UNIVERSITÉ DE TECHNOLOGIE DE BELFORT-MONTBÉLIARD

**Tronc Commun- Projet Tuteuré d’IFD2**

Deuxième année

# 

# **Drone quadricoptère**

**Projet réalisé par:**

Victor MIRA

Antoine DUPAYRAT

Aymeric PERNOD

Catherine TURLE

**Sous la direction de:**

Nicolas GAUD

Pierre ROMET

**ANNÉE UNIVERSITAIRE 2019-2020**

**SOMMAIRE**

[**I. Présentation du projet** 3](#_Toc29717716)

[**II. La conception du mini drone initiale** 4](#_Toc29717717)

[**A.** **La télécommande** 4](#_Toc29717718)

[1. Fonctionnement de la télécommande 5](#_Toc29717719)

[2. Montage 5](#_Toc29717720)

[3. Codes 6](#_Toc29717721)

[4. Difficultés rencontrées 10](#_Toc29717722)

[**B.** **Le Drone** 11](#_Toc29717723)

[1. Le gyroscope 12](#_Toc29717724)

[a. Fonctionnement 12](#_Toc29717725)

[b. Montage 12](#_Toc29717726)

[c. Code (code basique repris d’un exemple de la librairie) 13](#_Toc29717727)

[d. Difficultés 14](#_Toc29717728)

[2. Les moteurs 15](#_Toc29717729)

[a. Montage 15](#_Toc29717730)

[b. Code 15](#_Toc29717731)

[c. Difficultés 16](#_Toc29717732)

[3. Mise en commun des composants 17](#_Toc29717733)

[a. Fonctionnement du drone dans sa globalité 17](#_Toc29717734)

[b. Montage 17](#_Toc29717735)

[c. Difficultés 18](#_Toc29717736)

[**III. Drone de M. Romet** 19](#_Toc29717737)

[**A.** **Fonctionnement des nouveaux moteurs et différence de fonctionnement avec le mini-drone** 20](#_Toc29717738)

[**B.** **Code permettant de faire tourner les 4 moteurs en même temps** 20](#_Toc29717739)

[**C. Etude du code du nouveau drone** 21](#_Toc29717740)

[**D. Difficultés rencontrées** 24](#_Toc29717741)

[**IV. Conclusion** 24](#_Toc29717742)

[**V. Annexes** 25](#_Toc29717743)

## **Présentation du projet**

Dans ce projet nous avions pour objectif de concevoir un mini drone capable de voler et pouvant être contrôler à distance. Pour ce faire, voici le matériel dont nous disposions :

* Une carte programmable nano Arduino
* Une carte programmable Adafruit Huzzah32 – ESP32 Feather
* Stepper pour moteurs DC FeatherWing pour cartes programmables Feather
* 2 modules nRF24L01+
* 2 joysticks (Analog 2-axis Thumb Joystick with Select Button)
* 1 gyroscope LSM9DSO
* 4 moteurs pour Crazyflie 2.0
* 4 hélices (Crazyflie Nano Quadcopter 4 x CW CCW Spare Propellers DIY)
* 2 batteries de 3,7V

Nous avons donc deux parties dans notre projet : **la télécommande**, qui servira au pilotage du drone. Celle-ci est composée de la nano Arduino, des deux joysticks et d’un des modules nRF (qui sert d’émetteur). Puis **le drone** qui sera composé du reste des composants ainsi que d’une armature en plexiglas.

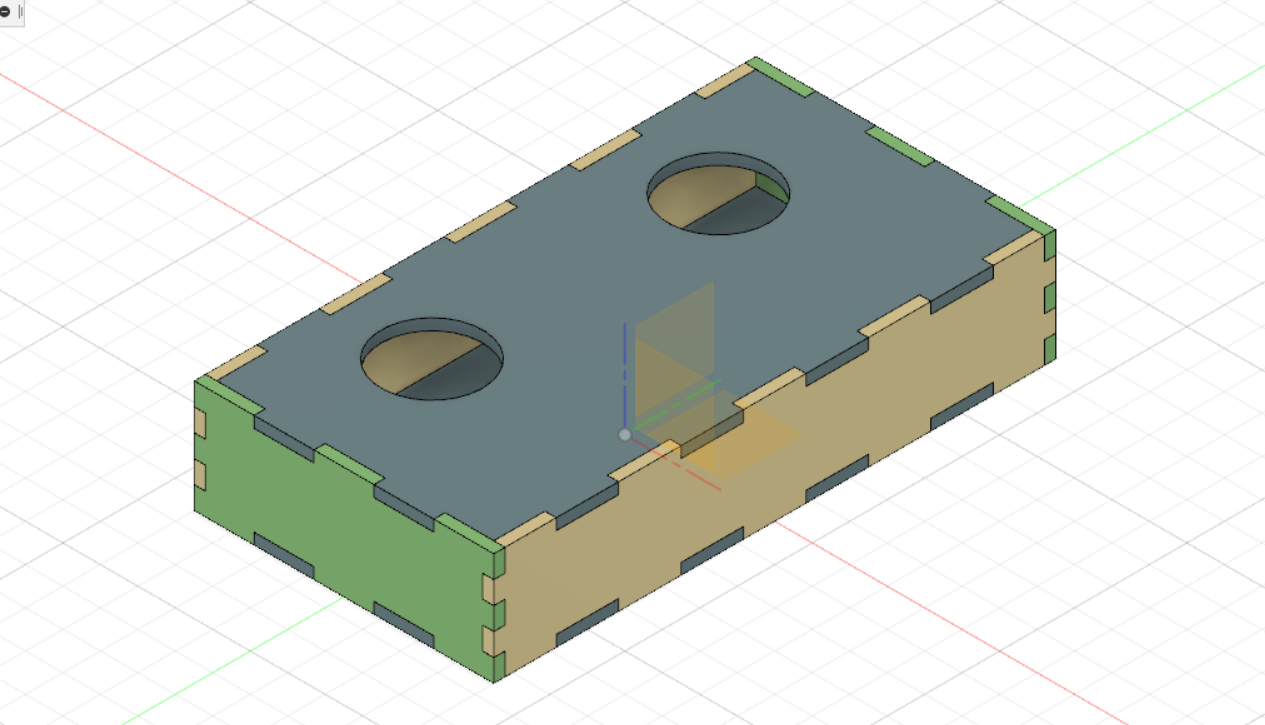
Cependant nous avons eu des difficultés à concevoir ce mini drone, difficultés que nous expliciterons plus tard. C’est pourquoi nous allons d’abord parler de notre tentative de conception de ce mini-drone, puis de notre travail sur le drone de M. Romet.

## **La conception du mini drone initiale**

### **La télécommande**

Vous trouverez ci-contre une photo montrant l’intérieur de la télécommande :

Voici le boitier de la télécommande, nous l’avons réalisé grâce au logiciel Autodesk Fusion 360. Nous avons simplement repris un modèle que nous avons légèrement modifié.

Voici ce que nous avions sur le logiciel :

#### **Fonctionnement de la télécommande**

Le fonctionnement de la télécommande est très simple. Une batterie alimente la carte Arduino nano à laquelle sont branchés les deux joysticks ainsi que le **module nRF**. Le principe consiste en l’acquisition des signaux analogiques des joysticks par la carte, que le module nRF de la télécommande va envoyer au module nRF attaché au drone.

Pour ce qui est du fonctionnement du transfert des informations, celui-ci est simple. On ouvre un **channel d’écriture (côté émetteur)** qui devra correspondre au **channel de lecture du second nRF receveur**, et l’on envoie nos informations dessus. L’émetteur écrit sur le channel tandis que le receveur reçoit les mêmes informations.

#### **Montage**



#### **Codes**

Voici les codes que nous avons utilisés pour l’envoi et la réception de ces données :

* Librairies : nRF24L01, RF24, RF24\_config : utilisation du module nRF
* SPI : utilisation des pins de communications d’une carte programmable

Emission :

#include <Arduino.h>

#include <nRF24L01.h>

#include <RF24.h>

#include <RF24\_config.h>

#include <SPI.h>

RF24 radio(7,8); //pin CE,CSN (dépend de la carte utilisée)

const byte address[6] = "00001"; //channel d'écriture/lecture

//l'envoi des données se fait par l'intermédiaire d'un entier à 16 bits au lieu d'un int, car

//l'envoie de 3 int en tableau est trop volumineux

uint16\_t joyStickInput[3] = {0, 0, 0};

void setupEmission() {

radio.begin();

radio.openWritingPipe(address);

radio.setPALevel(RF24\_PA\_MIN);

radio.stopListening();

}

void affichageInput() {

Serial.print(" Joystick 2 X : ");

Serial.println(joyStickInput[1]);

Serial.print("JoyStick 1 Y : ");

Serial.print(joyStickInput[0]);

Serial.print(" JoyStick 2 Y : ");

Serial.println(joyStickInput[2]);

Serial.println("");

}

void loopEmission() {

//Acquisition des valeurs de chaque axe des joysticks

joyStickInput[0] = analogRead(A0);

joyStickInput[1] = analogRead(A3);

joyStickInput[2] = analogRead(A2);

//Affichage des valeurs

affichageInput();

//Envoi des valeurs

radio.write(joyStickInput, sizeof(joyStickInput));

}

void setup() {

Serial.begin(9600);

setupEmission();

}

void loop() {

loopEmission();

}

Réception :

#include <Arduino.h>

#include <SPI.h>

#include <nRF24L01.h>

#include <RF24.h>

#include <RF24\_config.h>

RF24 radio(7,8); //pin CE, CSN

const byte address[6] = "00001"; //channel d'écriture/lecture

int joyStickInput[3] = {0, 0, 0};

int joyStickLeftY;

int joyStickRightX;

int joyStickRightY;

void setupReception() {

radio.begin();

radio.openReadingPipe(0, address);

radio.setPALevel(RF24\_PA\_MIN);

radio.startListening();

}

void acquisitionInput() {

if(radio.available())

{

radio.read(joyStickInput, sizeof(joyStickInput));

for(int i = 0; i<=2; i++) //recuperation des données de l'émetteur //et mapping des valeurs pour réutilisation de celles-ci

{

joyStickInput[i] = map(joyStickInput[i], 0, 1023, -255, 255);

}

joyStickLeftY = joyStickInput[0];

joyStickRightX = joyStickInput[1];

joyStickRightY = joyStickInput[2];

}

}

void affichageInput() {

Serial.print(" Joystick 2 X : ");

Serial.println(joyStickRightX);

Serial.print("JoyStick 1 Y : ");

Serial.print(joyStickLeftY);

Serial.print(" JoyStick 2 Y : ");

Serial.println(joyStickRightY);

Serial.println("");

}

void loopReception() {

acquisitionInput();

affichageInput();

}

void setup() {

Serial.begin(9600);

setupReception();

}

void loop() {

loopReception();

}

#### **Difficultés rencontrées**

Nous avons rencontré plusieurs difficultés lors de la mise en place de cette télécommande, des difficultés qui sont de l’ordre du hardware comme du software.

**Hardware :**

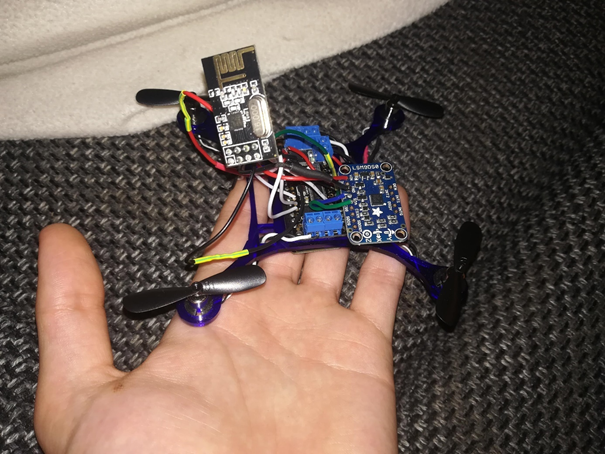
Pour commencer, nous n’avions pas les mêmes cartes dès le début. Nous avions une Adafruit pro Trinket 3V pour la réception ainsi qu’une nano Arduino (non officielle) pour l’émission. Nous nous sommes aperçus après plusieurs tests que la carte Adafruit pro trinket ne possédait pas de port série (à moins d’être connecté en FTID) ainsi que l’on ne pouvait pas upload de code sur la carte à moins d’appuyer sur le bouton reset l’instant avant d’upload. Ces problèmes n’empêchent pas la communication entre les deux modules nRF, mais ils nous ont freinés dans notre progression. Nous avons de toutes façons changer de carte après plusieurs séances pour cause d’autres problèmes liés au contrôle des moteurs

De plus, nous avons rencontré des soucis avec la carte nano non officielle. Ces soucis avaient apparemment pour origine la librairie SPI. En effet, cette librairie est prévue pour le hardware d’une carte Arduino officielle. Cependant la connectique des pins SPI de la carte nano non officielle doit être différente de celle d’une carte nano officielle, ce qui pouvait être à l’origine de nos soucis. Nous avons donc échangé la carte nano non officielle avec une carte nano officielle. Pour ce qui est de la carte Adafruit pro Trinket 3V, elle aurait éventuellement pu marcher pour la réception, mais des soucis liés au contrôle des moteurs nous ont fait changer de carte du côté réception donc nous n’avons pas pu conclure quant à l’utilisation d’une Adafruit pro trinket 3V dans une communication entre deux modules nRF.

**Software :**

Lorsque nous avons changé nos cartes et réglé les problèmes hardware, nous avons rencontré une dernière difficulté de l’ordre du software. En effet, nous réussissions à envoyer l’information d’un axe d’un joystick, mais envoyer un tableau de 3 ou 4 entiers n’étaient pas possibles. Nous nous sommes alors rendu compte qu’il s’agissait d’un problème de taille de paquets. En effet, l’envoie d’un tableau de 3 entiers de type « int » étant trop volumineux, nous avons dû envoyer nos valeurs sous forme d’un tableau d’entiers de type « uint16\_t ».

### **Le Drone**

Voici ci-dessous une photo du mini-drone :

Aucun composant n’est fixé à l’armature et le câblage est encore précaire, cela s’explique par le fait que nous attendions de réussir à faire marcher tous les éléments du drone pour passer à cette étape de finalisation, étape que nous n’avons jamais atteinte.

Nous allons d’abord décomposer le drone composant par composant afin de voir leurs montages simples ainsi que les codes qui leurs sont associés.

#### **Le gyroscope**

##### **Fonctionnement**

Le gyroscope est un capteur qui récupère l’accélération du drone suivant les axes X Y et Z (grâce à un accéléromètre) ainsi que la vitesse angulaire (grâce à un gyromètre) suivant ces mêmes axes (il permet également de reprendre les mesures faites par un magnétomètre, mais nous n’en n’avons pas besoin ici).

Dans notre situation, nous voudrions récupérer des angles selon les axes X et Y, nous verrons plus tard pourquoi. Nous avons deux manières de récupérer l’angle que fait le gyroscope par rapport à l’horizontal : grâce à la vitesse angulaire de celui-ci et grâce à l’accélération de celui-ci.

Nous avons vu sur internet que l’utilisation courante est un mélange des deux dans lequel on prend environ 95% des valeurs du gyromètre et 5% des valeurs de l’accéléromètre. Cependant nous avons utilisés un exemple n’utilisant que les valeurs de l’accéléromètre et n’avons pas réussi à appliquer les méthodes trouvées sur internet mélangeant accéléromètre et gyromètre. Les valeurs d’angle trouvées étant cohérentes et stables à long terme, nous n’avons rien changé au code.

##### **Montage**

##### **Code (code basique repris d’un exemple de la librairie)**

Libraires : Wire, SPI, SFE\_LSM9DS0.

Les librairies Wire et SPI sont utilisées dans la librairie SFE\_LSM9DS0, qui permet d’utiliser les fonctions pour calculer les angles qui nous intéressent.

Voici le code utilisé pour récupérer et afficher les angles que le gyroscope fait en X et en Y par rapport à l’horizontal :

#include <Arduino.h>

#include <SPI.h> // Included for SFE\_LSM9DS0 library

#include <Wire.h>

#include <SFE\_LSM9DS0.h>

// SDO\_XM and SDO\_G are both grounded, so our addresses are:

#define LSM9DS0\_XM 0x1D // Would be 0x1E if SDO\_XM is LOW

#define LSM9DS0\_G 0x6B // Would be 0x6A if SDO\_G is LOW

// Create an instance of the LSM9DS0 library called `gyroData` the

// parameters for this constructor are:

// [SPI or I2C Mode declaration],[gyro I2C address],[xm I2C add.]

LSM9DS0 gyroData(MODE\_I2C, LSM9DS0\_G, LSM9DS0\_XM);

void printOrientation(float x, float y, float z);

void setupGyro();

void loopGyro();

void setupGyro()

{

gyroData.begin();

}

void loopGyro(){

gyroData.readAccel(); //Acquisition des valeurs de l'accéléromètre

printOrientation(gyroData.calcAccel(gyroData.ax),gyroData.calcAccel(gyroData.ay), gyroData.calcAccel(gyroData.az));

Serial.println();

}

float pitch, roll;

void printOrientation(float x, float y, float z){

//Calcul des angles depuis les valeurs d'accéléromètre

pitch = atan2(x, sqrt(y \* y) + (z \* z));

roll = atan2(y, sqrt(x \* x) + (z \* z));

pitch \*= 180.0 / PI;

roll \*= 180.0 / PI;

//Affichage des angles

Serial.print("Pitch, Roll: ");

Serial.print(pitch, 2);

Serial.print(", ");

Serial.println(roll, 2);

}

void setup(){

Serial.begin(9600);

setupGyro();

}

void loop(){

loopGyro();

}

##### **Difficultés**

Pour cette partie là, nous n’avons pas rencontré de difficultés particulières.

#### **Les moteurs**

##### **Montage**

##### **Code**

Librairies : Adafruit\_MotorShield, utility/Adafruit\_MS\_PWMServoDriver

#include <Arduino.h>

#include <Adafruit\_MotorShield.h>

#include "utility/Adafruit\_MS\_PWMServoDriver.h"

Adafruit\_MotorShield AFMS = Adafruit\_MotorShield();

Adafruit\_DCMotor \*myMotor1 = AFMS.getMotor(1);

Adafruit\_DCMotor \*myMotor2 = AFMS.getMotor(2);

Adafruit\_DCMotor \*myMotor3 = AFMS.getMotor(3);

Adafruit\_DCMotor \*myMotor4 = AFMS.getMotor(4);

void setupMotors() {

AFMS.begin();

//sens de rotation des moteurs

myMotor1->run(FORWARD);

myMotor2->run(FORWARD);

myMotor3->run(FORWARD);

myMotor4->run(FORWARD);

}

void setup() {

setupMotors();

}

void loop() {

//on attribue une vitesse (entre 0 et 255) pour chaque moteur, peut être fait dans le setup

//si l'on veut que la vitesse reste la même

myMotor1->setSpeed(100);

myMotor2->setSpeed(200);

myMotor3->setSpeed(150);

myMotor4->setSpeed(10);

}

##### **Difficultés**

* Tout d’abord nous avons changé de carte Arduino à plusieurs reprises avant de finir sur une Feather ESP32 accompagnée d’un Shield pour moteur DC et stepper. Car il était impossible de faire tourner les moteurs avec simplement des pin pwm.
* Une fois la bonne carte et le bon Shield trouvés nous avons dû connecter le 3v3 du la Feather sur le Vin du Shield et de même avec les Ground afin d'alimenter le Shield, ce qui n’était pas évident pour nous. Nous avons donc mis un certain temps avant de le comprendre.

#### **Mise en commun des composants**

##### **Fonctionnement du drone dans sa globalité**

De manière générale, le drone fonctionne de cette manière : la télécommande sert à commander l’inclinaison du drone en avant et sur le côté (Y et X) ainsi que la puissance globale des moteurs. L’inclinaison en avant est appelé le **« PITCH »** tandis que celle sur le côté est appelé le **« ROLL »**. Le gyroscope lui va permettre au drone de se stabiliser. Ici, l’asservissement du drone est dit sur l’assiette. Ce qui veut dire que le drone va constamment chercher à revenir à une position horizontale.

Plus précisément, nous allons utiliser le joystick gauche de la télécommande pour gérer la puissance des quatre moteurs et le joystick droit pour le pitch (axe y du joystick) et le roll (axe x du joystick), autrement dit le joystick droit va gérer l’inclinaison du drone. Ce joystick droit va permettre de créer une différence (ou un offset) entre les différentes puissances des moteurs afin d’incliner le drone.

Pendant ce temps, le gyroscope va calculer l’angle que fait le drone en Y et en X afin de régler la puissance des moteurs en fonction de ces angles. Par exemple : Si le drone penche vers l’avant (donc pitch positif), alors nous allons ajouter un offset à la puissance des moteurs avants et soustraire cet offset à la puissance des moteurs arrières. En effet, les moteurs avant tourneront plus vite et les moteurs arrières tourneront moins vite. Cet offset étant égal à l’angle que fait le drone vers l’avant, multiplié par un coefficient, la stabilisation se fera naturellement et progressivement. Le coefficient que l’on applique à l’angle sert à calibrer la correction que l’on souhaite appliquer, si l’on souhaite qu’elle soit plus ou moins rapide.

##### **Montage**

##### **Difficultés**

Nous avons réussi à faire fonctionner les composants un à un ainsi qu’à comprendre le fonctionnement du drone et des algorithmes que l’on devait appliquer, cependant nous avons rencontrés quelques problèmes liés au hardware que nous n’avons su résoudre :

* Pour commencer, nous avons eu des problèmes liés à (nous pensons) **l’alimentation**. En effet, nous avons quelques fois réussi à faire fonctionner nos quatre moteurs à une certaine puissance, cependant ces situations étaient très rares et instables.

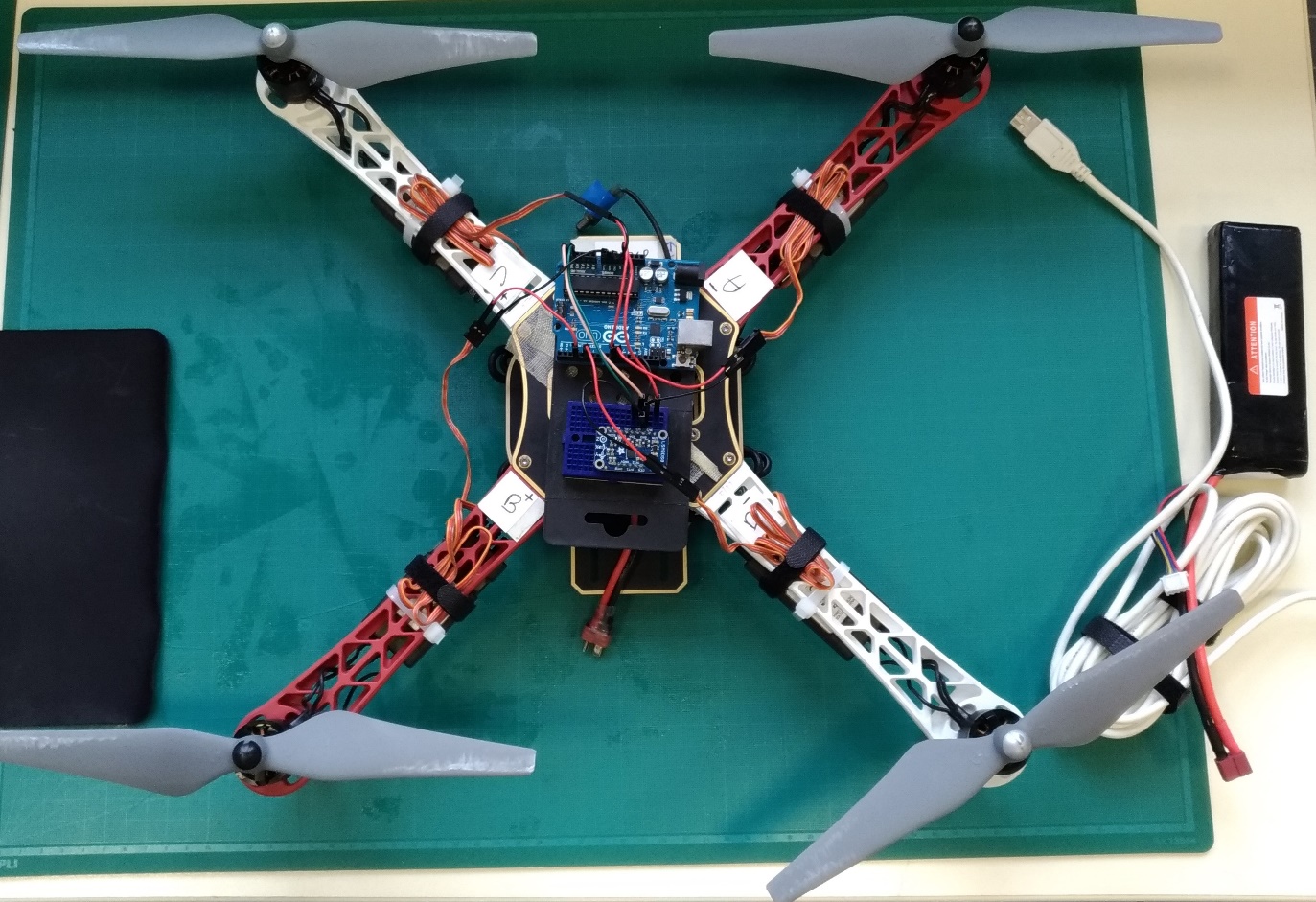
Un des quatre moteurs finissait par s’éteindre, ou les autres moteurs par faiblir et nous n’avons pas réussi à trouver la réelle source du problème. Nous n’avons donc jamais réussi à faire tourner les 4 moteurs à puissance maximale de manière stable.

* De plus, nous avons rencontré un problème de masse. il a été impossible de faire tourner les 4 moteur à une vitesse suffisante pour faire décoller le drone (1er prototype). En effet la batterie ne délivrait pas un voltage suffisant pour les moteurs qui étaient branchés en série, il en résultait donc que les moteurs 1, 2 et 3 tournait alors que le 4eme ne se lançait pas lorsque les moteurs était “normalement” censé être à pleine puissance.
* Ensuite, une fois le nRF soudé sur la carte, nous n’avons plus réussi à faire fonctionner les moteurs.

Un grand nombre de ces problèmes ont peut-être une solution simple, cependant la multitude d’entre eux et le manque de temps ont fait que nous n’avons pas réussi à les régler

## **III. Drone de M. Romet**

Avant d’introduire cette partie, nous tenons à remercier M. Romet pour nous avoir prêté son drone afin que l’on puisse essayer de le faire voler nous-même, sans avoir de contraintes d’hardware.

Nous avons donc pu travailler sur le drone suivant :

Le mini-drone n’étant pas fonctionnel et potentiellement impossible à faire fonctionner (pour les raisons évoquées ultérieurement), nous avons commencé à travailler sur le drone de M. Romet.

Le principe de fonctionnement reste le même, et les composants également. Les seules différences viennent de la carte, qui est un Arduino Uno, ainsi que des moteurs qui sont des servomoteurs brushless. Ainsi, les explications ultérieures et les codes ultérieures sont toujours valables et ont été utilisé pour ce drone (sauf pour les moteurs).

### **Fonctionnement des nouveaux moteurs et différence de fonctionnement avec le mini-drone**

Ce nouveau drone est entièrement alimenté par une batterie externe de 12V. Cette batterie va alimenter les 4 moteurs et servira également à alimenter la carte. La carte, elle, n’alimente alors plus les moteurs. Ceux-ci sont tout de même branchés sur des sorties PWM de la carte, mais il ne s’agit pas là d’une alimentation. Il s’agit d’un moyen pour la carte de leur envoyer des informations. En effet, cette alimentation sera retranscrite sous forme d’information par les moteurs, ce qui permet alors de les contrôler. Le montage est donc similaire à celui vu précédemment, avec une alimentation et un ground.

### **Code permettant de faire tourner les 4 moteurs en même temps**

Librairies : Servo.h, ESC.h : permettent le contrôle de nos servomoteurs

#include <Arduino.h>

#include <Servo.h>

#include "ESC.h"

Servo esc[4];

void setupMotors() {//Fonction qui permet l'initialisation des moteurs

//Attribution des pins correspondants

esc[0].attach(10);

esc[1].attach(9);

esc[2].attach(5);

esc[3].attach(3);

//Initialisation

esc[0].writeMicroseconds(1000);

esc[1].writeMicroseconds(1000);

esc[2].writeMicroseconds(1000);

esc[3].writeMicroseconds(1000);

delay(2000);//Délai nécessaire à l'initialisation

motorsSpeed(); //Lancement des moteurs

}

void motorsSpeed() {

for(int i = 0; i < 4; i++)

{

esc[i].writeMicroseconds(1500);

}

}

void setup() {

setupMotors();

}

void loop() {

}

Nous n’avons pas rencontré de difficulté particulière quant au fonctionnement des moteurs.

### **Etude du code du nouveau drone**

La partie hardware étant déjà achevée, nous n’avions plus qu’à écrire un code fonctionnel afin de faire voler le drone. Pour rappel, voici l’objectif :

* Contrôler la puissance des moteurs avec le joystick gauche
* Faire avancer/reculer/tourner le drone avec le joystick droit
* Garder le drone stable

Nous n’allons pas détailler l’entièreté du code ici, cependant nous allons expliquer ce que nous jugeons être le plus important. (le code commenté est disponible dans le dossier que nous avons envoyé).

Setup :

Pour commencer, nous initialisons tout ce que nous devons initialiser (le nrf, le gyro et les moteurs).

void setup() {

Serial.begin(9600);

setupGyro();

setupMotors();

setupReception();

}

void loop() {

acquisitionInput();//Récupération des valeurs des joysticks envoyées par la télécommande

loopGyro();

delay(50);//délai permettant une évolution plus lente et contrôlable de la puissance des moteurs avec le joystick gauche

gaz(); //calcul de la nouvelle puissance commune aux 4 moteurs

correction(); //calcul de l'offset que l'on doit imposer à chaque moteur en fonction de son degré de stabilisation et du joystick droit

}

Loop :

Nous allons d’abord récupérer les valeurs des joysticks depuis la télécommande (grâce au nRF) ainsi que l’angle que fait le drone avec l’horizontal en X et en Y (roll et pitch). Ensuite, nous allons nous servir des valeurs récupérées du joystick gauche afin de déterminer la puissance commune que doivent avoir les 4 moteurs lorsqu’ils sont en position “assiette” (à l’horizontal).

Enfin, nous allons déterminer l’offset que doit avoir chaque moteur par rapport à la puissance commune en fonction de l’inclinaison du drone et de celui du joystick droit.

Ce qui donne :

void correction() {

//Le paramètre correspond à l'offset que l'on va appliquer à la puissance de chaque moteur par rapport

//à la puissance commune déterminée par le joystick gauche (fonction gaz)

offsetMoteur1(pitch + roll - (joyStickRightX + joyStickRightY)/coefficientJoystickDroit);

offsetMoteur2(pitch - roll + (joyStickRightX - joyStickRightY)/coefficientJoystickDroit);

offsetMoteur3(-pitch + roll - (joyStickRightX - joyStickRightY)/coefficientJoystickDroit);

offsetMoteur4(-pitch - roll + (joyStickRightX + joyStickRightY)/coefficientJoystickDroit);

}

Encore une fois, n’est présenté ici que le fonctionnement général du programme, certaines petites fonctions comme les “offsetMoteurX” ne sont pas montrées ici du fait de leur principe très simple.

Le code présenté ici marche théoriquement pour l’utilisation d’un drone, il est exploitable. Cependant nous n’avons tout de même pas réussi à faire voler le drone car il fallait calibrer nos différentes valeurs (notamment les coefficients multipliés aux angles pour la stabilisation du drone) et nous avons manqué de temps.

### **Difficultés rencontrées**

Comme nous venons de l’évoquer, du temps en plus nous auraient peut-être permis de faire voler le drone de manière plus stable après quelques ajustements de nos valeurs et quelques modifications.

De plus nous avons rencontrées de légères difficultés au niveau de la programmation de manière générale. En effet, le fait de ne pas pouvoir forcément tester son code instantanément implique de devoir effectuer des sauvegardes de manières très régulières et à être très organisé dans son travail.

Enfin, il est clair qu’il devient vite facile de se perdre dans son programme (ceci est encore plus vrai lorsqu’on travaille avec d’autres personnes), il est donc très important de partitionner son travail et de structurer son programme en beaucoup de fonctions très claires et explicites.

## **IV. Conclusion**

Ce projet nous a permis de comprendre les principes de la programmation embarquée et nous a permis de mettre en place un environnement de développement pour Arduino. Par ailleurs, si nous avions eu un délai plus important, nous aurions pu perfectionner le drone et peut être même designer le boitier qui sert de télécommande.

Nous avons aussi pu découvrir de nouveau logiciels, comme Fritzing et Fusion, très utilisés dans le monde du travail mais inconnus pour nous jusqu’à ce jour.

## **V. Annexes**

**Annexe 1: Outils et méthodologies**

**A. Méthodes de travail :**

Pour mener à bien notre projet, nous nous sommes réparti les tâches. Aymeric et Antoine ont travaillé sur la télécommande, avec la gestion des joysticks, la lecture de leurs valeurs et leur transmission par radio, entre deux NRF.

Victor et Catherine ont, quant à eux, travaillé sur le drone, avec la gestion des moteurs, l’utilisation du gyroscope et le pilotage des moteurs en fonction des valeurs données par le gyroscope.

Nous nous sommes vus deux fois par semaines, en cours, pour créer notre projet et échanger sur nos avancées. Cependant, nous n’avions pas assez de temps pour réaliser entièrement notre projet, nous sommes donc allés au CrunchLab le jeudi après-midi. Nous avons aussi avancé notre projet sur nos temps libre. En plus de ces entrevues, la personne qui récupérait la boite avec les composants, d’une séance à l’autre, pouvait avancer le travail chez elle.

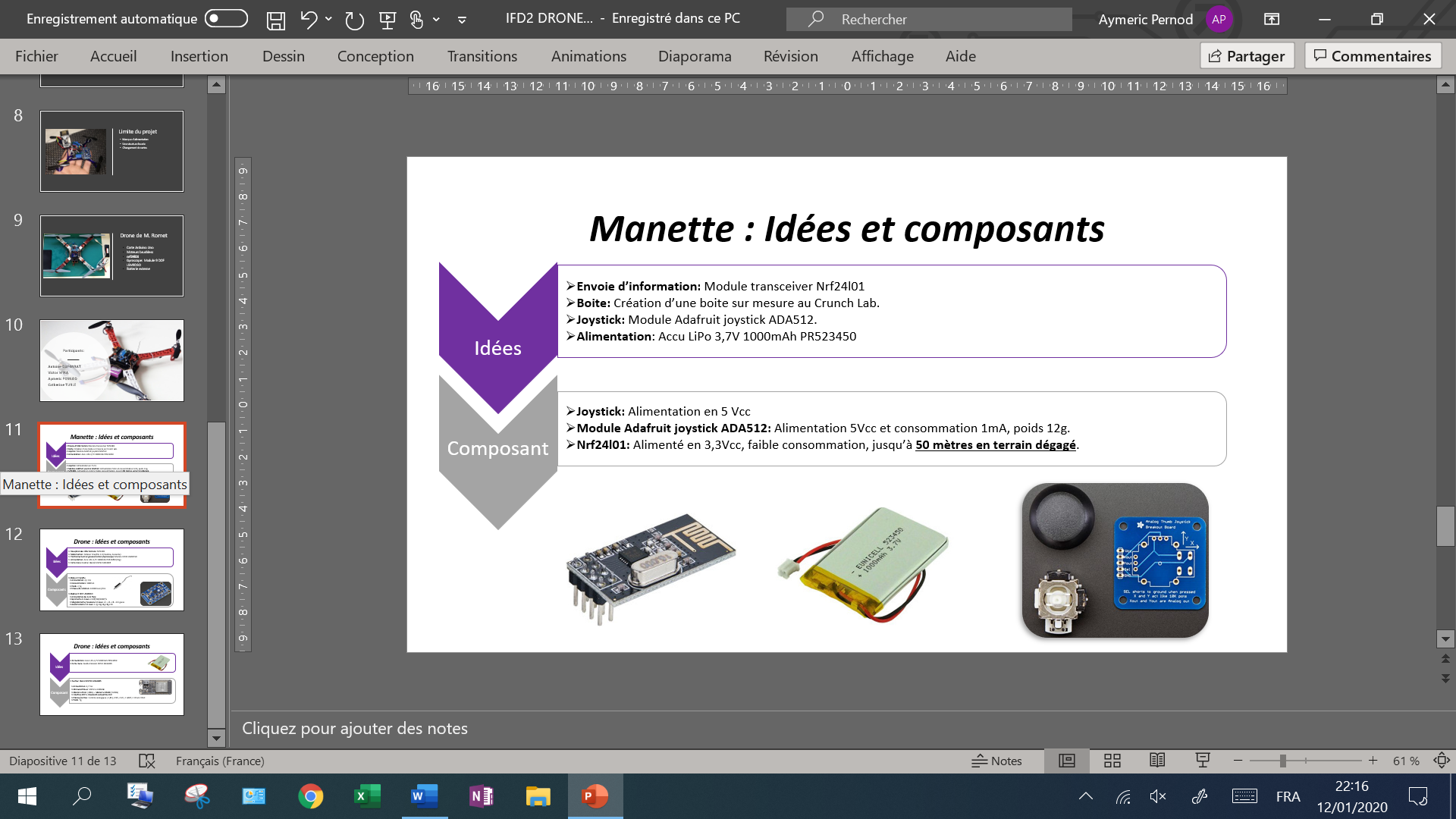
**B. Logiciels utilisés:**

Nous avons aussi utilisé les logiciels suivants pour nous aider dans la conception du drone et de sa télécommande :

**-Fritzing**, pour créer et représenter les circuits électroniques que nous allions monter et connaitre la place dont nous allions avoir besoin pour les mettre en place,

**-Fusion**, pour créer en 3D la télécommande et l’armature du drone. Nous avons ensuite pu imprimer en 3D ou découper au laser les parties nécessaires au montage du drone et de sa télécommande,

-**Google Drive** au début puis **GitHub**, pour partager entre nous les recherches que nous avions faites et les codes et schéma que nous avions créés.

**Annexes 2 : Composants**

