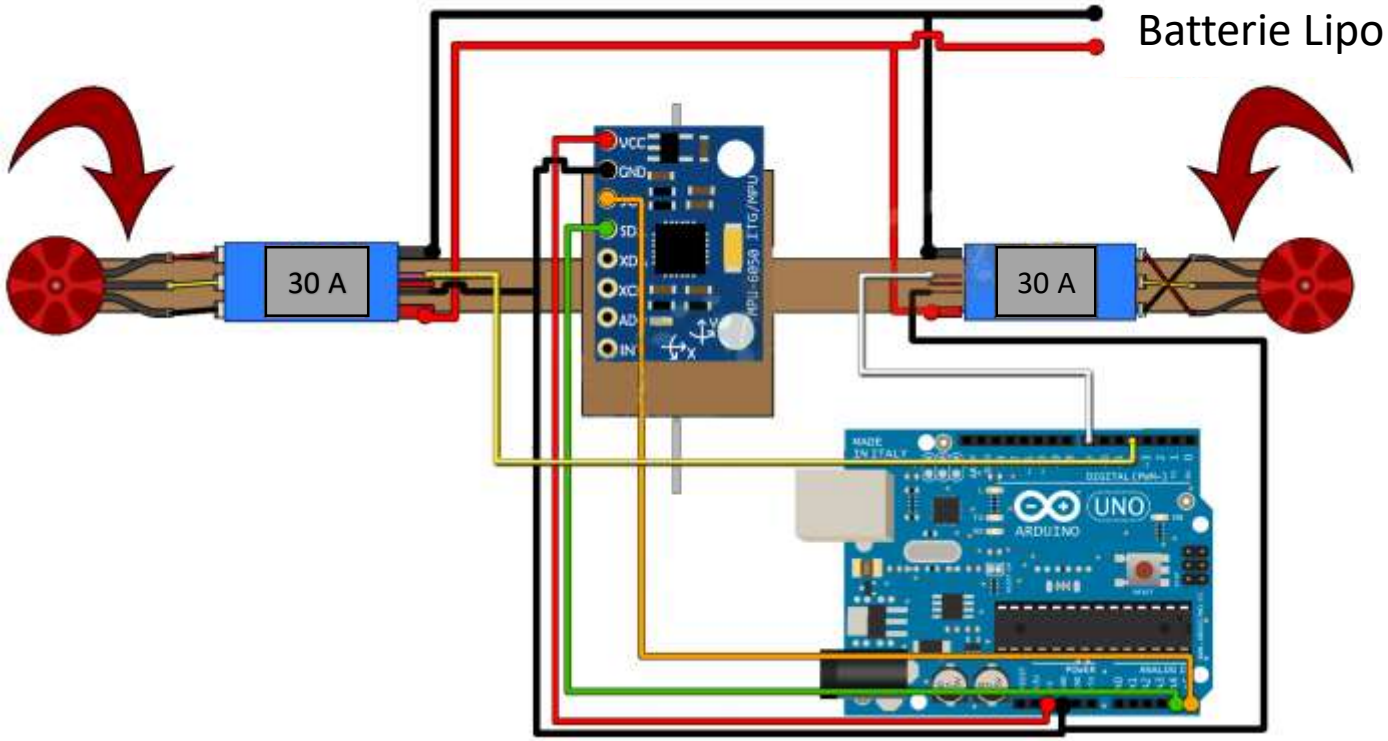


Figure 14 : Schéma électrique du montage

L’expérience et les résultats

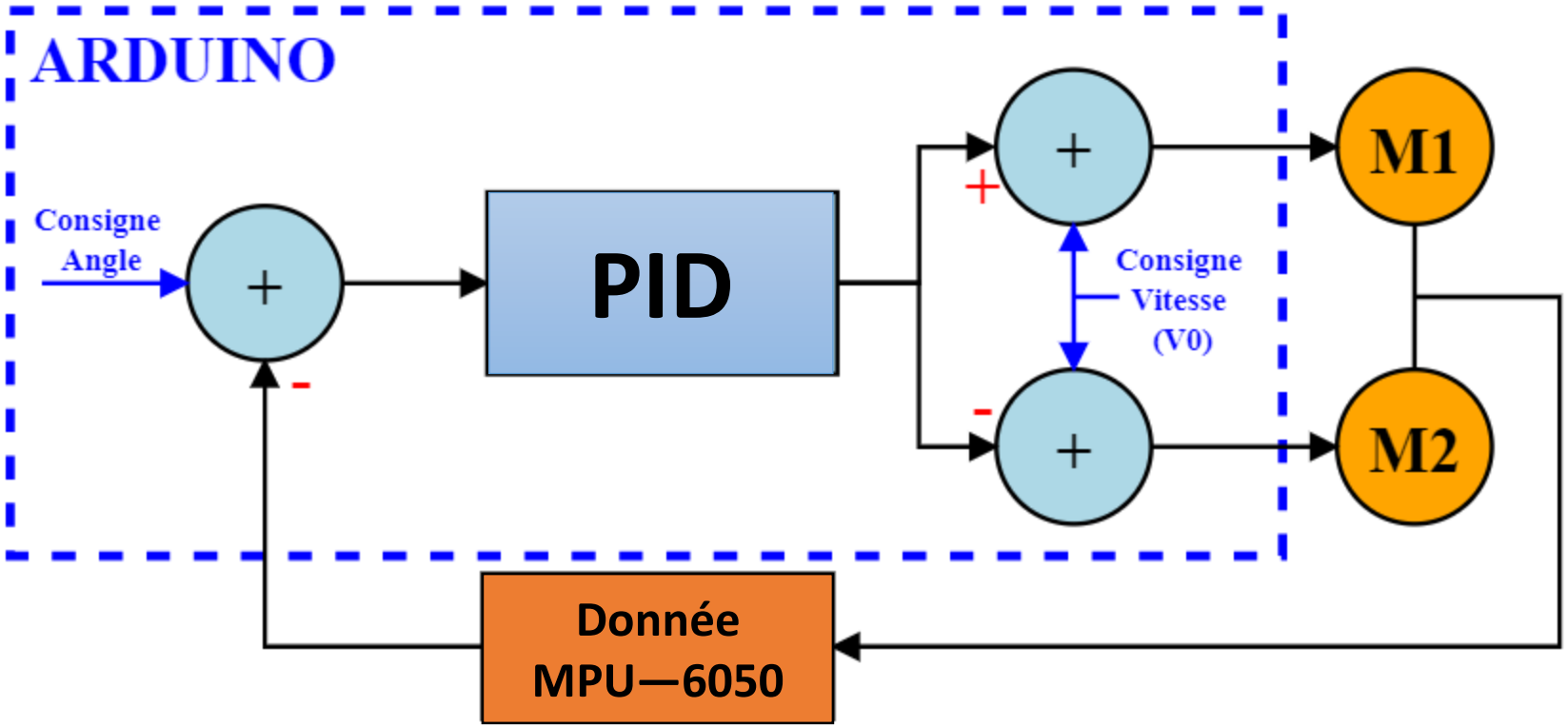


Figure 15 : la Boucle d’Asservissement d’un Drone Bicoptère

L’expérience et les résultats

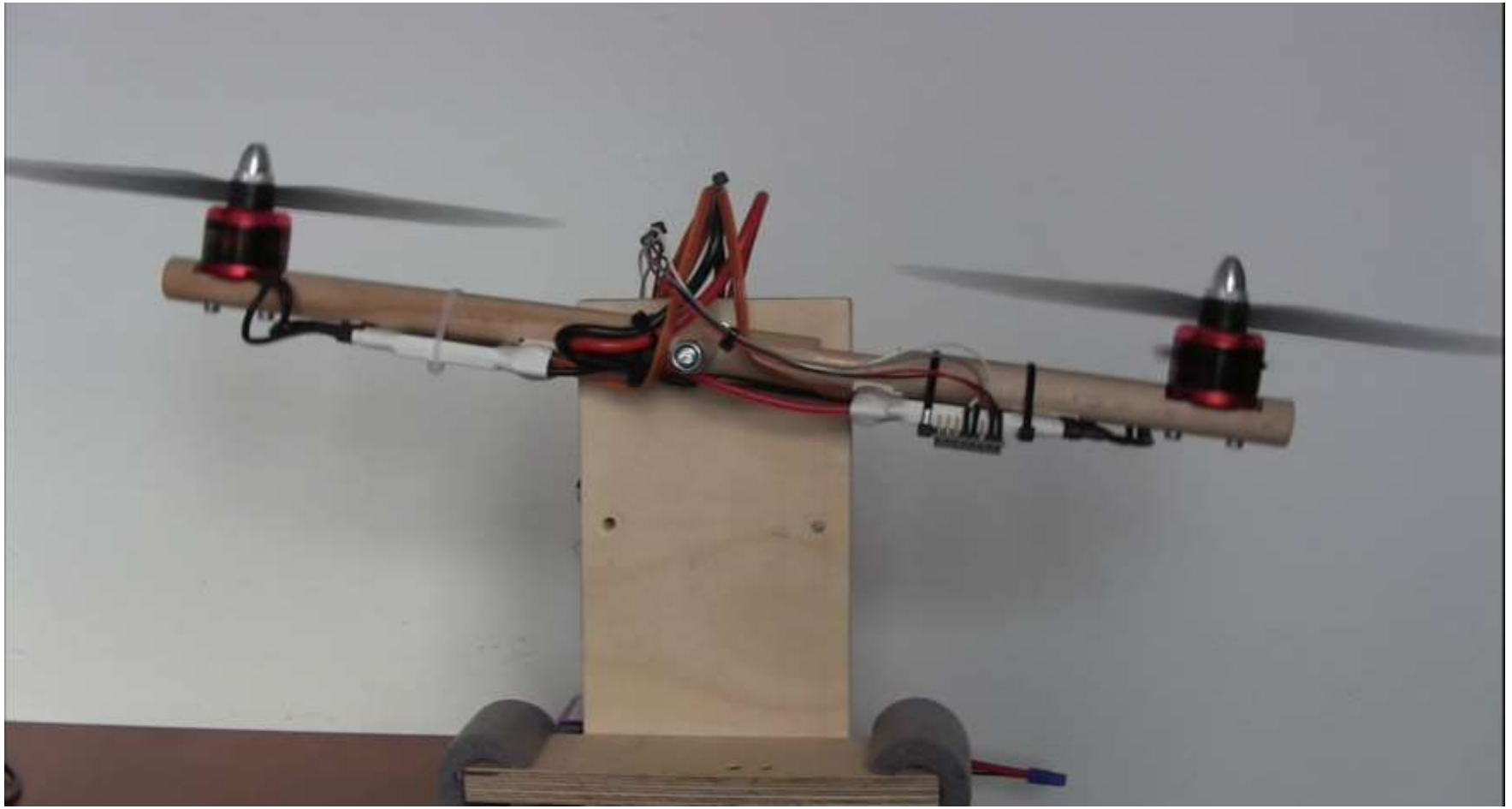


Figure 16 : la balance bicoptère réel

Les coefficients choisis

Type de controlleur	Kp	KI	Kd
Contrôleur P	0.5	--	--
Contrôleur PI	0.45	0.833	--
Contrôleur PID	0.6	0.5	0.125

Figure 17: tableau des coefficients choisis pour les différents correcteur

La réponse de l’angle de roulis

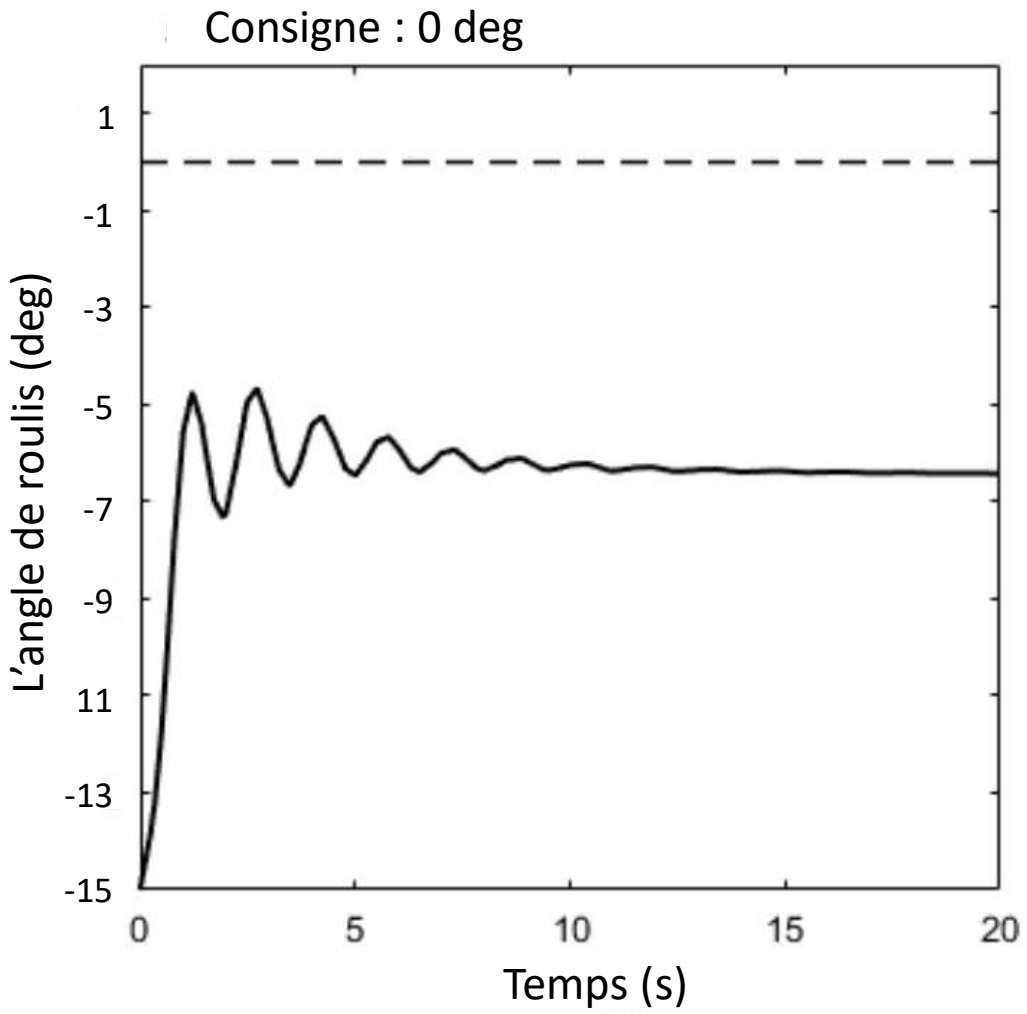


Figure 18 :Réponse en boucle fermée du système à l'aide du contrôleur P

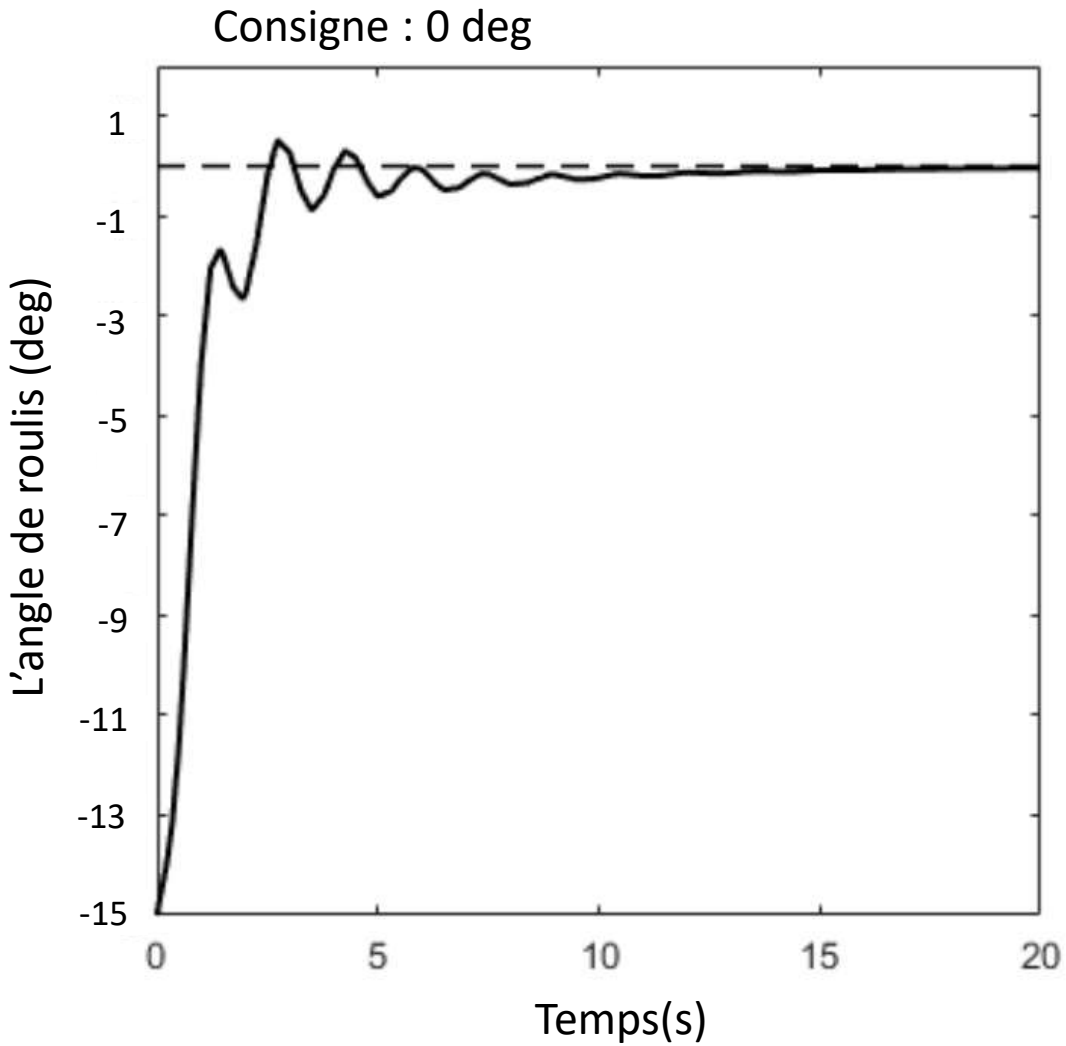


Figure 19 :Réponse en boucle fermée du système à l'aide du contrôleur PI

La réponse de l’angle de roulis

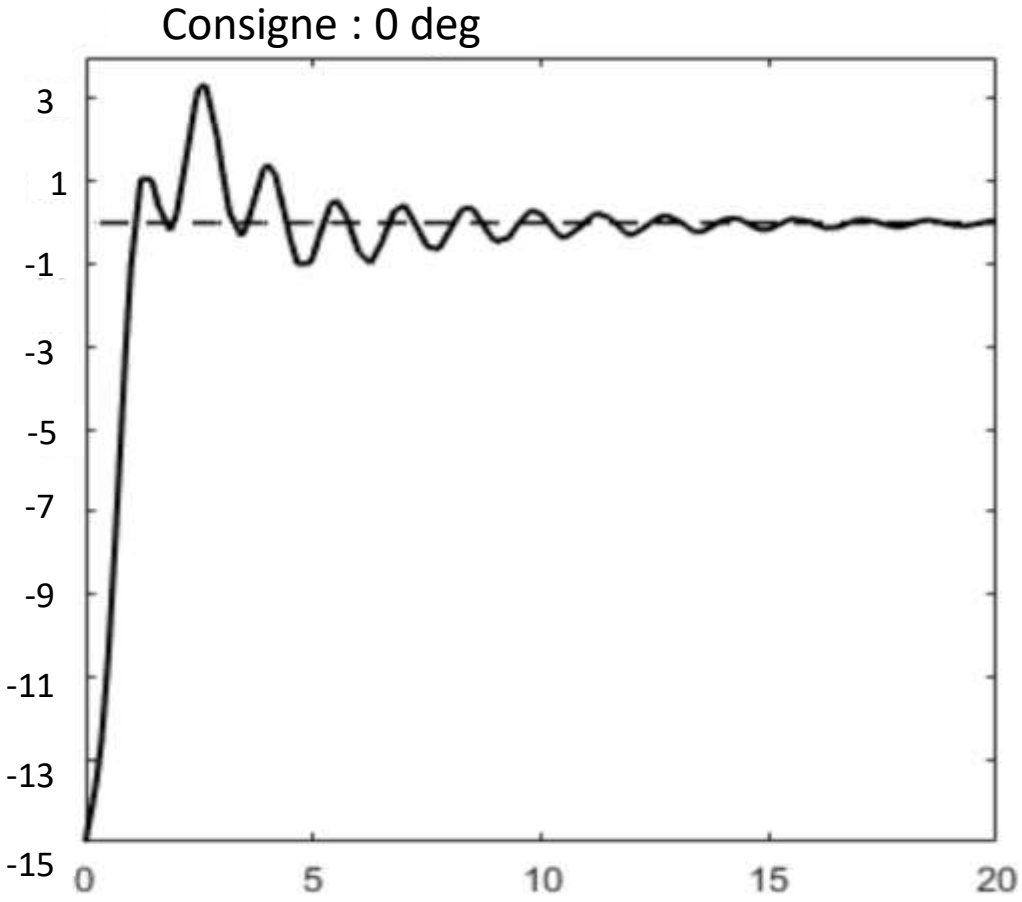


Figure 19 :Réponse en boucle fermée du système à l'aide du contrôleur PID

CONCLUSION

Merci pour votre attention

Annexe

PID_balance_arduino | Arduino 1.8.15 (Windows Store 1.8.49.0)

File Edit Sketch Tools Help

PID_balance_arduino §

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


Servo right_prop;
Servo left_prop;
int16_t Acc_rawX, Acc_rawY, Acc_rawZ,Gyr_rawX, Gyr_rawY, Gyr_rawZ;
float Acceleration_angle[2];
float Gyro_angle[2];
float Total_angle[2];
float elapsedTime, time, timePrev;
int i;
float rad_to_deg = 180/3.141592654;
float PID, pwmLeft, pwmRight, error, previous_error;
float pid_p=0;
float pid_i=0;
float pid_d=0;
////////////////PID CONSTANTS////////////////
double kp=0.6;
double ki=0.5;
double kd=0.125;
///////////////////////////////////////////////
double throttle=1300;
float desired_angle = 0;
void setup() {
  Wire.begin();
  Wire.beginTransmission(0x68);
  Wire.write(0x6B);
  Wire.write(0);
  Wire.endTransmission(true);
  Serial.begin(250000);
  right_prop.attach(3);
  left_prop.attach(5);
  time = millis();
  left_prop.writeMicroseconds(1000);
  right_prop.writeMicroseconds(1000);
  delay(7000);
}//end of setup void
```

PID_balance_arduino | Arduino 1.8.15 (Windows Store 1.8.49.0)

File Edit Sketch Tools Help

PID_balance_arduino §

```
void loop() {

//////////////////////////////I M U/////////////////////////////////////
    timePrev = time;
    time = millis();
    elapsedTime = (time - timePrev) / 1000;
     Wire.beginTransmission(0x68);
     Wire.write(0x3B); //Ask for the 0x3B register- correspond to AcX
     Wire.endTransmission(false);
     Wire.requestFrom(0x68,6,true);
     Acc_rawX=Wire.read()<<8|Wire.read(); //each value needs two registres
     Acc_rawY=Wire.read()<<8|Wire.read();
     Acc_rawZ=Wire.read()<<8|Wire.read();
     Acceleration_angle[0] = atan((Acc_rawY/16384.0)/sqrt(pow((Acc_rawX/16384.0),2) + pow((Acc_rawZ/16384.0),2)))*rad_to_deg;
     /*---Y---*/
     Acceleration_angle[1] = atan(-1*(Acc_rawX/16384.0)/sqrt(pow((Acc_rawY/16384.0),2) + pow((Acc_rawZ/16384.0),2)))*rad_to_deg;
   Wire.beginTransmission(0x68);
   Wire.write(0x43); //Gyro data first adress
   Wire.endTransmission(false);
   Wire.requestFrom(0x68,4,true); //Just 4 registers
   Gyr_rawX=Wire.read()<<8|Wire.read(); //Once again we shif and sum
   Gyr_rawY=Wire.read()<<8|Wire.read();
   /*---X---*/
   Gyro_angle[0] = Gyr_rawX/131.0;
   /*---Y---*/
   Gyro_angle[1] = Gyr_rawY/131.0;
   /*---X axis angle---*/
   Total_angle[0] = 0.98 *(Total_angle[0] + Gyro_angle[0]*elapsedTime) + 0.02*Acceleration_angle[0];
   /*---Y axis angle---*/
   Total_angle[1] = 0.98 *(Total_angle[1] + Gyro_angle[1]*elapsedTime) + 0.02*Acceleration_angle[1];
```

Code ARDUINO de la balance bicoptère

Annexe

PID_balance_arduino | Arduino 1.8.15 (Windows Store 1.8.49.0)

File Edit Sketch Tools Help

PID_balance_arduino §

```
/*////////////////////////////P I D///////////////////////////////////*/
error = Total_angle[1] - desired_angle;
pid_p = kp*error;
if(-3 <error <3)
{
  pid_i = pid_i+(ki*error);
}
pid_d = kd*((error - previous_error)/elapsedTime);
PID = pid_p + pid_i + pid_d;
if(PID < -1000)
{
  PID=-1000;
}
if(PID > 1000)
{
  PID=1000;
}
pwmLeft = throttle + PID;
pwmRight = throttle - PID;
//Right
if(pwmRight < 1000)
{
  pwmRight= 1000;
}
if(pwmRight > 2000)
{
  pwmRight=2000;
}
//Left
if(pwmLeft < 1000)
{
  pwmLeft= 1000;
}
if(pwmLeft > 2000)
{
  pwmLeft=2000;
}

left_prop.writeMicroseconds(pwmLeft);
right_prop.writeMicroseconds(pwmRight);
previous_error = error; //Remember to store the previous error.

}//end of loop void
```