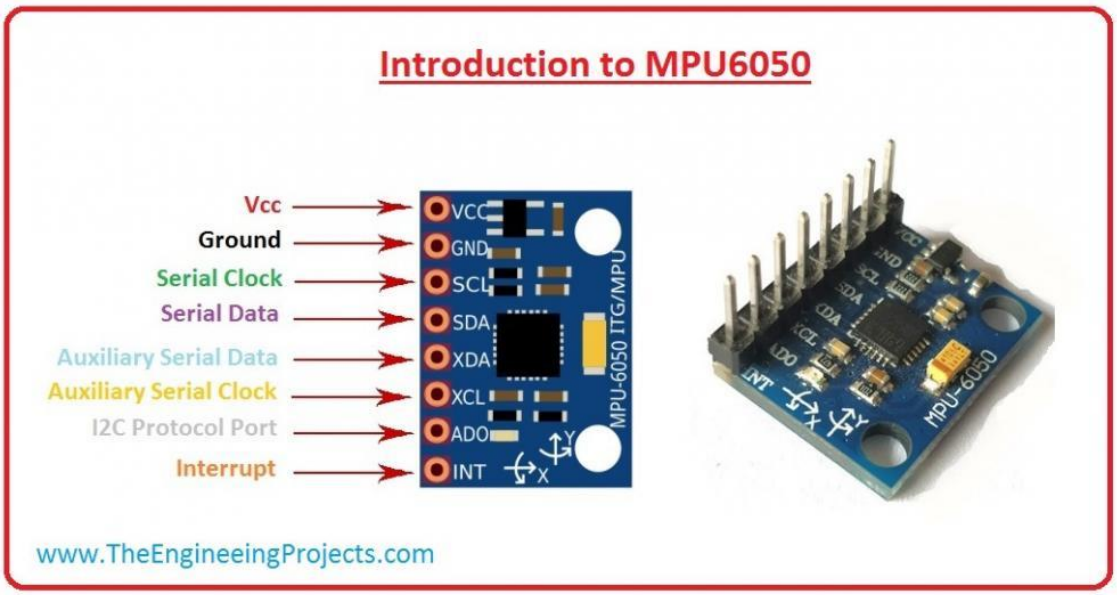
DRONE STATIONNAIRE

PROJET ELECTRONIQUE PEIP 2 LEHYAN FLOURIOT G3

Rapport de séance 3 :

L'objectif de cette séance est d'essayer de faire voler le drone d'une manière stable !

Pour ce faire, je vais avoir besoin d'un accéléromètre, nommé MPU6050. Ce petit composant électronique se présente comme-ceci :



Le composant va transmettre à l'Arduino des données de position et plus précisément d'inclinaison.

Cela en fait un composant que l'on nomme accéléromètre 6 axes (3 axes d'inclinaison et 3 axes d'accélération).

Il va nous permettre de mesurer très précisément les corrections à faire pour maintenir le drone droit et donc stable.

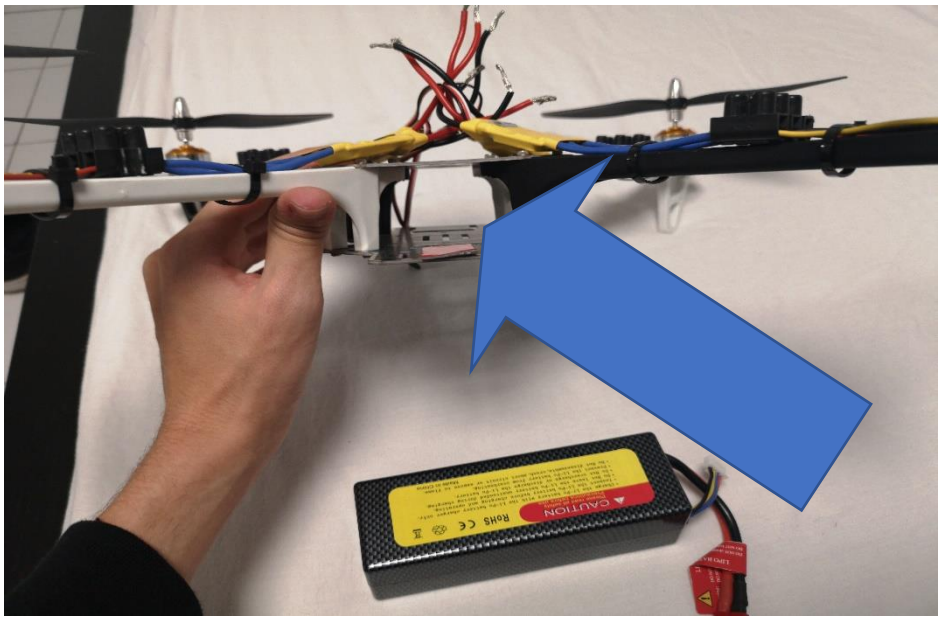
Lors de cette étape, le drone risque de chuter de nombreuses fois, je vais donc réaliser les tests sans les hélices dans un premier temps, puis, je vais probablement essayer de le protéger avec une sorte de coque (je ne sais pas encore).

Commençons par peaufiner ce drone.

Je ne crois pas l'avoir précisé mais les moteurs ne doivent PAS être alimentés par le 5V de l'Arduino et encore moins par USB avec l'ordinateur.

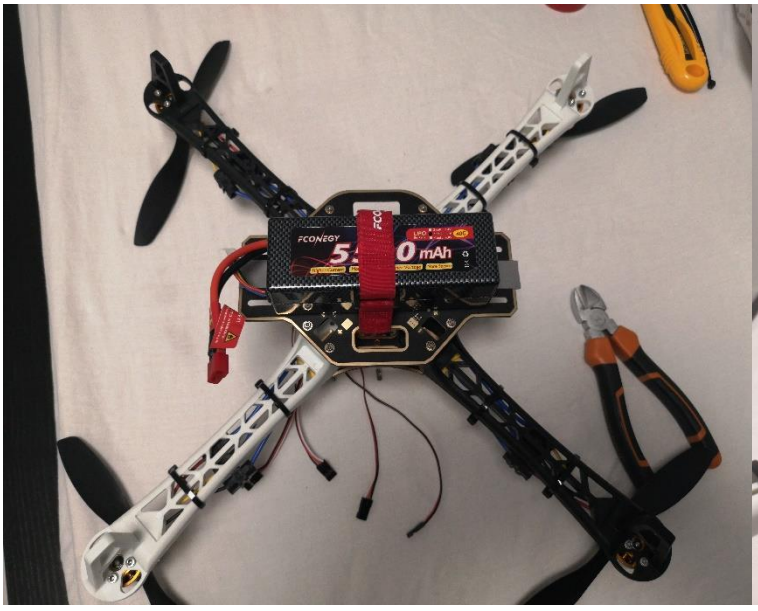
En effet, il faut donc que je fixe la batterie sur le drone afin de le rendre autosuffisant en énergie.

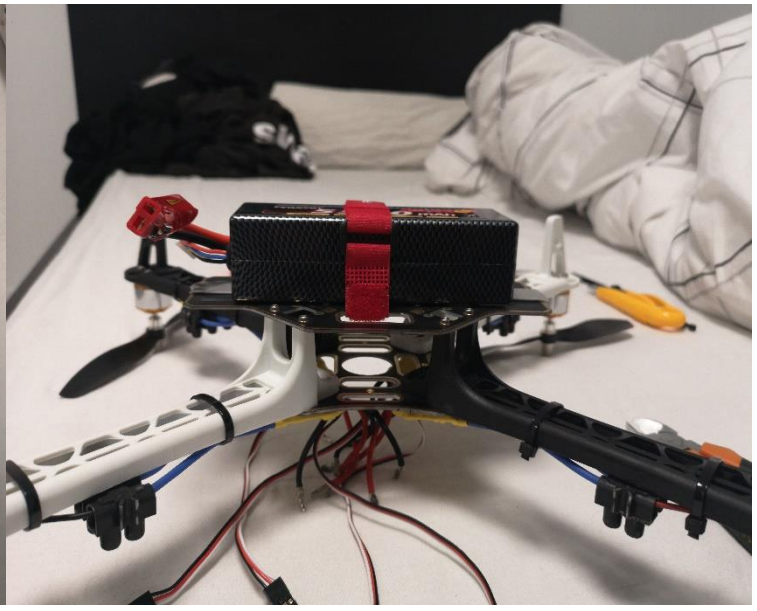
Par contre, je pensais pour mettre la batterie dans le petit espace entre le haut le bas du drone



Mais la batterie est trop grosse et je ne possède ni entretoise, ni vis suffisamment grande pour trouver augmenter l'espace. Je vais donc pour le moment la scotcher avec du « double face » en dessous, et la maintenir avec des colliers de serrage. Je mettrai donc l'Arduino et l'accéléromètre ici à la place de la batterie.

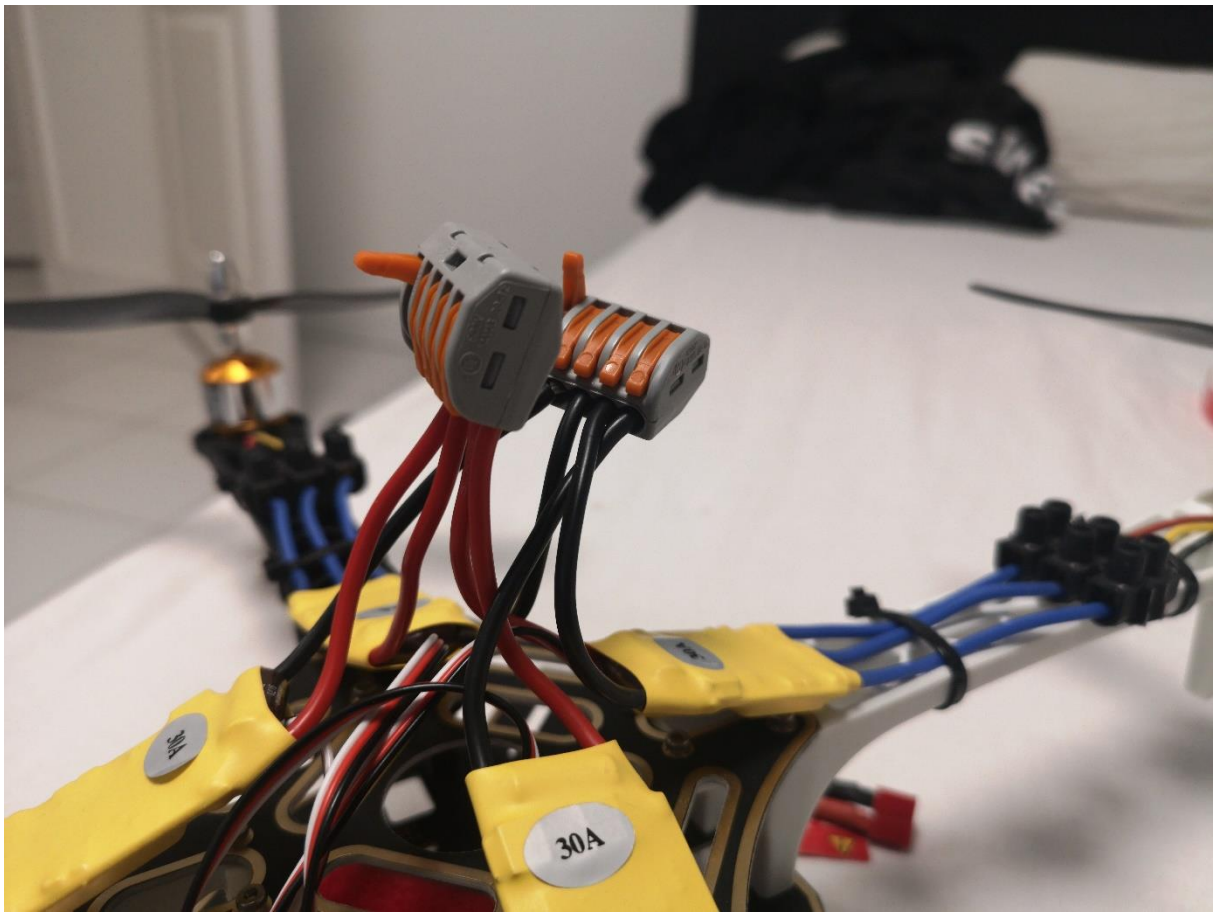
Voilà ce que cela donne :





Cela étant fait, nous allons regrouper les alimentations de chaque moteur afin de s'y retrouver plus facilement.

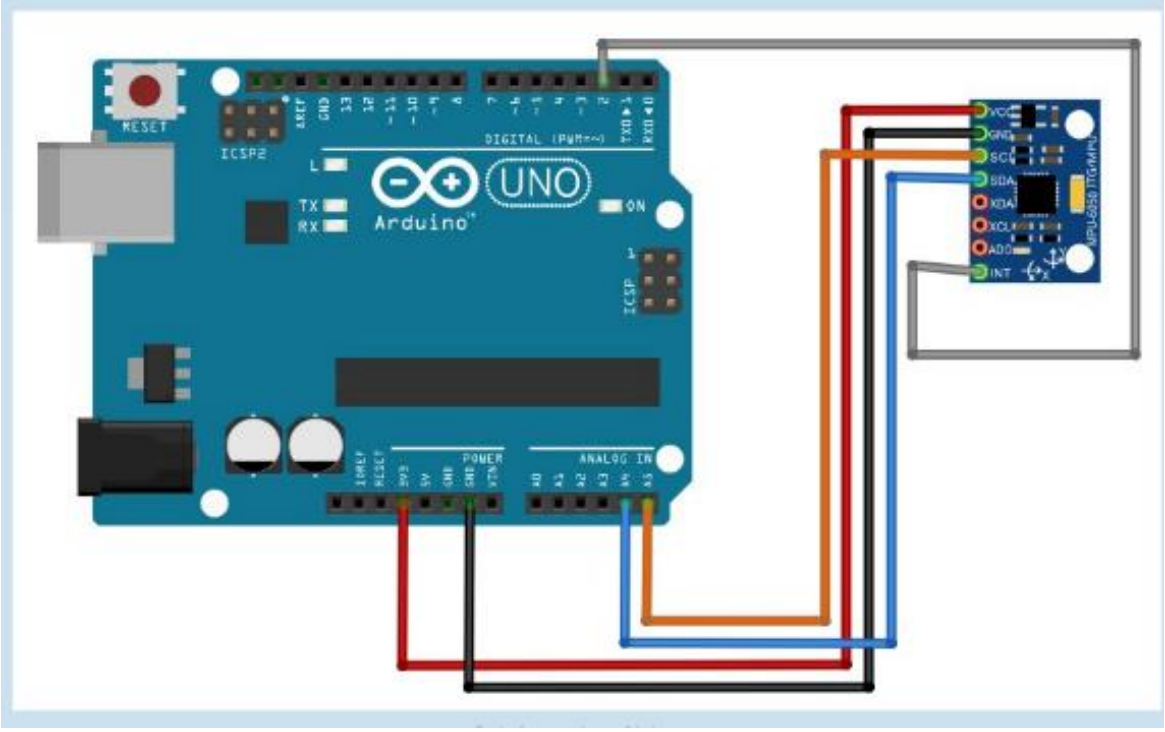
Tous les fils rouges dans une connecteur rapide, tous les fils noirs dans un autre :



J'aurai plus qu'a brancher la batterie aux connecteurs pour alimenter aisément chaque moteur + ESC.

Rédigeons maintenant un code qui sera implémenter dans l'Arduino pour un test de stabilisation.

Il faut dans un premier temps comprendre comment utiliser l'accéléromètre MPU6050.



Nous utiliserons ces branchements-là. Les fils rouge et noir représentent l'alimentation, orange et bleu seront les valeurs que le capteur va nous transmettre (analogique donc) et le gris (interrupteur digital) est utilisé pour améliorer la précision.

Avant de pouvoir stabiliser notre drone, il faut calibrer une bonne fois pour toute notre capteur MPU6050.

Pour ce faire, nous allons nous servir de librairies (de Luis Rodenas) existantes sur GitHub : IC2dev et MPU6050 (je les aie également mis sur mon GitHub).

Le code est le suivant, il est très bien expliqué en anglais alors je ne vais pas tout détailler.

Je vais simplement décrire le principe dans les grandes lignes.

```
// I2Cdev and MPU6050 must be installed as libraries
#include "I2Cdev.h"
#include "MPU6050.h"
#include "Wire.h"

//////////////////////////////////   CONFIGURATION   /////////////////////////////
//Change this 3 variables if you want to fine tune the skecth to your needs.
int buffersize=1000;     //Amount of readings used to average, make it higher to get more precision but sketch will be slower  (default:1000)
int acel_deadzone=8;     //Acelerometer error allowed, make it lower to get more precision, but sketch may not converge  (default:8)
int giro_deadzone=1;     //Giro error allowed, make it lower to get more precision, but sketch may not converge  (default:1)
```

```
// default I2C address is 0x68
// specific I2C addresses may be passed as a parameter here
// AD0 low = 0x68 (default for InvenSense evaluation board)
// AD0 high = 0x69
//MPU6050 accelgyro;
MPU6050 accelgyro(0x68); // <-- use for AD0 high

int16_t ax, ay, az,gx, gy, gz;

int mean_ax,mean_ay,mean_az,mean_gx,mean_gy,mean_gz,state=0;
int ax_offset,ay_offset,az_offset,gx_offset,gy_offset,gz_offset;

/////////////////////////////////   SETUP   /////////////////////////////////
void setup() {
  // join I2C bus (I2Cdev library doesn't do this automatically)
  Wire.begin();
  // COMMENT NEXT LINE IF YOU ARE USING ARDUINO DUE
  TWBR = 24; // 400kHz I2C clock (200kHz if CPU is 8MHz). Leonardo measured 250kHz.

  // initialize serial communication
  Serial.begin(115200);

  // initialize device
  accelgyro.initialize();

  // wait for ready
  while (Serial.available() && Serial.read()); // empty buffer
  while (!Serial.available()){
    Serial.println(F("Send any character to start sketch.\n"));
    delay(1500);
  }
  while (Serial.available() && Serial.read()); // empty buffer again

  // start message
  Serial.println("\nMPU6050 Calibration Sketch");
  delay(2000);
  Serial.println("\nYour MPU6050 should be placed in horizontal position, with package letters facing
up. \nDon't touch it until you see a finish message.\n");
  delay(3000);
  // verify connection
  Serial.println(accelgyro.testConnection() ? "MPU6050 connection successful" : "MPU6050 connection
failed");
  delay(1000);
  // reset offsets
  accelgyro.setXAccelOffset(0);
```

```
  accelgyro.setYAccelOffset(0);
  accelgyro.setZAccelOffset(0);
  accelgyro.setXGyroOffset(0);
  accelgyro.setYGyroOffset(0);
  accelgyro.setZGyroOffset(0);
}

////////////////////////////////   LOOP   ////////////////////////////////
void loop() {
  if (state==0){
    Serial.println("\nReading sensors for first time...");
    meansensors();
    state++;
    delay(1000);
  }

  if (state==1) {
    Serial.println("\nCalculating offsets...");
    calibration();
    state++;
    delay(1000);
  }

  if (state==2) {
    meansensors();
    Serial.println("\nFINISHED!");
    Serial.print("\nSensor readings with offsets:\t");
    Serial.print(mean_ax);
    Serial.print("\t");
    Serial.print(mean_ay);
    Serial.print("\t");
    Serial.print(mean_az);
    Serial.print("\t");
    Serial.print(mean_gx);
    Serial.print("\t");
    Serial.print(mean_gy);
    Serial.print("\t");
    Serial.println(mean_gz);
    Serial.print("Your offsets:\t");
    Serial.print(ax_offset);
    Serial.print("\t");
    Serial.print(ay_offset);
    Serial.print("\t");
    Serial.print(az_offset);
    Serial.print("\t");
```

```
    Serial.print(gx_offset);
    Serial.print("\t");
    Serial.print(gy_offset);
    Serial.print("\t");
    Serial.println(gz_offset);
    Serial.println("\nData is printed as: acelX acelY acelZ giroX giroY giroZ");
    Serial.println("Check that your sensor readings are close to 0 0 16384 0 0 0");
    Serial.println("If calibration was succesful write down your offsets so you can set them in your
projects using something similar to mpu.setXAccelOffset(youroffset)");
    while (1);
  }
}


///////////////////////////////////   FUNCTIONS   ///////////////////////////////////
void meansensors(){
  long i=0,buff_ax=0,buff_ay=0,buff_az=0,buff_gx=0,buff_gy=0,buff_gz=0;


  while (i<(buffersize+101)){
    // read raw accel/gyro measurements from device
    accelgyro.getMotion6(&ax, &ay, &az, &gx, &gy, &gz);


    if (i>100 && i<=(buffersize+100)){ //First 100 measures are discarded
      buff_ax=buff_ax+ax;
      buff_ay=buff_ay+ay;
      buff_az=buff_az+az;
      buff_gx=buff_gx+gx;
      buff_gy=buff_gy+gy;
      buff_gz=buff_gz+gz;
    }
    if (i==(buffersize+100)){
      mean_ax=buff_ax/buffersize;
      mean_ay=buff_ay/buffersize;
      mean_az=buff_az/buffersize;
      mean_gx=buff_gx/buffersize;
      mean_gy=buff_gy/buffersize;
      mean_gz=buff_gz/buffersize;
    }
    i++;
    delay(2); //Needed so we don't get repeated measures
  }
}


void calibration(){
  ax_offset=-mean_ax/8;
  ay_offset=-mean_ay/8;
  az_offset=(16384-mean_az)/8;
```

```
  gx_offset=-mean_gx/4;
  gy_offset=-mean_gy/4;
  gz_offset=-mean_gz/4;
  while (1){
    int ready=0;
    accelgyro.setXAccelOffset(ax_offset);
    accelgyro.setYAccelOffset(ay_offset);
    accelgyro.setZAccelOffset(az_offset);

    accelgyro.setXGyroOffset(gx_offset);
    accelgyro.setYGyroOffset(gy_offset);
    accelgyro.setZGyroOffset(gz_offset);

    meansensors();
    Serial.println("...");

    if (abs(mean_ax)<=acel_deadzone) ready++;
    else ax_offset=ax_offset-mean_ax/acel_deadzone;

    if (abs(mean_ay)<=acel_deadzone) ready++;
    else ay_offset=ay_offset-mean_ay/acel_deadzone;

    if (abs(16384-mean_az)<=acel_deadzone) ready++;
    else az_offset=az_offset+(16384-mean_az)/acel_deadzone;

    if (abs(mean_gx)<=giro_deadzone) ready++;
    else gx_offset=gx_offset-mean_gx/(giro_deadzone+1);

    if (abs(mean_gy)<=giro_deadzone) ready++;
    else gy_offset=gy_offset-mean_gy/(giro_deadzone+1);

    if (abs(mean_gz)<=giro_deadzone) ready++;
    else gz_offset=gz_offset-mean_gz/(giro_deadzone+1);

    if (ready==6) break;
  }
}
```

Alors, attaquons les explications…

Tout d'abord le MPU6050 fonctionne selon un concept de « cases » dans lesquels on va venir piocher les infos. Ces « cases » sont désignées par leurs « noms » dans notre cas par exemple ça sera 0x68. C'est le concept de I2C.

Ce n’est pas un fonctionnement avec lequel je suis très à l’aise alors je ne vais pas trop modifier ces valeurs par peur de modifications irréversible du matériel.

Après, c’est comme d’habitude : on initialise une vitesse de discussion et on va lancer le programme uniquement lorsque l’utilisateur va le demander.

Le principe du code étant de réaliser un très grand nombre de mesure sur surface plane afin de déterminer une inclinaison de référence.

Le code nous renverra par la suite des valeurs qu’il faudra noter avec soin car elles sont propres à notre configuration.

Essayons !

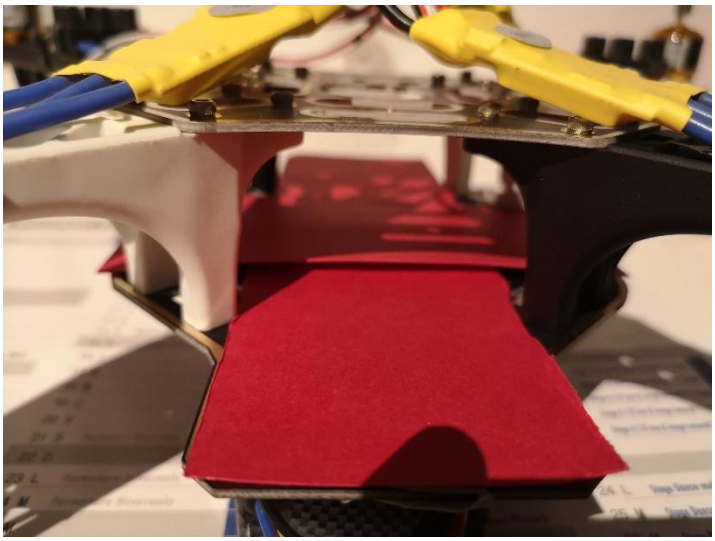
Il faut implémenter ce code dans l’Arduino et le fixer sur le drone.

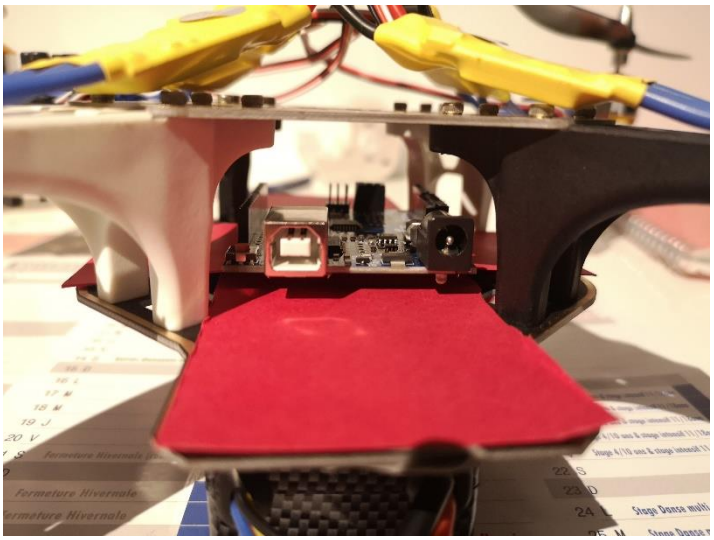
(Je ne sais pas si j’aurai le temps de finir le test avant la date de rendu du rapport, alors si je n’ai pas le temps, la suite figurera dans le rapport suivant. De plus je vais essayer de le filmer.)

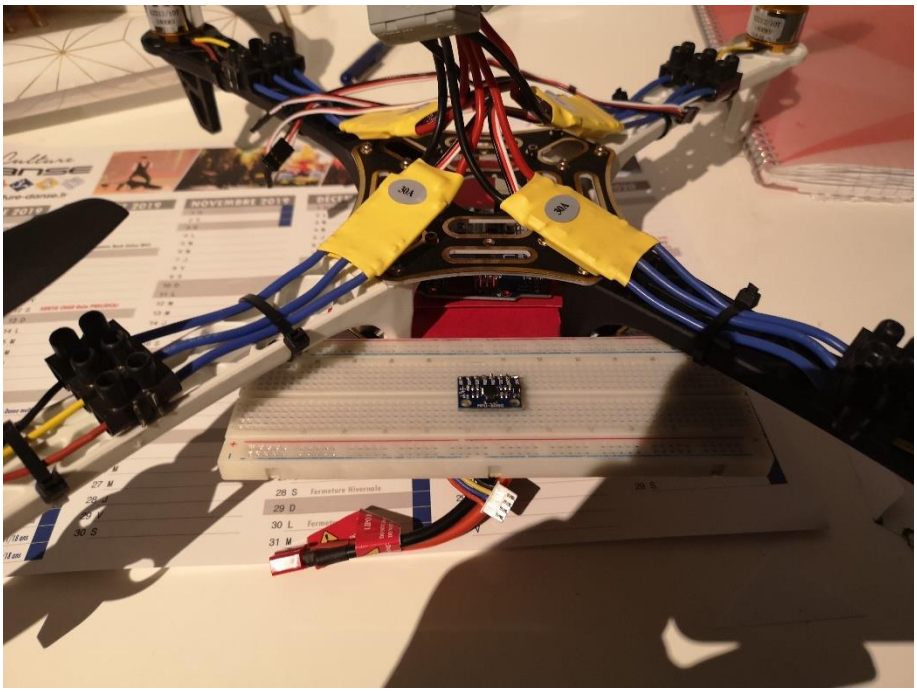
Je dois m’équiper d’un niveau à bulle afin de calibrer le plus précisément possible le capteur.

Je fixe également l’Arduino et le capteur au centre du drone, j’ai mis du carton sur la surface du châssis afin d’éviter les courts circuits etc… car il est en parti en métal/ fer/ aluminium ou je sais quoi (on n’est jamais trop prudent).

Ça donne ça :







C’est moche, mais ça va me permettre de tester.

IMPREVU : mon ordinateur portable à décider de ne plus reconnaitre la carte lorsque je la branche sur un port. J’ai donc téléverser le code depuis mon ordi fixe, mais le câble me permettant de relier le drone à l’ordinateur fixe lors de l’étape de calibrage n’est pas assez long, je dois donc attendre lundi prochain que mon père essaye de m’obtenir un nouvel ordi portable depuis son bureau. Je ne peux pas récupérer les valeurs d’offset précises. Je sais que cette manipulation fonctionne car je l’avais testé la semaine dernière avec des valeurs non précises. On peut d’ailleurs visualiser graphiquement ces résultats avec le code Visualisation3d sur Processing que j’ai mis sur GitHub.

Je suis donc contraint de m’arrêter ici en attendant un nouveau pc portable.

Je continuerai donc dans le prochain rapport.