DRONE STATIONNAIRE

PROJET ELECTRONIQUE PEIP 2 LEHYAN FLOURIOT G3

Rapport de séance 9 :

Nous allons découper cette séance en plusieurs parties distinctes car j'ai réalisé beaucoup de chose cette semaine.

Partie 1 : Le problème du PWM

Comme j'avais expliqué dans le rapport précédent, il me manque des sorties PWM. Je dois donc arriver à créer moi-même le PWM à l'aide de sorties D2-13 digitale.

Pour cela le principe est extrêmement simple :

Chaque 4ms j'envoie pendant un certains tems l'état **HAUT** puis je remets à l'état **BAS**.

Cela reproduit exactement le comportement d'un PWM.

Je vais utiliser cette technique pour les sorties liées au moteurs / ESCs. Ceux-ci prennent des valeurs allant de 1000 microsecondes à 2000 microsecondes.

Je dois donc générer un signal dans cette fourchette de temps.

Le code suivant (sur GitHub aussi) va s'occuper de générer ce fameux signal :

```
#include <Wire.h>

#define FREQ        250   // Sampling frequency
// ---------------------- Variables for servo signal generation -------------
unsigned int  period; // Sampling period
unsigned long loop_timer;
unsigned long now, difference;
unsigned int pulse = 1000;

void setup() {
  // put your setup code here, to run once:
  Serial.begin(57600);
  // Start I2C communication
    Wire.begin();
    TWBR = 24; // 400kHz I2C clock (200kHz if CPU is 8MHz)
  // Set pins #4 #5 #6 #7 as outputs
    DDRD |= B11110000;



    period = (1000000/FREQ) ; // Sampling period in µs

    // Initialize loop timer 2
    loop_timer = micros();
```

```
    //ESCs set up
    while(millis()<7000){
      applyMotorSpeed();
      Serial.println("ESC");
    }
}

void loop() {
  // put your main code here, to run repeatedly:
  pulse=1050;
  Serial.println(pulse);
  applyMotorSpeed();
}

void applyMotorSpeed() {
    // Fs = 250Hz : on envoie les impulsions toutes les 4000µs
    while ((now = micros()) - loop timer < 4000);

    // Update loop timer
    loop timer = now;

    // On active les broches #4 #5 #6 #7 (état HIGH)
    PORTD |= B11110000;
   // Serial.println("HIGH");

    // On boucle tant que toutes les sorties ne sont pas retournées à l'état LOW
    while (PORTD >= 16) {
        now = micros();
        difference = now - loop timer;


        if (difference >= pulse) PORTD &= B01111111; // Passe la broche #7 à LOW
        if (difference >= pulse) {PORTD &= B11101111;
          //Serial.println("LOW");
        } // Passe la broche #4 à LOW
        if (difference >= pulse) PORTD &= B11011111; // Passe la broche #5 à LOW
        if (difference >= pulse) PORTD &= B10111111; // Passe la broche #6 à LOW

    }
}
```

Le **setup()** va mettre le pulse initial pour initier les ESCs (cette partie est propre à mes ESCs, sans cette étape, les moteurs ne s'allument pas car les ESCs doivent par « sécurité » recevoir un signal minimal pendant un certain temps avant de lancer les moteurs).

La fonction applyMotorSpeed() génère justement le PWM pour chaque sortie 4,5,6,7.

Les **Serial.println()** était simplement un moyen pour moi de tester le code, ils servent à rien désormais.

SPOIL, finalement je ne pense pas avoir besoin de cette fonction car j'ai pu libérer une sortie PWM (le TX du Bluetooth) donc je ne m'en servirais probablement pas, sauf si le drone est trop lent à se stabiliser. En effet, cette technique est beaucoup plus rapide que la génération d'un signal classique avec la bibliothèque **<Servo.h>.**

Partie 2 : La protection des hélices

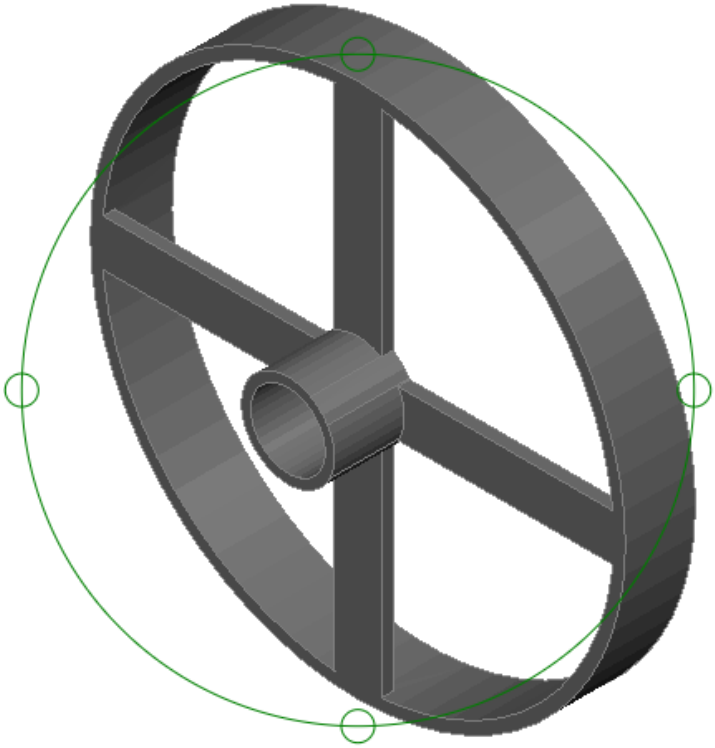
Comme montré également dans le dernier rapport, j'ai imprimé des protections d'hélices en plastique. MAIS, ils ne sont pas très adaptés.

En effet elles présentent plusieurs problèmes :

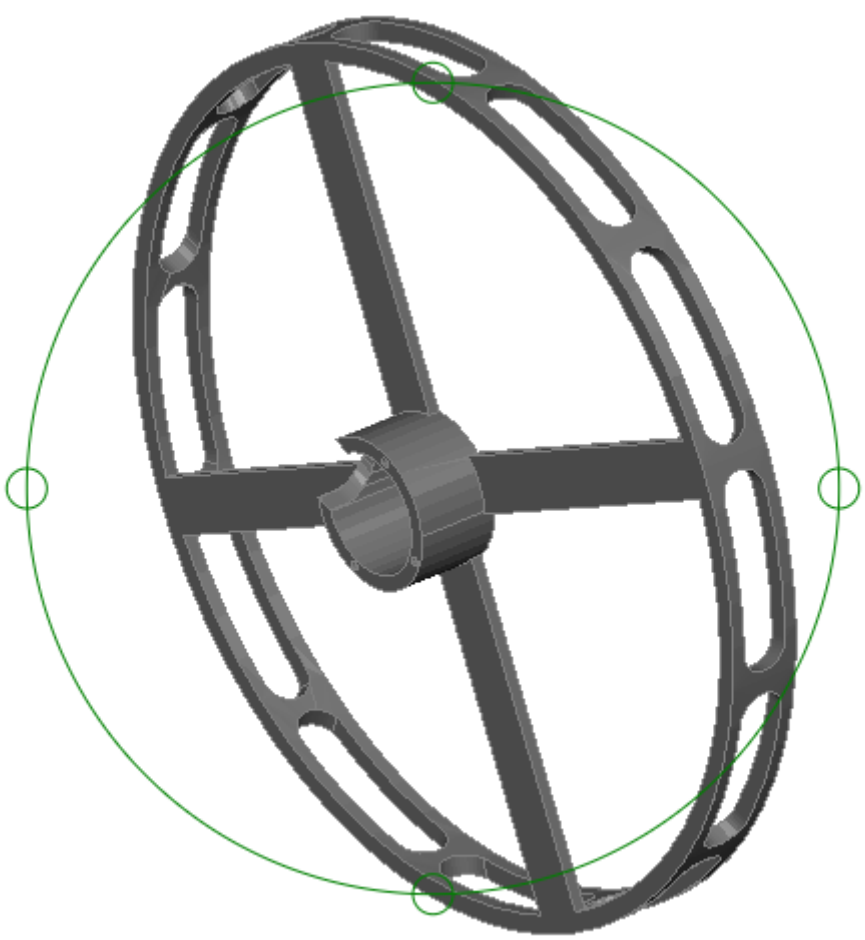
- Elles sont trop lourdes (le drone a largement assez de puissance, mais rajouter du poids inutilement est contre-productif)
- Elles sont trop hautes et trop « justes » (les hélices frottent s'il y a de petite vibrations)
- Il manque un petit trou pour faire passer les fils du moteur
- Initialement je voulais les coller, mais je vais plutôt les visser, donc je dois faire des trous

Tous ces problèmes m'obligent donc à refaire un model sur SolidWorks. Celui-ci sera plus « vide » et percé / troué.

Ancienne pièce :



Nouvelle piece :



Cette nouvelle pièce devrait résoudre tous ces problèmes.

Partie 3 : Le Bluetooth et la visualisation 3D du drone (sur GitHub)

Oui, oui. On peut voir le drone sur une fenêtre de l'ordinateur sans même qu'il soit branché (#LeBluetooth). Et le plus fou dans tout ça, c'est que le drone va actualiser son inclinaison **en DIRECT LIVE !**

J'ai personnalisé le tout premier code sur Processing que j'avais publié, de sorte qu'il récupère les informations envoyées par Bluetooth par le drone. Celui-ci va ensuite mettre à jour l'orientation d'une forme (pour l'instant un simple pavé) afin de reproduire exactement les mouvements du drone. Démonstration lors de l'oral ☺.

Le premier problème qui est apparu est le taux de rafraîchissement du programme. Pour résoudre cela, je dois simplement augmenter les BAUDS du module Bluetooth.

A l'aide du DATASHEET, je mets la main sur la commande à envoyer au model AT+BAUD7 (cela va passer le taux de 9600 à 57600). Le programme est maintenant super fluide, hâte de montrer cela, je vais juste essayer de donner un semblant de forme de drone au solide du programme (un pavé c'est pas très beau).

Bluetooth FAIT !

Partie 4 : Le Circuit imprimé (tout sur GitHub)

Cette étape était sans aucun doute la plus longue de la semaine, mais clairement la plus indispensable. Les câbles qui se baladent sur la planche de test, qui se débranchent en vol, qui s'emmêlent… ça comment à me rendre fou. Donc enlevons tous les câbles et créons notre propre circuit imprimé !

Je ne m'étais jamais vraiment intéressé aux circuit imprimés, mais c'est finalement **très très** pratique !

J'ai donc pour but de ne laisser aucun câble, sortons le crayon et dessinons un schéma de câblage.

Téléchargeons ensuite le premier logiciel gratuit de création de circuit (AUTODESK Eagle) et lisons comment utiliser ce logiciel.

Après une petite heure de formation, commençons à retranscrire le dessin sur le logiciel.

Il me faut d'abord recréer la carte Arduino car ce circuit viendra s'emboiter par-dessus, puis disposer chaque élément et les reliés par des câbles sans oublier chaque GND et +5v.

Une autre heure (et 300 000 vérifications de branchements) plus tard, voilà le résultat :

Tous les composants sans liaisons

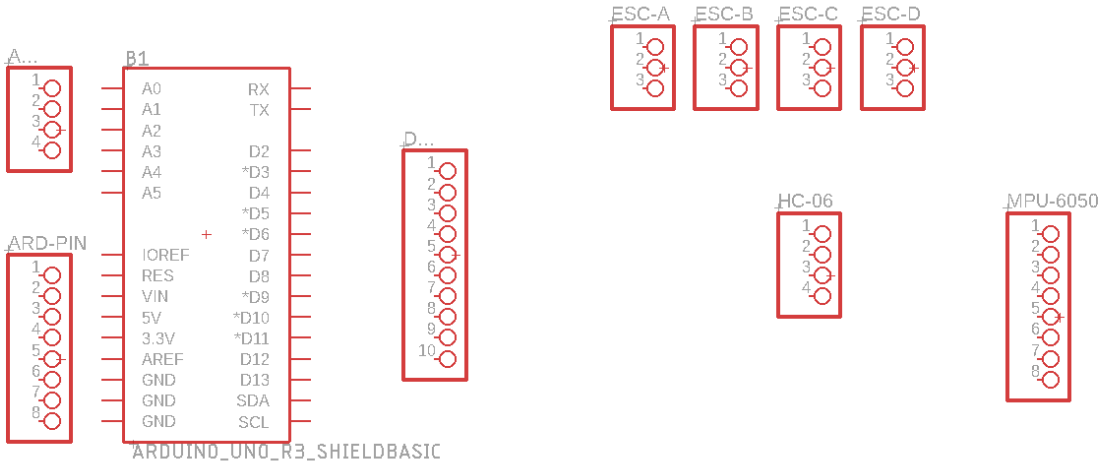
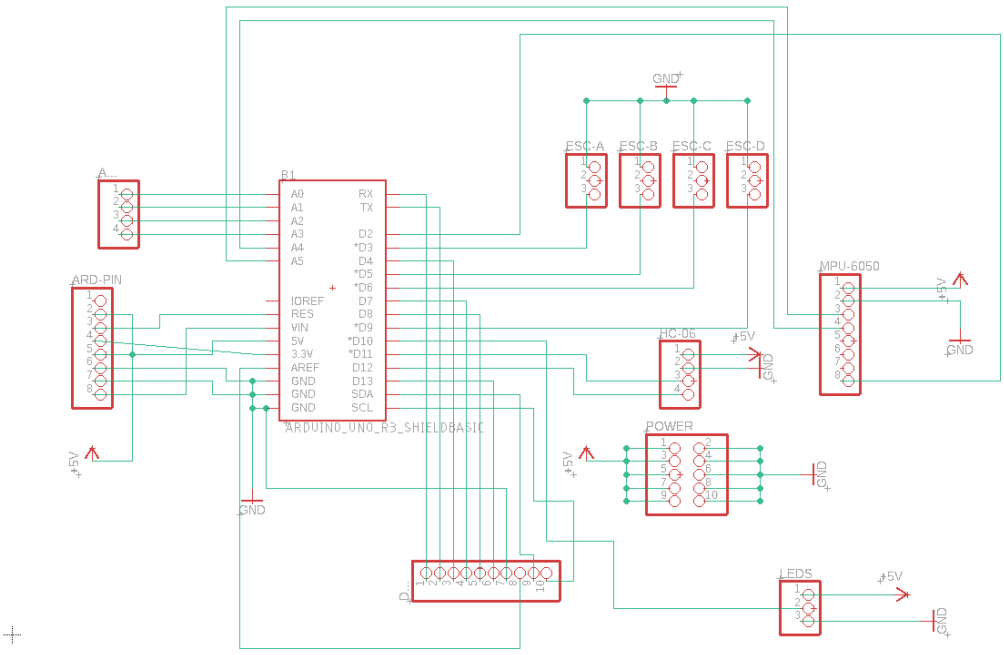
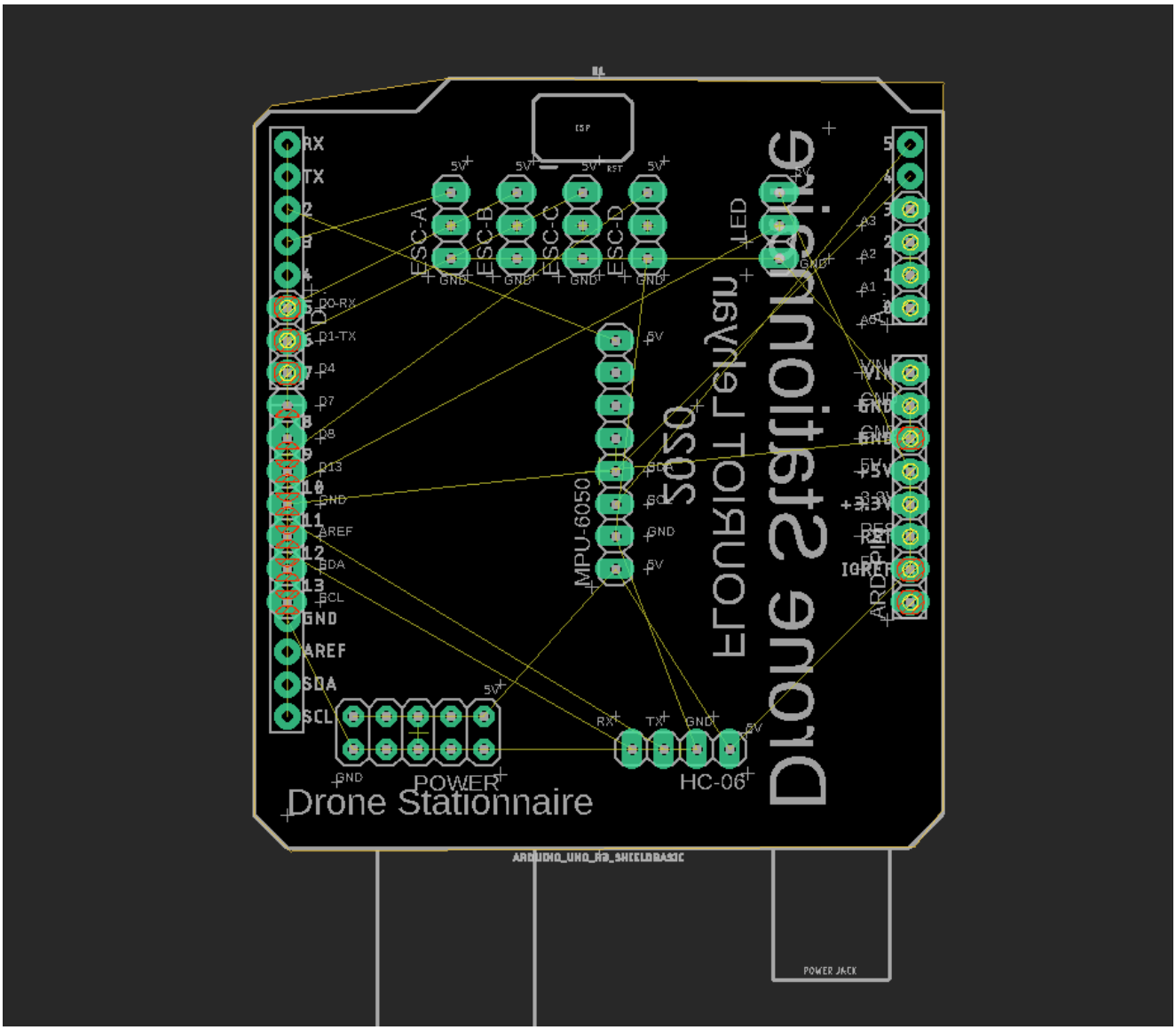


Schéma final

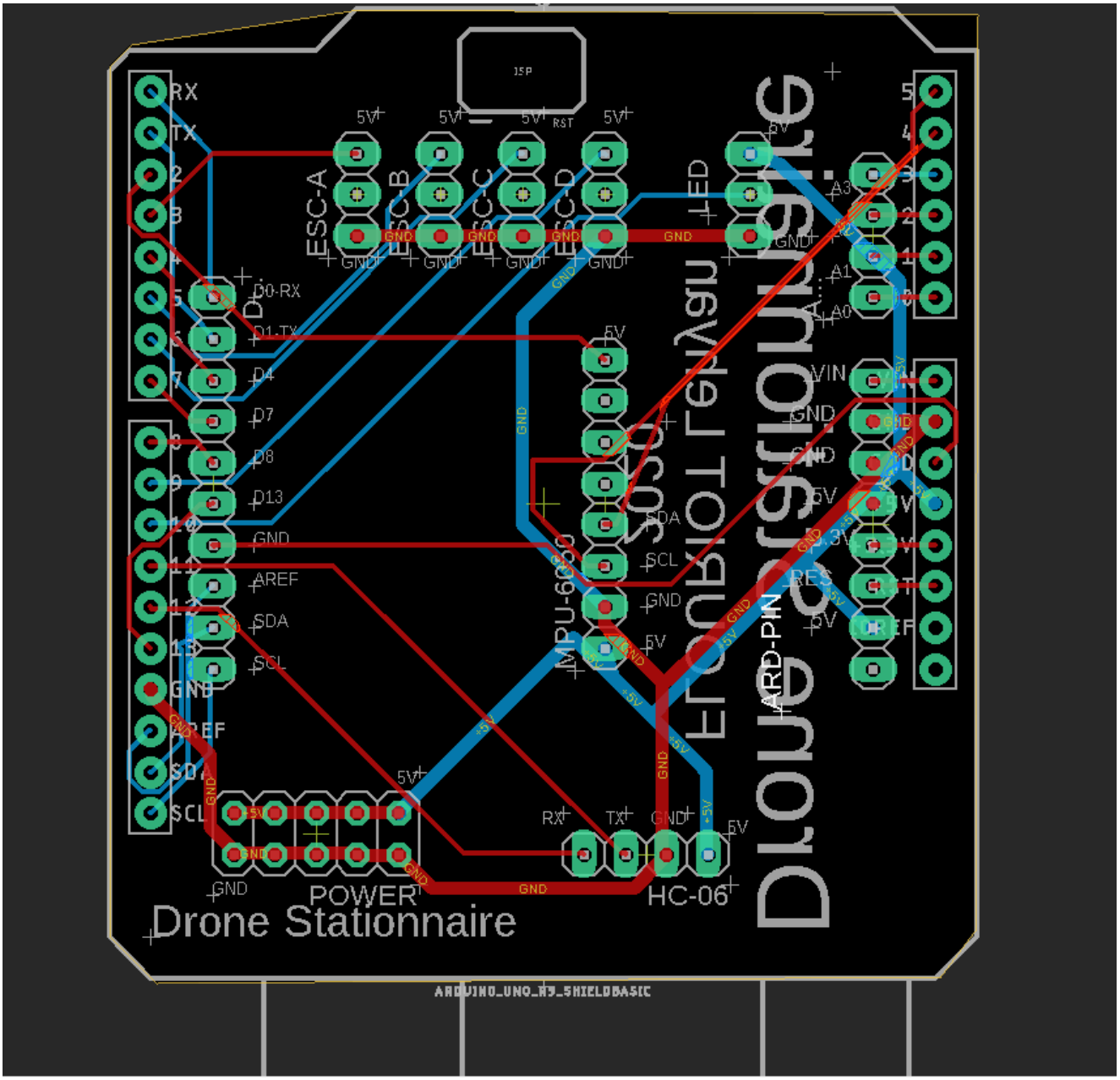


Le schéma étant terminé, il faut le transférer sur un fichier de Board.

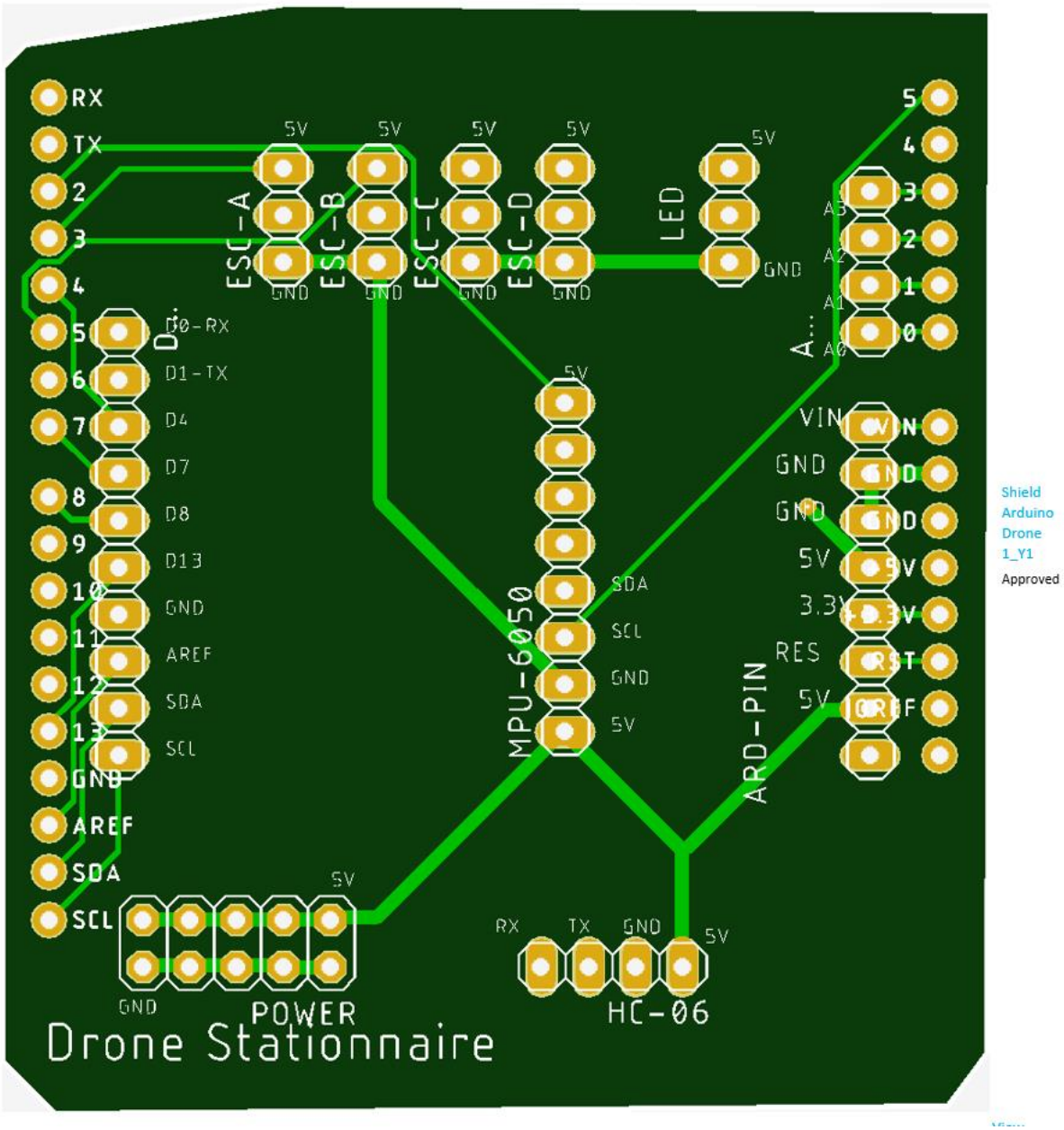
Fichier de la Board sans pistes (équivalent des câbles)



Fichier de Board avec pistes finales



Circuit imprimé final



Et voilà, le circuit est imprimé et en cour d'acheminement, il devrait arriver la semaine prochaine et évitera tous les câbles de gâcher on projet. De plus chaque composant sera directement emboité sur les PINs que je vais par la suite souder sur la carte.

Voilà pour cette séance, la prochaine sera forcément consacrée à la stabilisation, mais j'aimerai aussi faire une petite coque pour le haut du drone afin de ne pas avoir les composants à l'air libre.