Rapport de projet :

Réalisation d'un drone à base de la carte programmable « Arduino »

Adnane LAMNAOUAR
Khaoula ELMIMOUNI
Hafsa MAKBOUL
Ikram TABEK
Nadia AIT HMAD

AEROLEC AIAC
BUILD YOUR NEXT WAVE

Table des matières

Table des figures

Principales notations:

UAV: Unmanned Aerial Vehicle

ESC: Electronic Speed Controller

PWM: Pulse Width Modulation

Joystick: Manette

Throttle: Pédale à gaz

Pitch: Tangage

Roll: Roulis

Yaw: Lacet

Li-Po: Lithium- Polymer

3S: 3 cellules

FC : Flight Controller

PID : Correcteur proportionnel Intégral Dérivé

Introduction générale :

Objectifs :

Dans ce projet, nous allons faire la conception et la réalisation d'un drone à base de la carte électronique programmable Arduino.

Le contrôleur du vol va être à base d'Arduino, ainsi que le récepteur et la télécommande.

Le but de ce projet et de mettre en œuvre l'importance du code embarqué dans la carte du contrôleur de vol, et expliciter les différentes parties construisant un drone télécommandé.

Organisation du travail :

Nous allons réaliser un émetteur avec Arduino et son récepteur. Nous allons par la suite faire une réalisation du controleur du vol avec Arduino en se concentrant d'une part sur la part matérielle et sur la partie logicielle d'autre part.

Partie I : Transmission-Réception :

Introduction :

Dans cette partie, nous nous basons sur la réalisation d'un système Transmission-Réception primaire.

Emetteur :

Notre but, est de réaliser un système capable d'envoyer quatre informations pour diriger le drone :

- Throttle : Equivalent à la pédale à gaz
- Pitch : Tangage
- Roll : Roulis
- Yaw : Lacet

Figure 1 Les trois angles d'un drone dans l'espace

Partie Matériel :
Nous utiliserons le matériel suivant :

- 1 x Arduino Nano
- PA Module sans fil
- 1 x NRF24L01 Module sans fil
- 2 x Joystick Arduino
- 2 x Condensateurs 100uF (16V ou plus)

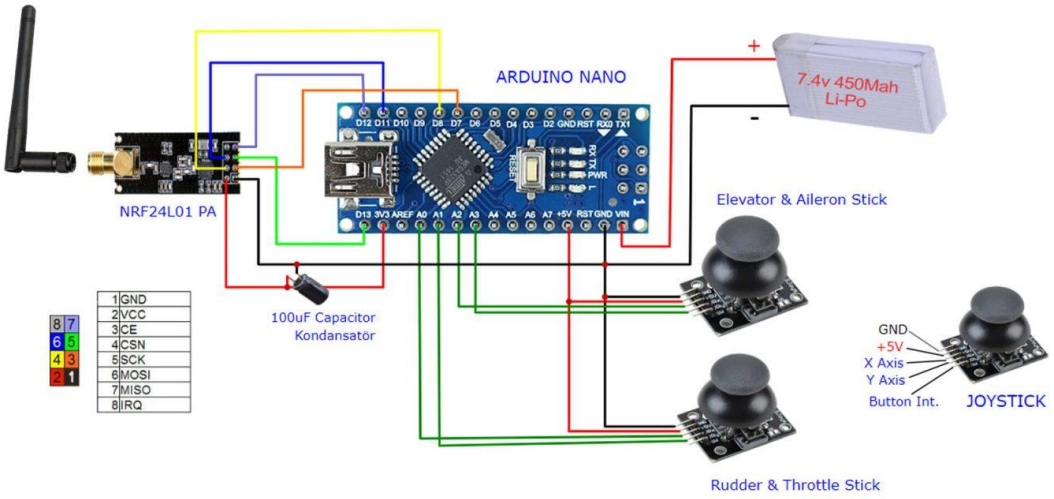


Figure 2: Le schéma de l'émetteur

En premier temps, on a basé notre étude et notre premier montage sur un avion télécommandé. Ce site est : https://www.rcpano.net/2020/02/17/simple-and-cheap-radio-control-making-for-rc-models-diy-rc/

Partie Logiciel :

Le code de cette partie (voir la figure 3) a pour but d'envoyer une structure en langage informatique Arduino basé sur le langage C++ et traduite après du domaine informatique au domaine des signaux.

Le signal envoyé (respectivement la structure informatique) contiendra les 4 informations nécessaires.

Notre code va récupérer les valeurs issues des joysticks que nous avons trouvé après l'expérience comprises entre 12 et 1020. Quand les joysticks sont dans leurs positions initiales nous trouvons une valeur d'entrée de 520.

Tous ces valeurs sont transformés entre 0 et 255 à l'aide de la fonction map() du langage Arduino utilisée à son tour par la fonction que nous avons défini : mapJoysticksValues() .

Pour prendre en compte ces informations, et pour plus de précision, le « mappage » a été divisé en deux parties et centrées finalement autour de 127.

Notre module NRF24L est un émetteur de 8 bits avec 32 canaux. Donc il est capable d'envoyer 32 informations in dépendantes prenants 256 valeurs différentes. C'est la cause derrière la conversion des valeurs du joysticks à une plage de [0, 255].

Le signal est finalement envoyé (dernière ligne du code) en une forme de structure mathématique « struct ». Cette dernière est stockée dans la mémoire. Pour l'envoyer, il est nécessaire donc d'indiquer son adresse et sa taille. La méthode utilisée pour envoyer cette structure est « write » de la bibliothèque « radio.h ».

emeteur_drone

```
// 4 Channel Transmitter |
#include <SPI.h>
#include <nRF24L01.h>
#include <RF24.h>
const uint64_t pipeOut = 0xE9E8F0F0E1LL;   //IMPORTANT: The same as in the receiver 0xE9E8F0F0E1LL
RF24 radio(7, 8); // select CE,CSN pin
struct Signal {
byte throttle;
byte pitch;
byte roll;
byte yaw;
};
Signal data;
void ResetData()
{
data.throttle = 127; // Motor Stop (254/2=127)
data.pitch = 127; // Center |  (Signal lost position)
data.roll = 127; // Center |  (Signal lost position
data.yaw = 127; // Center |  (Signal lost position
}
void setup()
{
//Start everything up
radio.begin();
radio.openWritingPipe(pipeOut);
radio.stopListening(); //start the radio comunication for Transmitter
ResetData();
}
// Joystick center and its borders
```

```
// Joystick center and its borders
int mapJoystickValues(int val, int lower, int middle, int upper, bool reverse)
{
val = constrain(val, lower, upper);
if ( val < middle )
val = map(val, lower, middle, 0, 128);
else
val = map(val, middle, upper, 128, 255);
return ( reverse ? 255 - val : val );
}
void loop()
{
// Control Stick Calibration
// Setting may be required for the correct values of the control levers.
data.throttle = mapJoystickValues( analogRead(A0), 524, 524, 1015, true );
data.roll = mapJoystickValues( analogRead(A1), 12, 524, 1020, true );      // "true" or "false" for servo direction |
data.pitch = mapJoystickValues( analogRead(A2), 12, 524, 1020, true );     // "true" or "false" for servo direction |
data.yaw = mapJoystickValues( analogRead(A3), 12, 524, 1020, true );       // "true" or "false" for servo direction |
radio.write(&data, sizeof(Signal));
}
```

Figure 3: Le code utilisé dans la partie Emetteur

Récepteur :

Le but est de concevoir un système capable de recevoir le signal envoyé par le système de transmission vu dans la partie précédente et d'en extraire les quatre informations. Nous allons par la suite les convertir en un signal PWM et le visualiser pour l'exploiter après.

Cette partie est équivalente à la réalisation d'un récepteur radio à PWM.

Figure 4 Récepteur à PWM Flysky FS-iA6

D'ailleurs, pour visualiser le signal nous n'utiliserons pas des contrôleurs de vitesse électriques (ESC) avec les moteurs sans ballets, mais il est plus économique, simple et moins risqué d'utiliser des servomoteurs.

En effet, la majorité des ESC, comme nous utiliserons dans ce projet sont commandé avec un signal PWM de largeur d'impulsion comprise entre 1000 µs et 2000 µs, tout comme un servomoteur ordinaire.

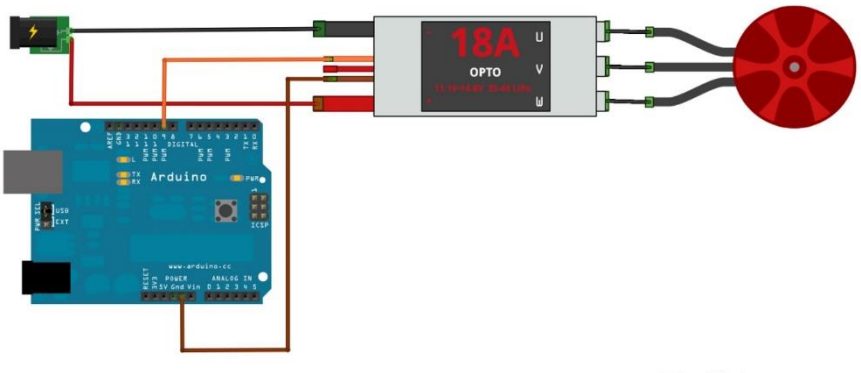


Figure 5: Le branchement d'un ESC avec Arduino

Partie matériel :
Nous utiliserons le matériel suivant :

- 1 x NRF24L01 : Nous utiliserons ce module dual en mode récepteur
- Arduino nano : Pour réaliser les calculs et gérer les entrées et les sorties
- 4 x Servomoteur : Jouent le rôle des actionneurs
- 1x Condensateur 100 µF
- Batterie : Pour l'alimentation

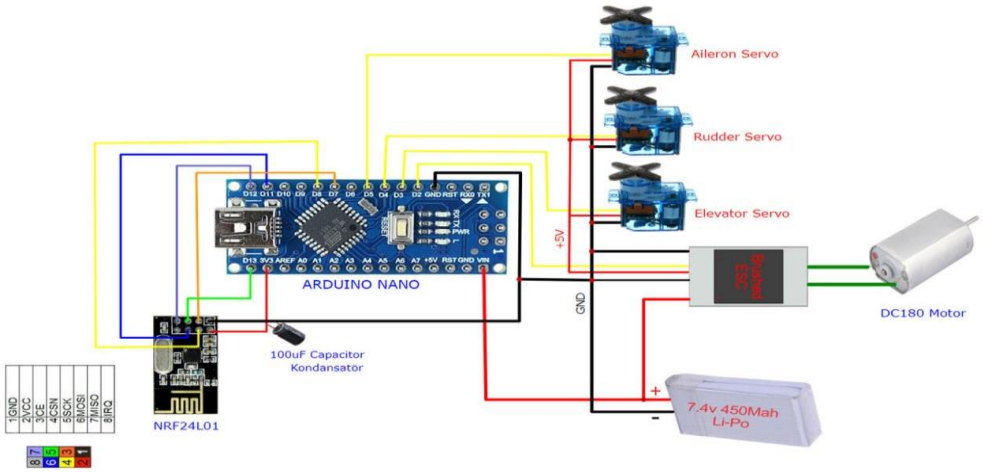


Figure 6: Montage du récepteur avec 3 servomoteurs et un ESC pour illustration

Partie logiciel :

Comme nous l'avons indiqué avant, cette étape consiste à recevoir le signal, et extraire de la structure informatique les quatre informations. Les informations sont traduites en PWM à travers les quatre pins de l'Arduino Nano :

- D2 pour le Throttle
- D3 pour le Pitch
- D4 pour le Roll
- D5 pour le Yaw

En utilisant le moniteur série avec la méthode « write » de la bibliothèque Serial, nous pouvons suivre et évaluer le bon fonctionnement de notre système.

recepteur_drone §

```
//  4 Channel Receiver
//  PWM output on pins D2, D3, D4, D5
#include <SPI.h>
#include <nRF24L01.h>
#include <RF24.h>
#include <Servo.h>
#include<string.h>
int ch_width_1 = 0;
int ch_width_2 = 0;
int ch_width_3 = 0;
int ch_width_4 = 0;
Servo ch1;
Servo ch2;
Servo ch3;
Servo ch4;
struct Signal {
byte throttle;
byte pitch;
byte roll;
byte yaw;
};
Signal data;
const uint64_t pipeIn = 0xE9E8F0F0E1LL;
RF24 radio(7, 8);
void ResetData()
{
// Define the inicial value of each data input.
// The middle position for Potenciometers. (254/2=127)
data.throttle = 127; // Motor Stop
data.pitch = 127;  // Center
data.roll = 127;   // Center
data.roll = 127;   // Center
data.yaw = 127;   // Center
}
void setup()
{ Serial.begin(9600);
  //Set the pins for each PWM signal
  ch1.attach(2);
  ch2.attach(3);
  ch3.attach(4);
  ch4.attach(5);
  //Configure the NRF24 module
  ResetData();
  radio.begin();
  radio.openReadingPipe(1,pipeIn);

  radio.startListening(); //start the radio comunication for receiver
}
unsigned long lastRecvTime = 0;
void recvData()
{
while ( radio.available() ) {
radio.read(&data, sizeof(Signal));
lastRecvTime = millis();   // receive the data
}
}
void loop()
{
recvData();
unsigned long now = millis();
if ( now - lastRecvTime > 1000 ) {
ResetData(); // Signal lost.. Reset data
```

```
ResetData(); // Signal lost.. Reset data
}
Serial.print("throttle=");
Serial.println(data.throttle);
Serial.print("pitch (elevator) =");
Serial.println(data.pitch);
Serial.print("roll (rudder)=");
Serial.println(data.roll);
Serial.print("yaw (aileron)=");
Serial.println(data.yaw);
Serial.println("--------------------");
//delay(1000);

ch_width_1 = map(data.throttle, 0, 255, 1000, 2000);     // pin D2 (PWM signal)
ch_width_2 = map(data.pitch,    0, 255, 1000, 2000);     // pin D3 (PWM signal)
ch_width_3 = map(data.roll,     0, 255, 1000, 2000);     // pin D4 (PWM signal)
ch_width_4 = map(data.yaw,      0, 255, 1000, 2000);     // pin D5 (PWM signal)
// Write the PWM signal
ch1.writeMicroseconds(ch_width_1);
ch2.writeMicroseconds(ch_width_2);
ch3.writeMicroseconds(ch_width_3);
ch4.writeMicroseconds(ch_width_4);
}
```

Figure 7: Le code embarqué dans la carte Arduino du récepteur

Conclusion :

Pendant la phase du test, nous avons rencontré assez de dysfonctionnements. Nous avons identifié ces problèmes comme étant causés par le contact non parfait et instable assuré par des câbles et les plaques d'essai. Un soudage sur une carte PCB révèle donc une solution considérable.

Nous avons fini les expériences de l'émetteur et récepteur, en bien maitrisant et manipulant le code et le matériel selon le besoin dans 30 jours environ, de 15 avril 2022 au 18 avril 2022.

Partie II : Contrôleur du vol

Introduction :

Le contrôleur du vol ou le Flight Controler en anglais est la partie la plus importante de chaque drone. C'est la partie responsable de l'équilibre de l'aéronef et le mouvement de cette dernière en traduisant les commandes issues du récepteur ou en exécutant celles venant d'un calculateur puissant comme le Raspberry pi, ou même en exécutant directement, si nous avons accès au code source (dans le cas des contrôleurs de vol à source ouverte) un programme de vol pour exécuter une mission.

Contrôleur du vol :

Partie materiel :

Nous utiliserons le matériel suivant :

- Arduino Nano : C'est la partie qui gérera les entrées issues du récepteur et les sorties qui serons les ESC ou les servomoteurs.
- MPU6050 Gyroscope – Accéléromètre : C'est le capteur qui va nous permettre de réaliser une boucle de retour afin d'asservir notre système.
- Battery Li-Po 3S : Pour l'alimentation

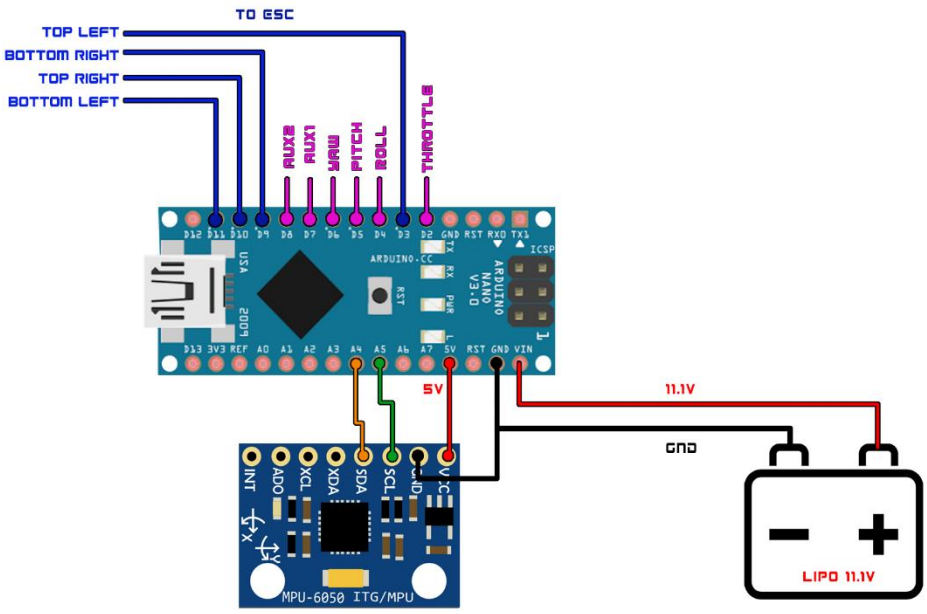


Figure 8: Branchement du controleir du vol

Le site de « Electronoobs » donne assez d'informations et partage ses expériences dans la réalisation d'un projet pareil que le nôtre:
http://electronoobs.com/eng_robotica_tut5_3.php

A titre d'illustration, la figure ci-dessous montre le montage que nous envisageons réaliser. Elle montre à gauche la partie « Récepteur » et à droite la partie « Contrôleur du vol ».

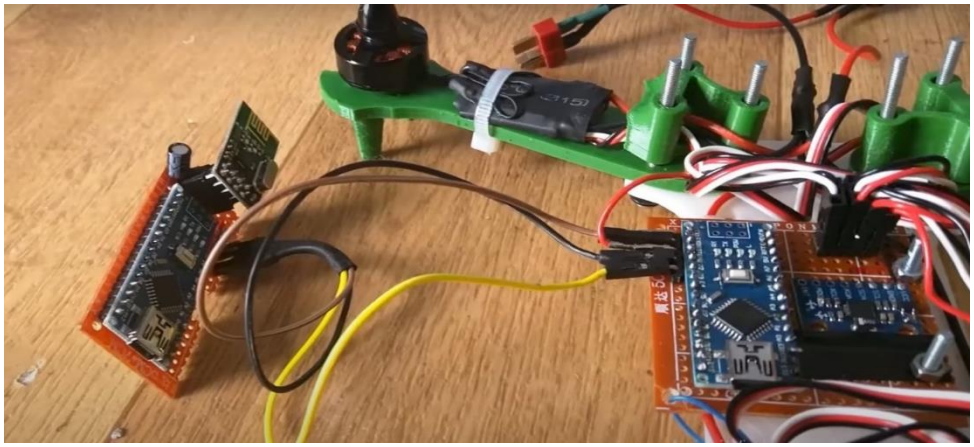


Figure 9: Illustration du montage final du site "Electroobs"

Partie logiciel :

Le firmware utilisé dans ce projet est de source ouverte. Il s'appelle MultiWii-2.4 .

Nous téléchargeons les fichiers nécessaires à partir du site :
http://www.multiwii.com/wiki/index.php?title=Main_Page

Le fichier compressé qui s'appelle MultiWii_2_4 se compose de

Ce site contient des informations sur la partie matérielle et la partie logicielle. Il contient aussi un lien d'un forum ou nous pouvons trouver des codes, des questions et des réponses et des idées diverses sur des projets utilisant ce programme MultiWii, que ce soient pour des avions, drones et même des robots non volant en intégrant de différentes technologies comme celle du Raspberry pi.

Ce code permet d'utiliser plusieurs fonctionnalités à notre robot, comme le Bluetooth et le GPS.

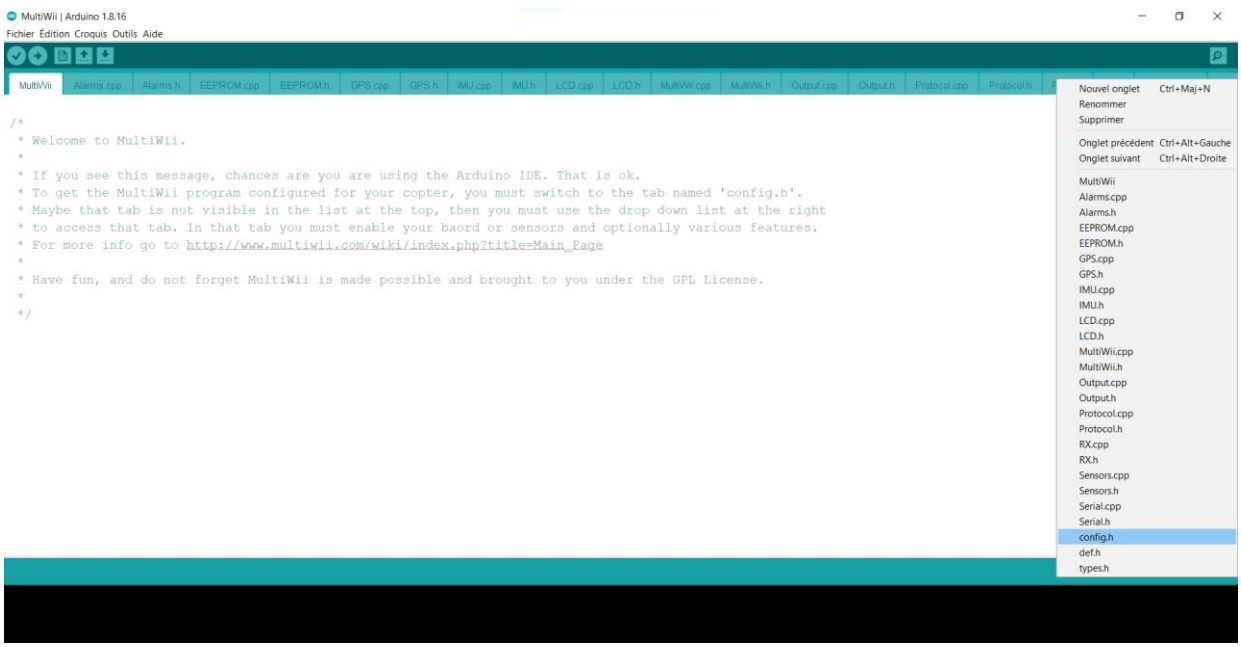


Figure 10: L'interface montrée en cliquant sur le fichier Arduino principal « MultiWii.ino »

La prochaine étape et de se diriger vers me fichier config.h et décommenter QuadX pour pouvoir utiliser le code pour notre drone quadricoptère en forme X.

```
/**************************    The type of multicopter    ***************************/
  //#define GIMBAL
  //#define BI
  //#define TRI
  //#define QUADP
  #define QUADX
  //#define Y4
  //#define Y6
  //#define HEX6
  //#define HEX6X
  //#define HEX6H  // New Model
  //#define OCTOX8
  //#define OCTOFLATP
  //#define OCTOFLATX
  //#define FLYING_WING
  //#define VTAIL4
  //#define AIRPLANE
  //#define SINGLECOPTER
  //#define DUALCOPTER
  //#define HELI_120_CCPM
  //#define HELI_90_DEG
```

Figure 11: Les types disponibles de robots supportés par MultiWII

L'ouverture des fichiers contenant le code dans Visual Studio Code va nous faciliter beaucoup sa lecture et sa compréhension. Cela en visitant le fichier d'extension « .h » et pour chaque attribut ou méthode, on visite son code détaillé se trouvant dans le fichier correspondant d'extension « .cpp » en maintenant « ctrl » et cliquer.

Après la première configuration du code selon nos besoins nous lançons le programme MultiWiiConf.exe pour continuer la configuration en suivant le chemin relatif : MultiWii_2_4/MiltiWiiConf/application.windows64/

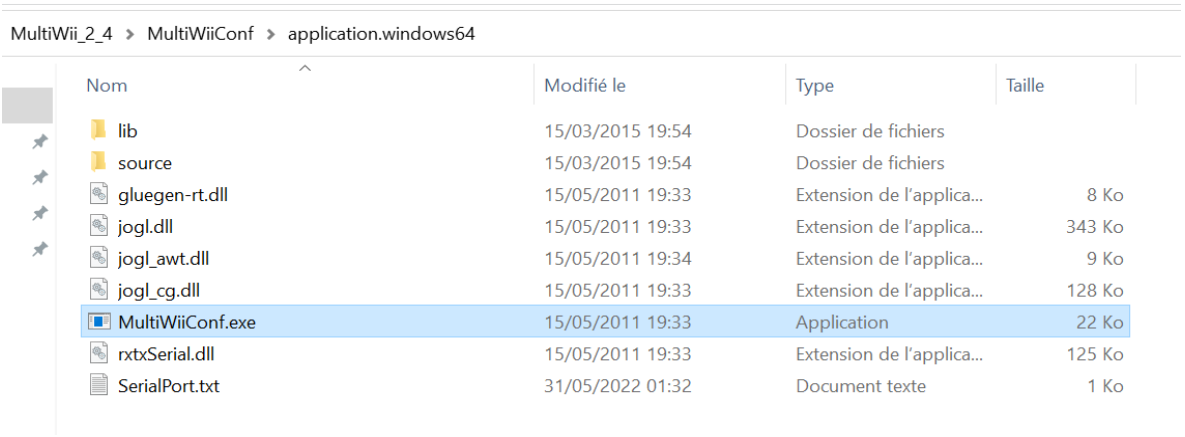


Figure 12: L'application de configuration MuiltiWii

Après la réalisation du montage, on connecte la carte Arduino à l'aide d'un câble USB avec notre ordinateur. En lançant cette application, une interface graphique développée en langage java émerge devant nous.

Nous choisissons le port série convenable, puis nous cliquons sur START.

Nous cliquons après sur CALIB_MAG pour choisir automatiquement les valeurs de P, I et D et pour calibrer notre FC automatiquement.

Après la calibration de notre FC, il est temps de l'activer : C'est « armer » le drone. Nous armons notre drone en maintenant le joystick gauche à droite et à sa limite (cela renvoie à mettre le signal correspondant au Yaw à 1000 µs) pour 330 ms en laissant le joystick à droite à la position minimale (cela renvoie à un signal de Throttle de 1000 µs).

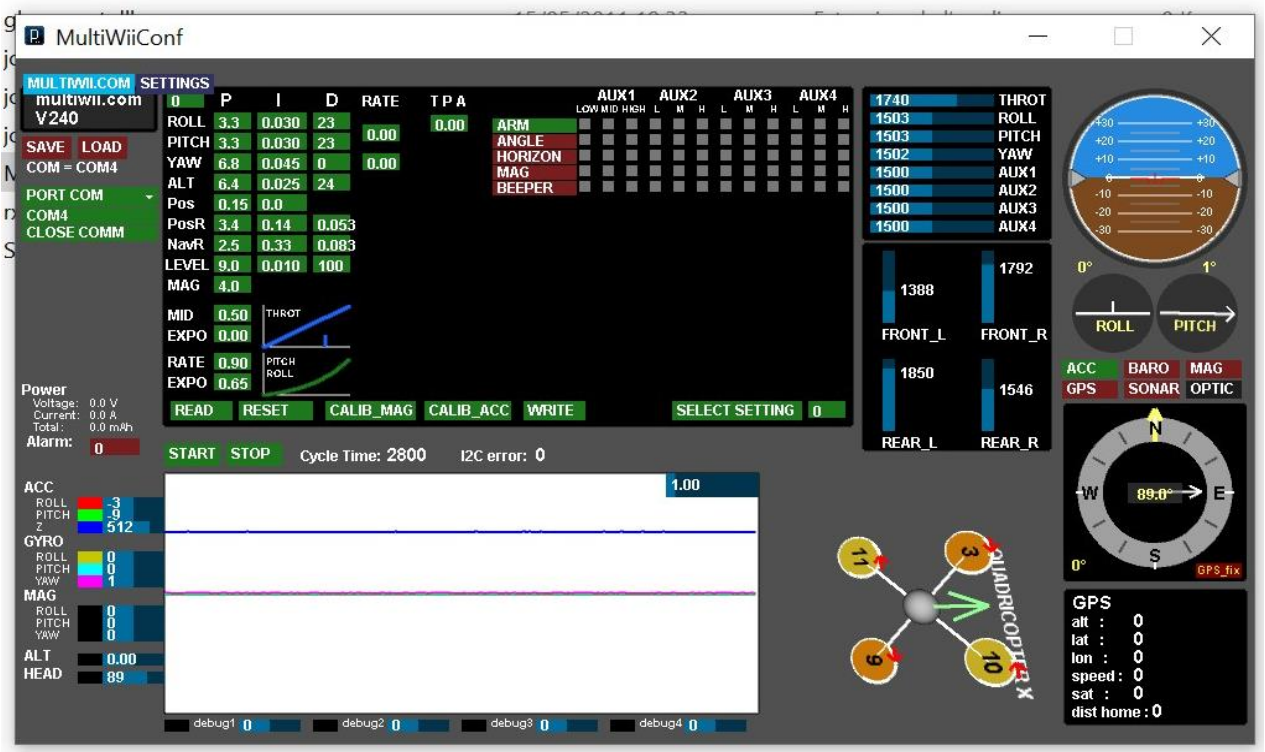
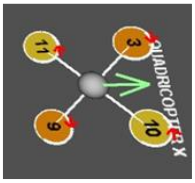
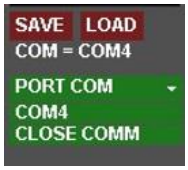


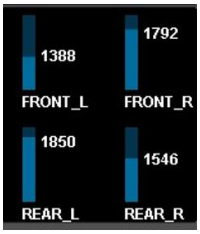
Figure 13: Interface graphique de configuration du drone

L'interface affiche plusieurs informations importantes issues de notre contrôleur du vol comme :

- L'orientation du contrôleur du vol dans l'espace
-







Nous allons démarrer notre test préliminaire, mais cette fois nous n'utiliserons pas le récepteur et l'émetteur que nous avons conçu à l'aide d'Arduino dans la première partie. L'utilisation de l'émetteur FS-i6 de l'entreprise « Flysky » et son récepteur était faite pour éviter les problèmes liés à l'instabilité, et pour nous concentrer sur la partie du contrôleur du vol (FC). L'utilisation d'un servomoteur est comme nous avons précédemment évoqué, pour des raisons de simplicité et de sécurité.

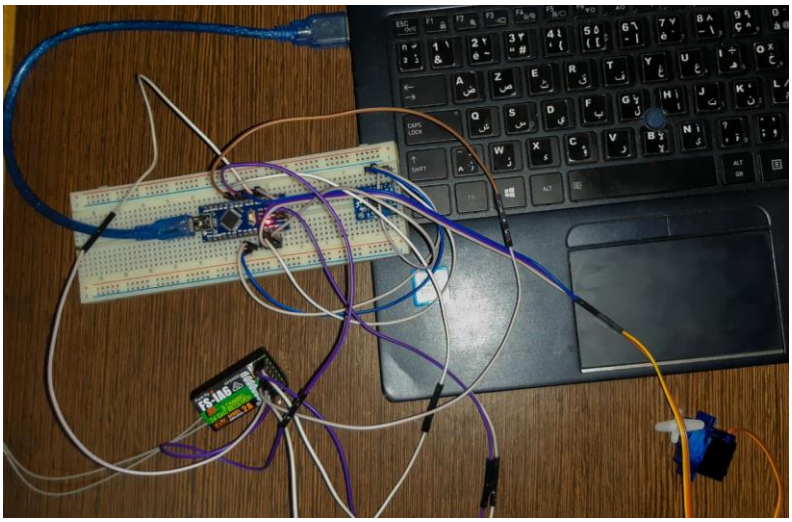


Figure 14: Le montage réaliser pour un test primaire du FC

Conclusion :

Cette partie nous a pris un temps plus grand que la précédente (1mois et demi environ) puisque le code de source ouverte n'est pas assez évident. Cette partie nous a permis de faire des recherches pour comprendre les différentes parties du contrôleur du vol en interne, ainsi que comment communiquer avec lui.

Nous pouvons lister les différents problèmes qui ont restés non résolus :

Problème de la calibration :

Notre ordinateur affiche que notre FC est en train de tourner même s'il est toujours en repos. Le problème persiste même après la répétition du processus e la calibration automatique du gyroscope et l'accéléromètre et même le changement du MPU6050 avec un autre. Ce problème cause l'instabilité de notre drone et le non équilibre. Cela peut être évalué au niveau du bloc du niveau des moteurs montrés sur l'interface graphique.

La solution envisagée :

Après une recherche, le problème provient probablement du capteur MPU6050, précisément de la valeur seuil de ce dernier appelée en anglais : offset. Pour résoudre le problème nous identifierons cette valeur seuil avec un test indépendant, puis nous ouvrons le code source de MultiWii et essayer de le modifier pour fonctionner correctement avec notre capteur.

Conclusion générale :

La prochaine partie consistera à résoudre le probleme du gyro et passer à la partie puissance pour test final.h

Ce travail a été le fruit d'une reflexion, recherche et passion. L'idée a débuté des années, mais à l'aide du cadre offert par le club Aerolec, nous avons trouvé l'opportunité de réaliser ce qu'était un « rêve ».