Par Elies HATOUM :
Mail : elieshatoum@hotmail.com

Rapport du FX09 : Projet DreamAir

Cspace 2022-2023

Révision 1.0

Août 2023

SOMMAIRE

0. ACRONYMES, ABREVIATIONS

Elec : Electronique

IPSA : Institut polytechnique des sciences avancés

IU : User Interface

1. Remercîment

Je tiens à exprimer ma profonde gratitude envers toutes les personnes qui ont apporté leur précieuse aide au projet DreamAir. Évidemment, un grand merci à l'association AeroIpsa pour avoir financé ce fabuleux projet pendant 2 ans.

Je tiens à remercier Laure-Emilie qui m'a motivé à créer le projet en 2021 et qui, pour sa première couture, a fabriqué un parachute qui a emmené une fusée à plus de 1000 mètres d'altitude.

Merci à tous les membres de l'association qui ne font pas partie du projet, mais qui ont apporté une aide significative, spécialement Kevin, sans qui aucune des pièces 3D de la fusée ne tiendrait debout, et Laure-Amelie qui a cousu le parachute du CanSat GreenAir.

Merci aux deux présidents de l'association qui ont soutenu le projet pendant ces deux années, Alizée, sans qui le corps de la fusée ne serait encore qu'un moule de PLA, et Vincent, qui a accompli le travail de 1000 hommes sur tous les blocs, et qui a encouragé le projet jusqu'à son décollage.

Évidemment, merci à tous les membres du projet qui l'ont rejoint dès sa création et qui ont donné de leur précieux temps. Merci à Ramdane qui a accompagné la construction et le largage du CanSat, merci à Angel qui a réalisé un bloc élec fonctionnel et esthétique, merci à Nasca pour son implication au Cspace et l'installation de la télémétrie en un temps record, et enfin, le plus grand merci à Lucas Pichon qui m'a accompagné du début de DreamAir jusqu'au vol de la fusée, qui a travaillé plus que n’importe qui sur ce projet, et a poursuivi son développement tout en agissant en tant que chef de projet durant mon absence.

2. INTRODUCTION

Le projet de la fusée DreamAir avait initialement pour but d'étudier les chocs et secousses subis par un CanSat largué à haute altitude. C'est avec cet esprit que le projet a été créé. DreamAir devait contenir un système de roulis utilisant les ailerons, d'où une emplanture égale à la somme du saumon et de la flèche.

Le but était de réduire au minimum les forces de roulis exercées lors de l'éjection du CanSat. Suite à des contraintes de temps, le contrôle de roulis a été abandonné afin d'évoluer dans l'expérience de la fusée en comparant l'IMU contenu dans la fusée avec celui du CanSat.

Le CanSat est nommé GreenAir. Il a été initialement lancé par drone en 2022, devenant ainsi le premier CanSat lancé par drone au Cspace. Au Cspace 2023, DreamAir et GreenAir ont été lancés le Mercredi 19 juillet 2023.

3. Description Mécanique

3.1 StrabTraj

STABILITO

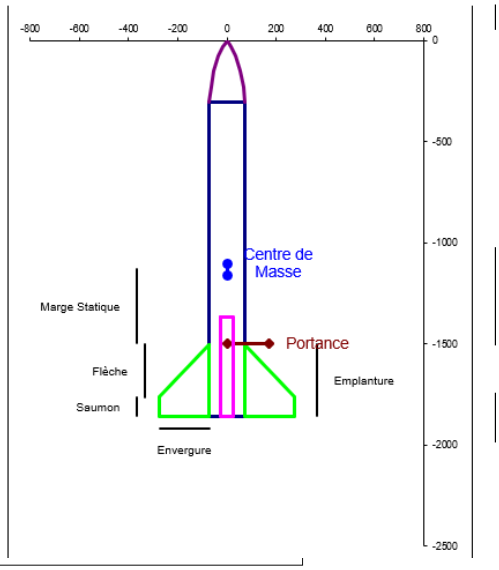
Stabilité de fusée à ailerons

Remplir les cases jaunes

	Fusée	
Nom	DreamAir	
Club	AeroIPSA	
Type	Fusée expérimentale.	
Masse	7860 g	sans propu
Centre de Masse	1060 mm	sans propu
Longueur totale	1858 mm	

	Propulseur
Type	Barasinga (Pro54-5G C)
Position du bas	1858 mm

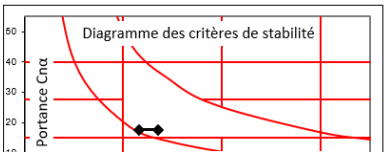
	Coiffe
Forme	Ogivale (pointue)
Hauteur	305 mm
Diamètre	146 mm



	Ailerons
	Mono-empennage
Emplanture 'm'	355 mm
Saumon 'n'	96 mm
Flèche 'p'	259 mm
Envergure 'E'	201 mm
Epaisseur 'ep'	5 mm
Nombre	4
Position du bas	1858 mm

18/07/2023	Min	Résultats		Max
Finesse	10	12.7		35
Portance	15	17.6	17.6	40
MargeStat.	2 D	2.32 D	2.70 D	6 D
Couple	40	40.8	47.7	100
XCp		1497 mm	1497 mm	
MS /L		18% L	21% L	

STABLE



TRAJECTO

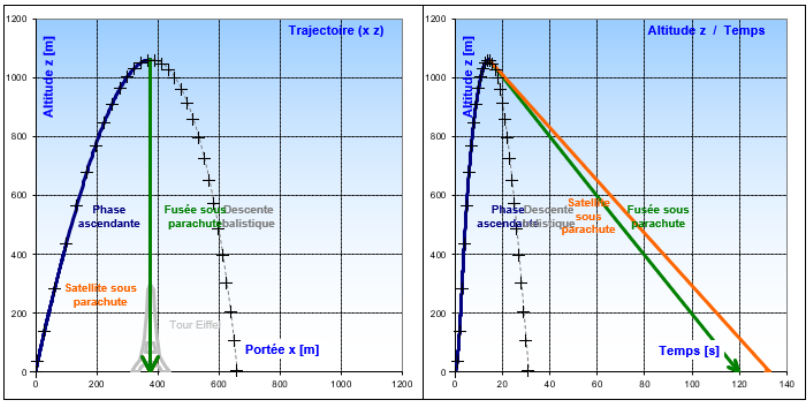
Trajectographie de fusée

Remplir les cases jaunes

	Fusée
Nom	DreamAir
Club	AeroIPSA
Masse totale	9.545 kg
Propulseur	Barasinga (Pro54-5G C)

	Traînée Aérdynamique
Surface Réf.	0.020762 m²
Cx	0.5

	Rampe de Lancement
Longueur	4 m
Élévation	80 °
Altitude	0 m



	DescenteSousParachute	
	Fusée	1 satellite
Masse	7.95 kg	0.562 kg
Dépotage	N/A	
Ouverture para	14.3 s	14.3 s
Surface para	1.25 m²	0.112 m²
Cx parachute	1	1
Vitesse du vent	5 m/s	5 m/s
Vitesse descente	10.1 m/s	9.0 m/s
Durée descente	105 s	118 s
Durée du vol	119 s	133 s
Déport latéral	± 525 m	± 591 m

18/07/2023	Temps	Altitude z	Portée x	Vitesse	Accélération	Efforts
Sortie de Rampe				24.6 m/s		
Vit max & Acc max				164 m/s	84 m/s²	
Largage du satellite	14.3 s	1060 m	376 m	22 m/s		34.3 N
Culmination, Apogée	14.3 s	1060 m	376 m	22 m/s		
Ouverture parachute fusée	14.3 s	1060 m	376 m	22 m/s		382.8 N
Impact balistique	31.1 s	~0 m	660 m	103.4 m/s		42536 J

	Pour localiser la fusée
Couleur fuselage/coiffe	Brun/Orange...
Couleur parachute fusée	Rouge...
Couleur parachute satellite	Jaune

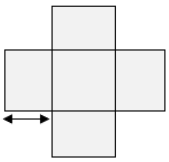
propu OK

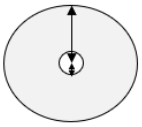
v3.4.2

Commentaire libre :

Calcul de la surface d'un parachute

Longeur du bord	500 mm
Largeur du coté	500 mm
Surface para	1.25 m²
Rayon exterieur	299 mm
Rayon intérieur	29 mm
Surface para	0.28 m²





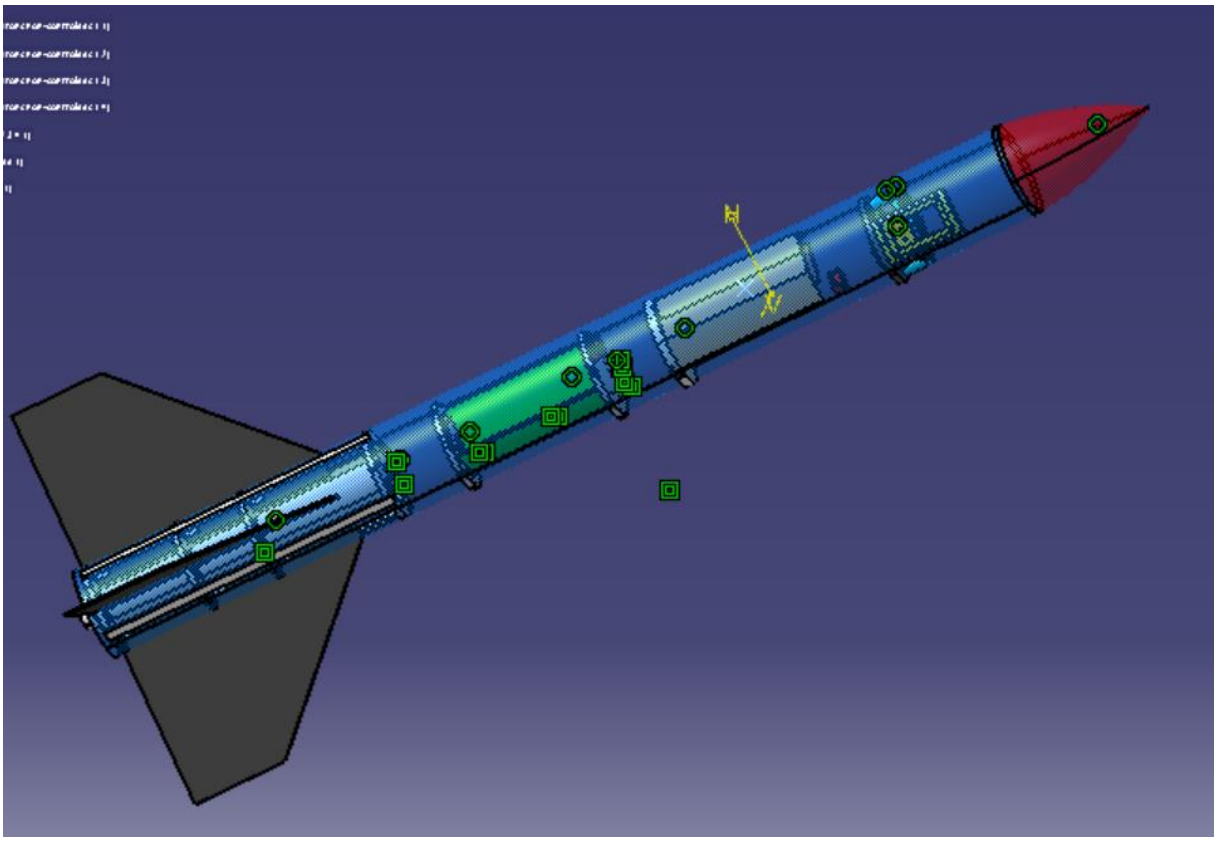
Résultats détaillés	Temps	Altitude z	Portée x	Vitesse	Accélération	Angle
	s	m	m	m/s	m/s²	°
Décollage	0	0	0	0	-	**80**
Sortie de Rampe	0.33	3.72	0.66	24.6	78.7	80.0
Vit max & Acc max	-	-	-	**164**	**83.9**	-
Fin de Propulsion	3.6	376	86	158	27.6	76.0
Culmination, Apogée	**14.3**	**1060**	376	22	9.8	2.3
Impact balistique	31.1	~0	**660**	103	2.0	-84.4
Ouverture parachute fusée	14.3	**1060**	**376**	**22**	9.8	2.3
Impact fusée sous para.	**119**	~0	-149 | 901	10	9.8	-
Largage du satellite	14.3	**1060**	376	**22**	9.8	2.3
Impact du satellite	**133**	~0	-215 | 967	9	9.8	**-**

3.2 Ailerons et corps de la fusée

Les ailerons ont été réalisés à l'association en fibre de carbone en sandwich avec une fine couche de mousse. Les ailerons sont composés de 8 feuilles de carbone de 150g/m2 de chaque côté de la mousse.

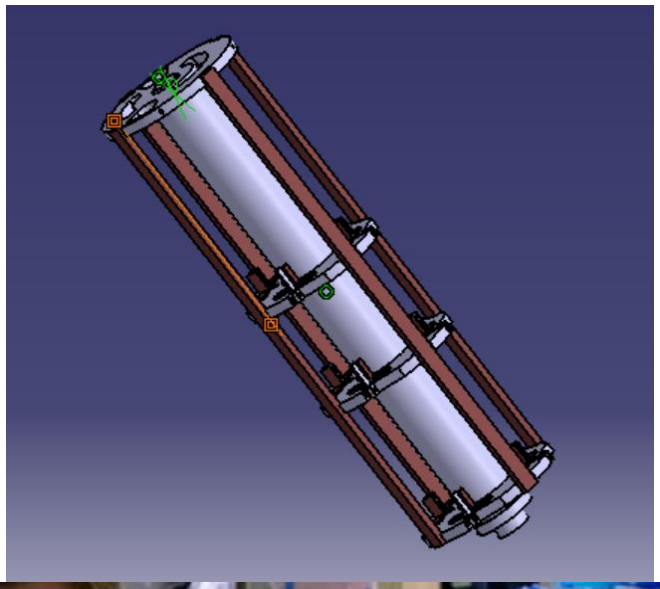
Le tube quant a été lui fut réalisé en fibre de verre et de la mousse autour d'un moule de PLA.

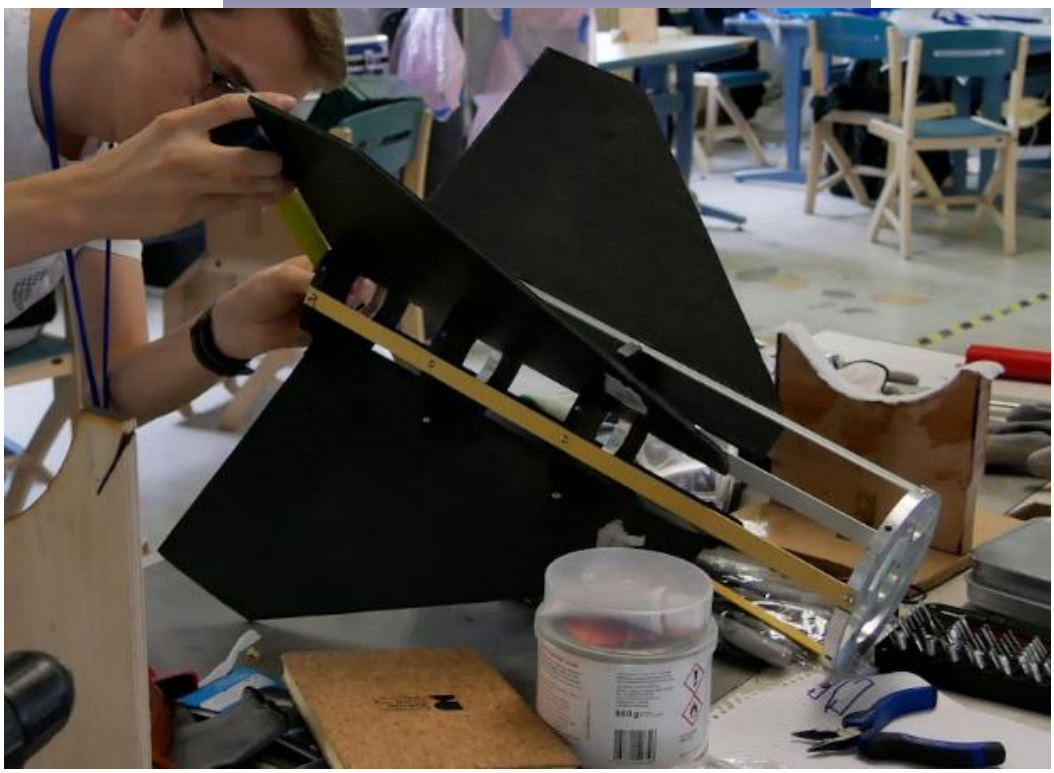
La structure de la fusée est dite « peau porteuse » c'est-à-dire que la peau de la fusée porte les blocs



3.3 Bloc Propulsion

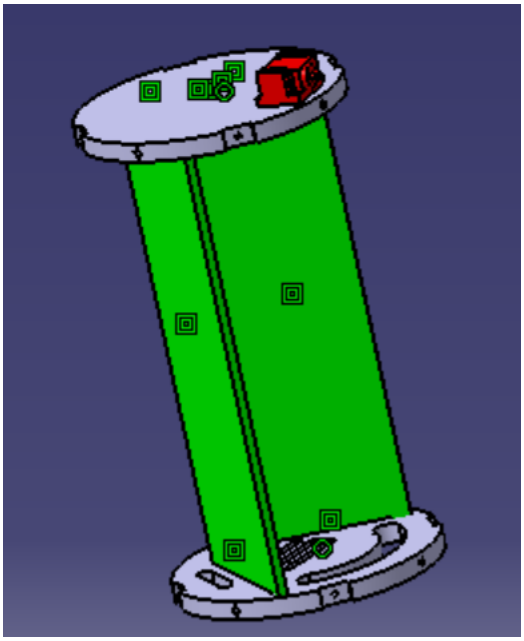
Le bloc de propulsion est composé de 4 bagues. La reprise de poussée est réalisée par la bague haute en aluminium, et les 3 autres bagues servent à centrer un moteur Pro-54.





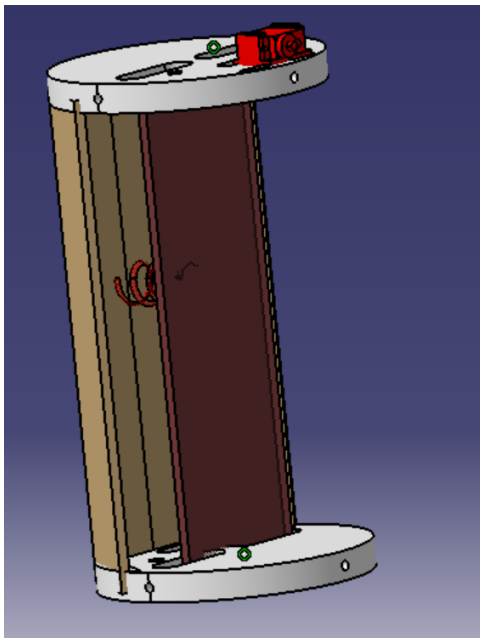
3.4 Bloc Parachute

Le bloc parachute est composé d'une cage contenant le parachute. Le parachute est lié à une sangle, elle-même attachée à la bague inférieure en aluminium. Celle-ci supporte le poids de la fusée lors de la descente.



3.5 Bloc CanSat

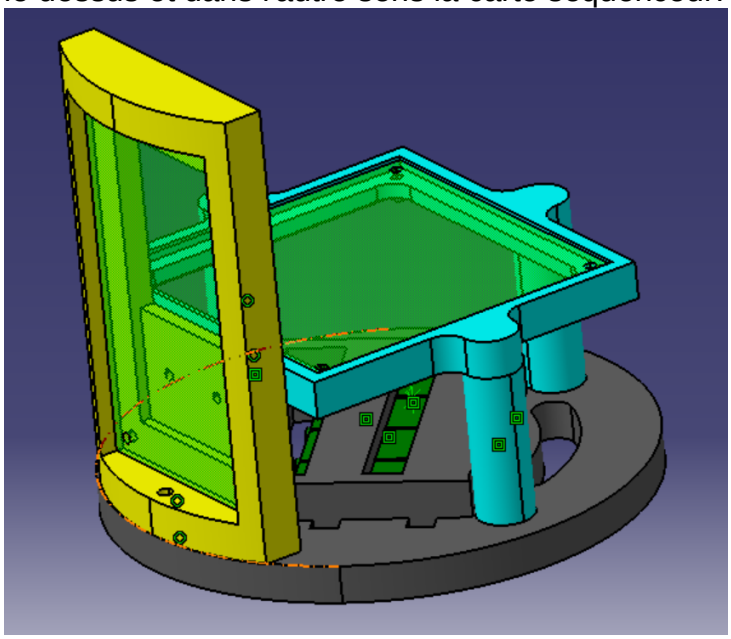
Le Bloc CanSat est responsable de l'éjection du CanSat GreenAir. L'éjection s'effectue grâce à un ressort poussant une plaque en bois, écrasant ainsi le CanSat entre la plaque et la face intérieure de la trappe du bloc. Lorsque le moteur libère la trappe, le CanSat est alors éjecté avec son parachute.



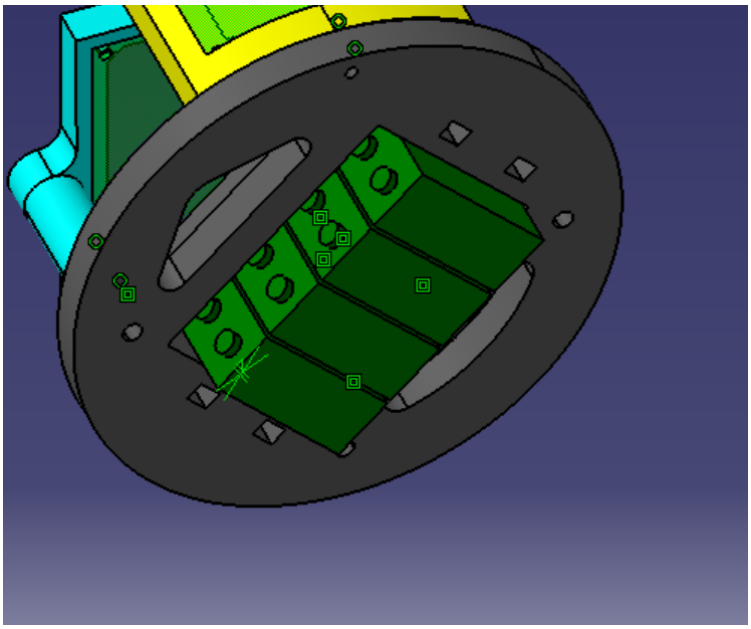
3.6 Bloc Electronique

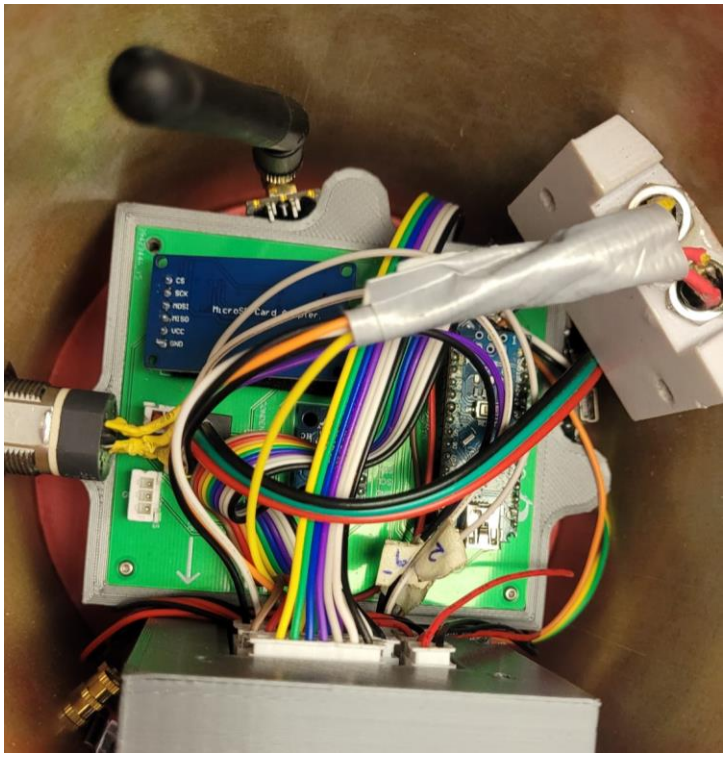
L'électronique de la fusée est le premier bloc en termes de hauteur dans la structure. Elle se compose de 3 cartes électroniques : le séquenceur, l'expérience et l'interface.

Deux ordinateurs de bord sont présents, l'un dans l'expérience et l'autre dans le séquenceur. Les pièces sont réalisées en 3D, la carte interface vient s'insérer dans la plaque jaune, la plaque bleue supporte deux cartes, la carte expérience sur le dessus et dans l'autre sens la carte séquenceur.



Les piles sont situées tout en bas, et tenus par des Serflex.

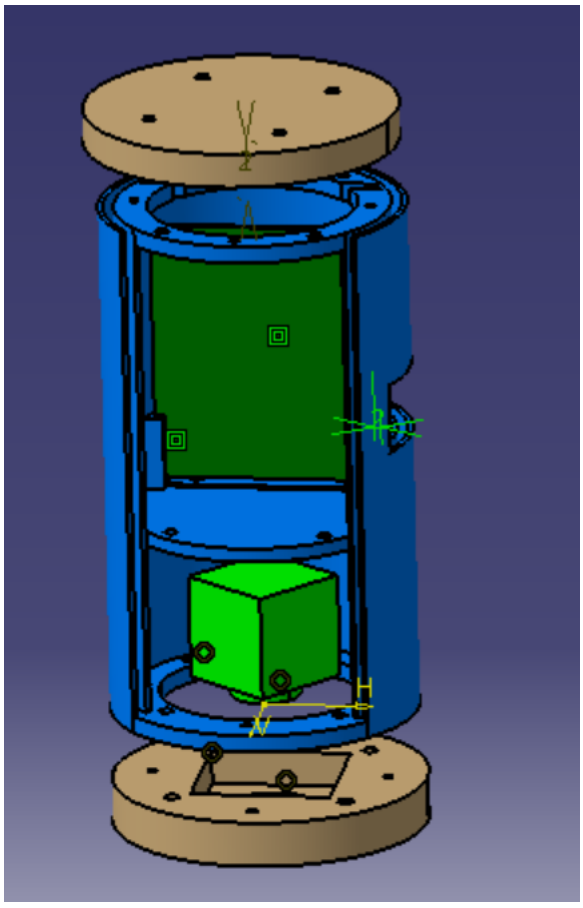




4. Expérience

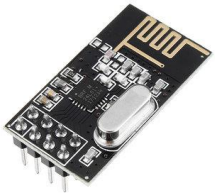
4.1 CanSat GreenAir

Le CanSat est composé de 4 pièces : la bague du dessus, la bague du dessous, le corps et la porte. Il est équipé d'une carte électronique permettant l'allumage de la fusée ainsi qu'un IMU permettant d'enregistrer les chocs subis par le CanSat.



4.2 Télémesure

La télémesure est effectuée avec le module nRF24L01. En 2.4 GHz avec un débit de 5dBi



La télémesure est effectué par la carte expérience et permet de tester la réception dans les conditions de vols de DreamAir

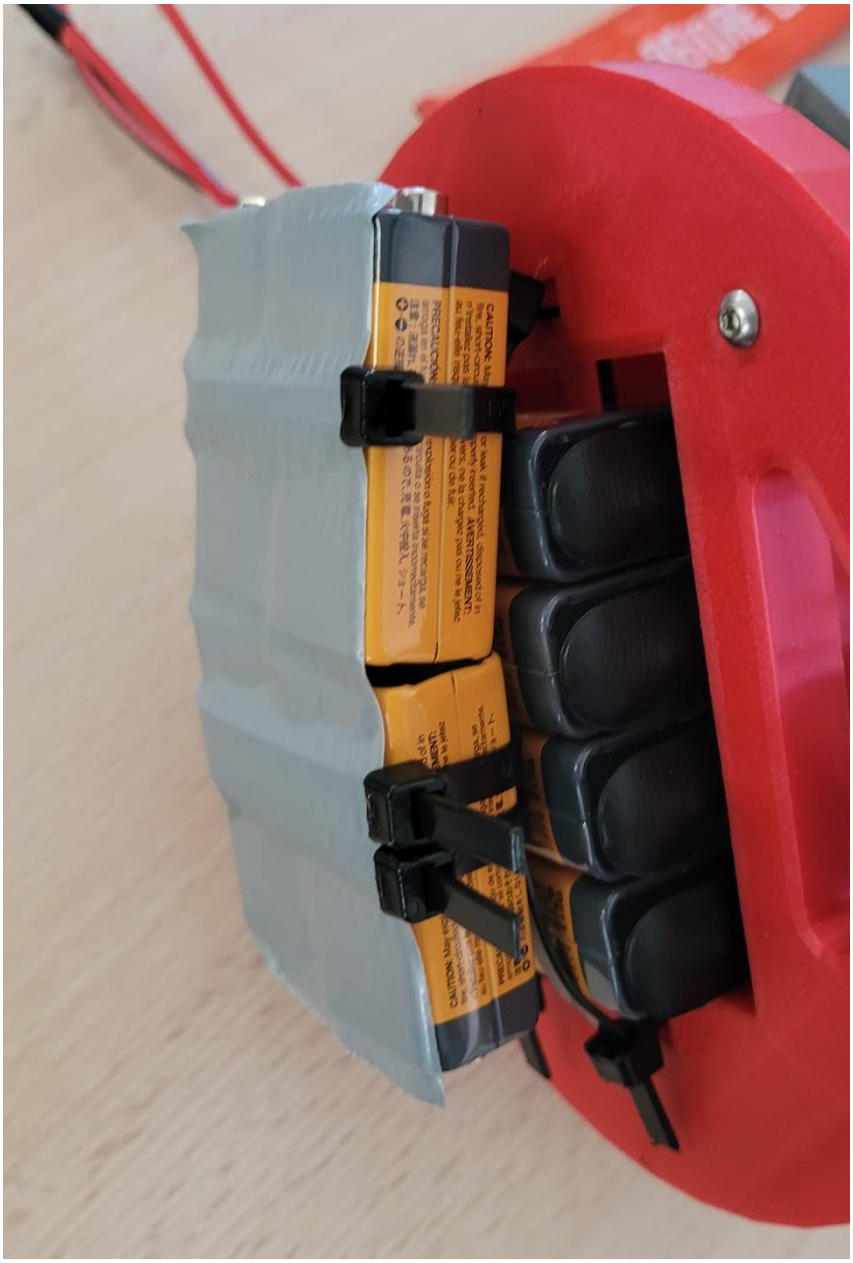
5. Description Electronique

5.1 Electronique DreamAir

L’électronique de DreamAir a été conçu afin d’avoir l’expérience et le séquenceur électroniquement isolé.

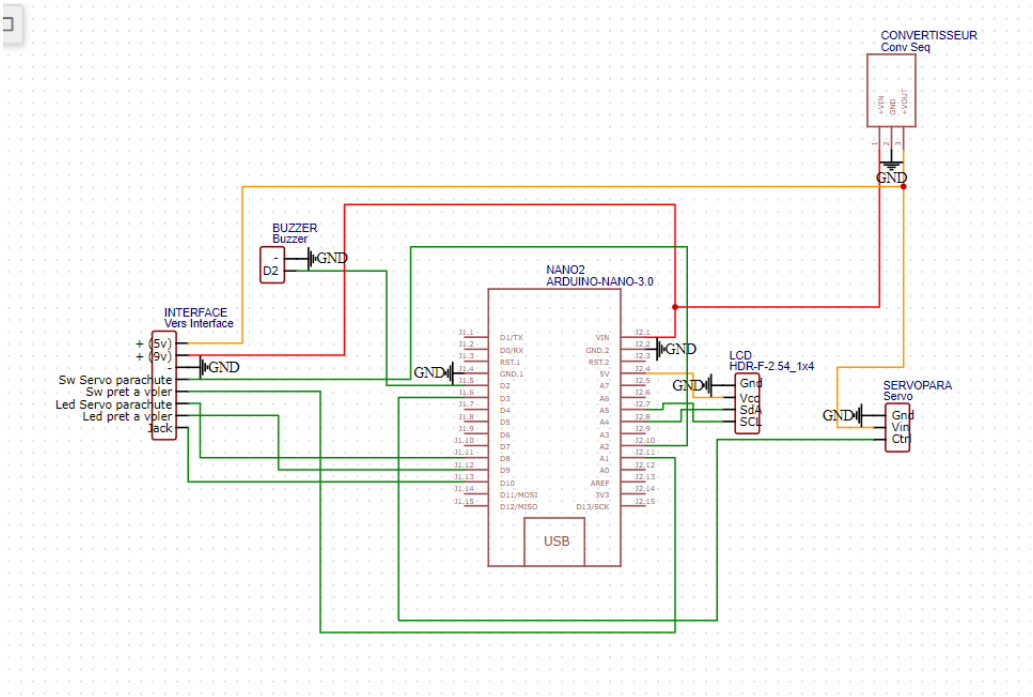
5.1.1 Batteries

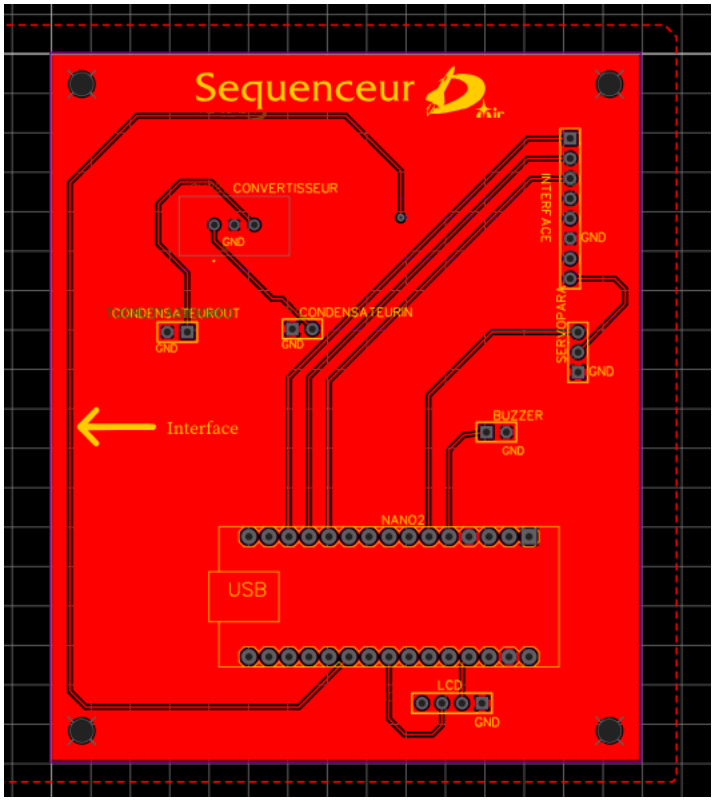
Les batteries sont situées sous l'électronique. Deux batteries 9V sont connectées au séquenceur et six batteries alimentent l'expérience. La durée des batteries était d'environ 5 heures. L'alimentation des batteries passait d'abord par la carte interface, qui servait de relais aux autres cartes.



5.1.2 Carte séquenceur

La carte séquenceur alimente principalement le moteur qui déclenche le parachute. Pour cela, elle est composée d'un ordinateur de bord responsable de la minuterie, d'un convertisseur 9V-5V pour alimenter le moteur, d'un petit servomoteur, d'un écran LCD et d'un buzzer qui finalement n'ont pas été utilisés, ainsi que d'un connecteur JST vers l'interface.



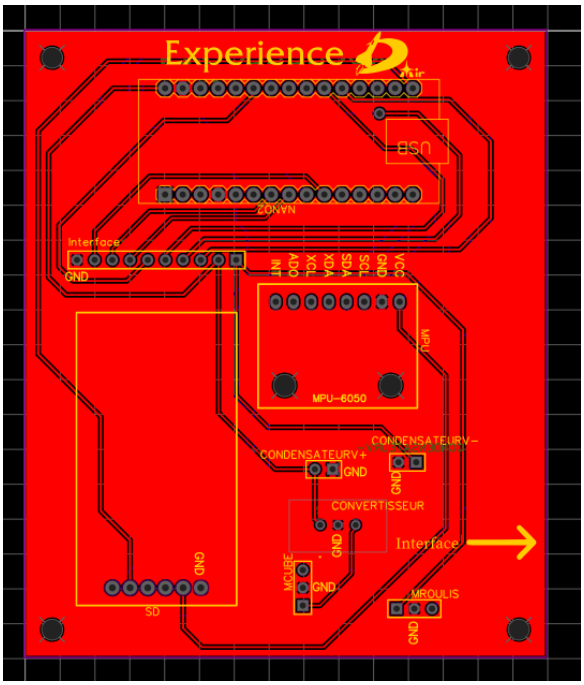


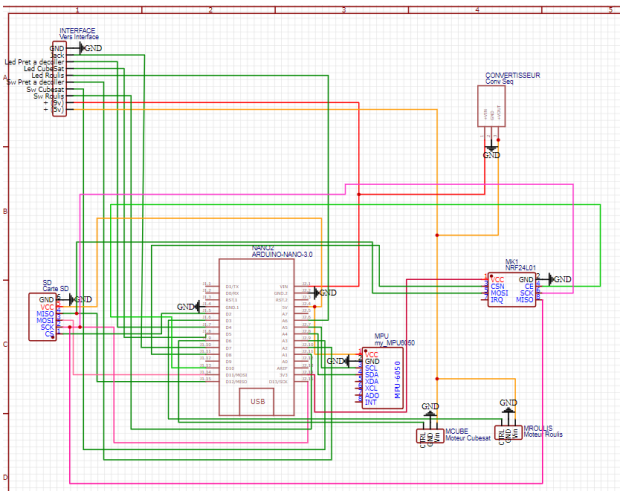
Le code Arduino du séquenceur est également très basique. Il effectue la minuterie et affiche sur l'interface l'état du moteur et de la minuterie.

Une commande a été ajoutée afin que les commandes des interrupteurs ne fonctionnent pas tant que le jack n'est pas inséré. Cela permet également de toujours allumer la fusée et de commencer la chronologie dans les mêmes conditions.

5.1.3 Carte expérience

La carte expérience alimente de nombreuses choses dans la fusée. Tout d'abord, elle comporte une minuterie pour le largage du CanSat, liée à un servomoteur. Également, un deuxième moteur qui n'était pas utilisé initialement pour le contrôle de roulis. La carte possède aussi la télémétrie de la fusée.

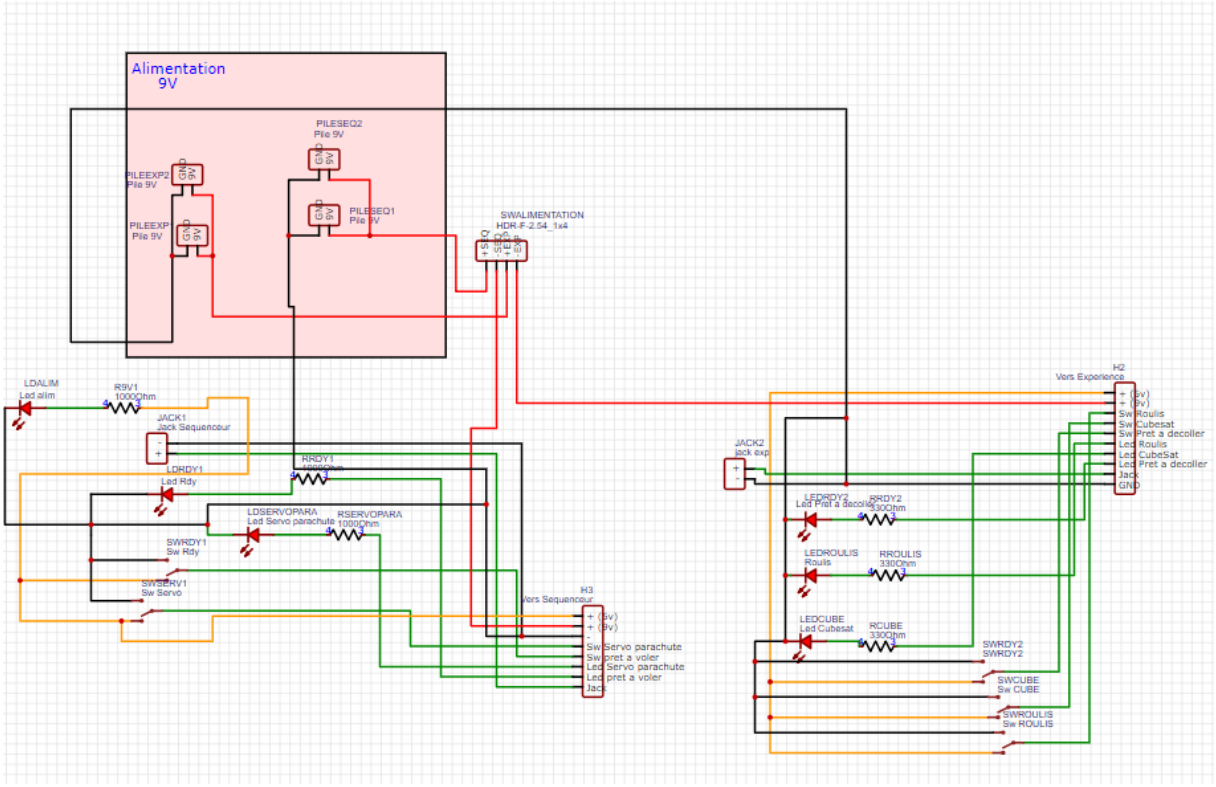


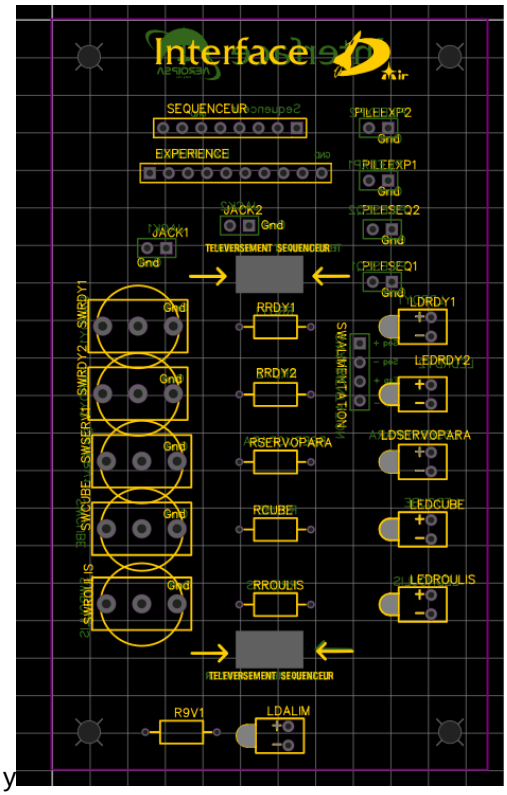


Comme pour l'expérience dans le code, la minuterie est lancée par le décrochage du jack, et sans se jack, on ne peut pas interagir avec l'UI.

5.1.4 Carte Interface

La carte interface est la carte UI de la fusée où l'on peut interagir avec la fusée. Elle affiche des LEDs pour connaître l'état de chaque carte, et c'est à partir de là que partent tous les câbles tout en isolant l'interface de l'expérience.





La carte possède 6 Leds : une pour l'alimentation, une pour chacun des jacks, une pour le déclanchement du servomoteur du parachute, une Leds pour le servomoteur responsable du largage du CanSat et enfin un annonçant le début du contrôle de roulis (non utilise)

5.2 Electronique CanSat GreenAir

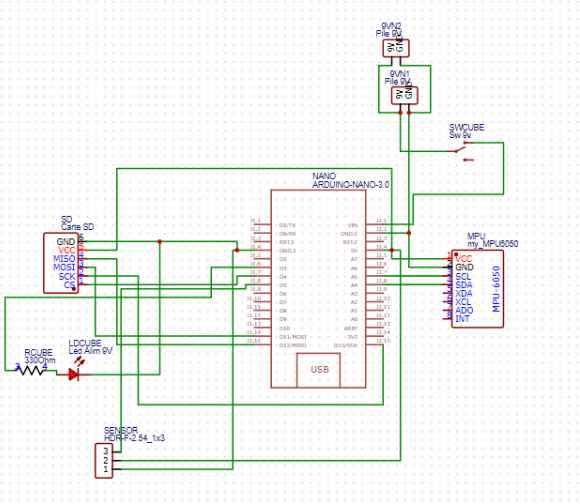
5.2.1 Batterie

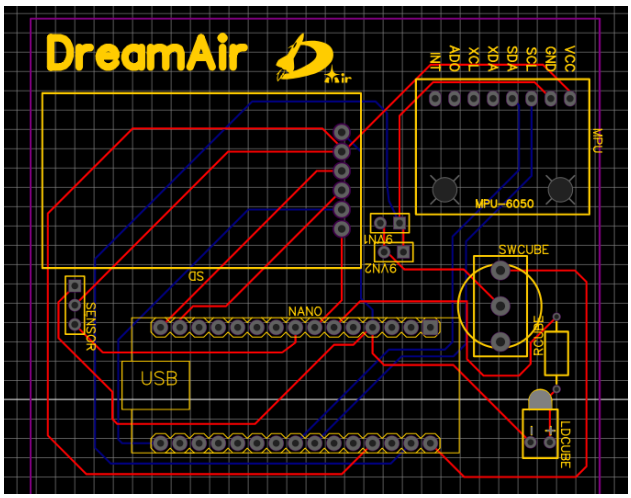
Le CanSat GreenAir est composé de 4 batteries 9v pour alimenter l'ordinateur de bord.

5.2.2 Carte CanSat

La carte CanSat est composé d'un Arduino nano, avec un IMU, une carte SD pour la récupération des données et enfin un buzzer.

Le buzzer sonne au bout de 3h jusqu'à qu'il n'est plus de batterie afin d'aider à la récupération.





5.2.3 Caméra vidéo

Une RunCam 5 a été placé en dessous du CanSAT pour filmer le largage. La RunCam 5 fut choisi pour son grand angle de camera et sa forme cubique facilitant

son intégration.

6. Déroulement du vol :

6.1 Condition de vol

Le vol s'est déroulé le mercredi 19 juillet 2023. Suite aux restrictions des aéroports voisins, le vol prévu le matin a été décalé à l'après-midi. Le vent soufflait légèrement, et le ciel était couvert d'un plafond nuageux à une hauteur de quelques centaines de mètres.

La fusée a volé sans se déstabiliser. Le parachute s'est bien déployé, et le CanSat s'est éjecté. La descente de la fusée a pu être filmée par le CanSat.

6.2 Récupération

6.2.1 Fusée DreamAir

La récupération de la fusée s'est effectuée 2 heures et 55 minutes après le décollage. La fusée a atterri à quelques centaines de mètres. Le séquenceur et l'expérience étaient toujours en marche (avec encore de la batterie).

6.2.2 CanSat GreenAir

Le CanSat fut trouvé heureusement à une dizaine de mètre de la fusée. Celui-ci était toujours en marche. Le buzzer du CanSat s'est mis à sonner au moment où nous nous approchions du CanSat.

7. Résultats

7.1 Télémesure

La télémesure, faite est étudié en post-vol par Nasca :

Expérience : Evaluation Module télémesure NRF24L01+PA+LNA
Portée théorique maximale : 800m - 1.1km Altitude max du vol : 1.5 km

Objectif : vérifier expérimentalement en conditions de vol les circonstances de décrochage puis de reprise de la liaison télémesure du module.

Données : Puissance émise : 20 dBm (100 mW)
Fréquence porteuse : 2 442 MHz
Débit d'émission : 1 Mbps Bandwidth : 900 kHz

Résultat du vol :

- **00s** Décollage
- **03,17s** perte momentanée
- **05,48s** perte définitive 575m (550m Altitude et 170m distance sol)
- **43,58s** réception fragmentaire
- **73,69s** réception stable 560m (100m Altitude et 550m distance sol)
- **113,71s** fin transmission

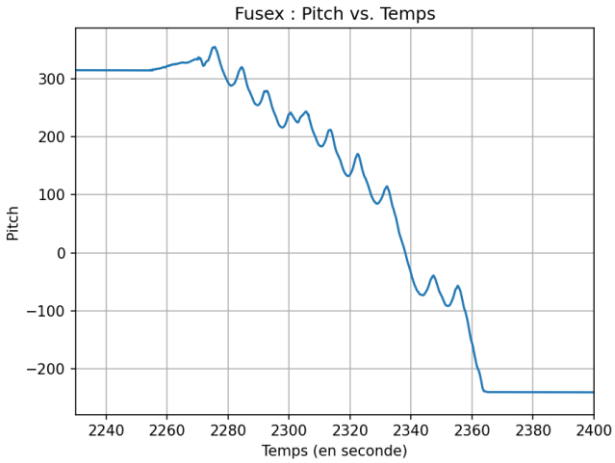
Résultat : le module est donc pleinement qualifié pour une utilisation partielle en vol fusex et complète en vol minif. Dans ce dernier cas, il constitue un intéressant point d'équilibre par sa consommation et son encombrement spatial réduit. Il est également utilisable pour une télédétection d'état pour la plupart des projets, pendant le début et la fin du vol, voire sa récupération couplée avec un GPS (ou n'omettant de le disposer opposer à la partie de la fusée touchant le sol en première). Ce module ne peut cependant être viable pour collecter des données expérimentales volumineuses et/ou devant être transmises à fréquence fixe.

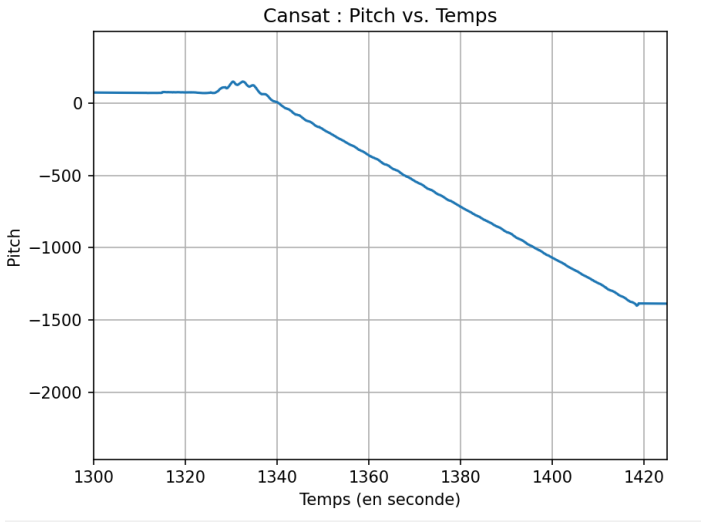
7.2 IMU DreamAir Comparaison IMU GreenAir

Nous allons comparer les 3 axes de rotation de la fusée et du CanSat. Les fichiers ont été analysés avec un script Python.

On notera que les graphiques ont été placés sur l'axe X au moment du décollage, qui est à 2250 secondes sur les graphiques de la fusée et à 1312 secondes sur les graphiques du CanSat.

On notera également que les deux capteurs dérivent négativement, mais uniquement à partir du décollage et non-pas au repos avant ou après le vol.

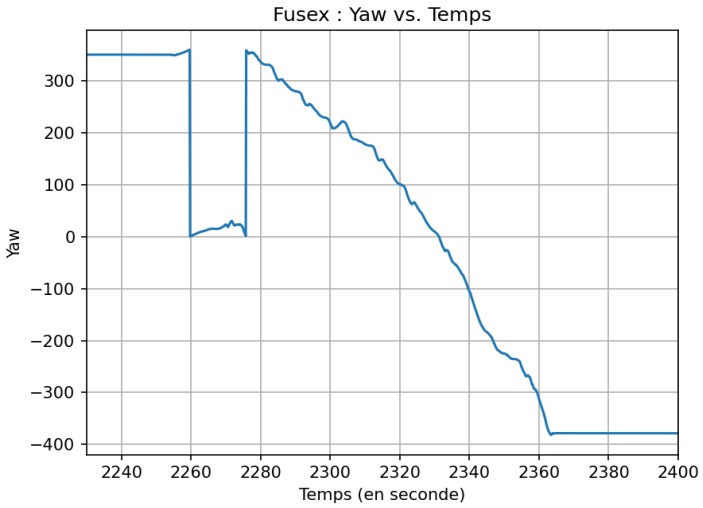


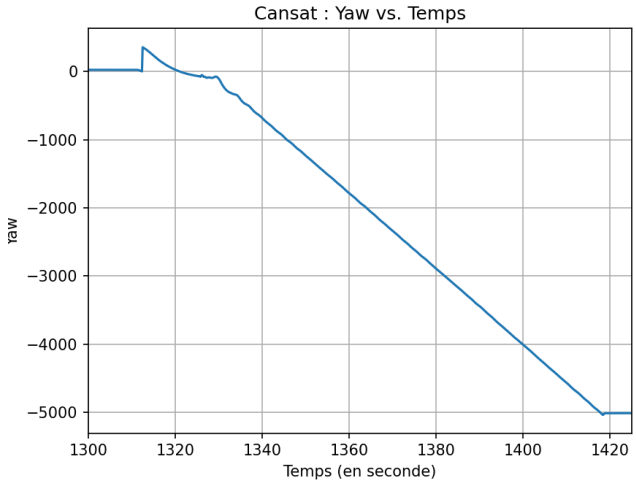


Sur l'axe de tangage (pitch) au niveau de la Fusex, il y a un mouvement constant sous forme d'oscillation. Cela représente probablement la descente en parachute, les mouvements de la fusée sur cet axe.

Le CanSat est très stable lors du décollage, mais on peut apercevoir un peu plus de mouvement sous forme d'un pic. Ceci représente probablement l'éjection du CanSat et sa stabilisation. Ensuite, le CanSat a eu un tangage constant jusqu'à son atterrissage.

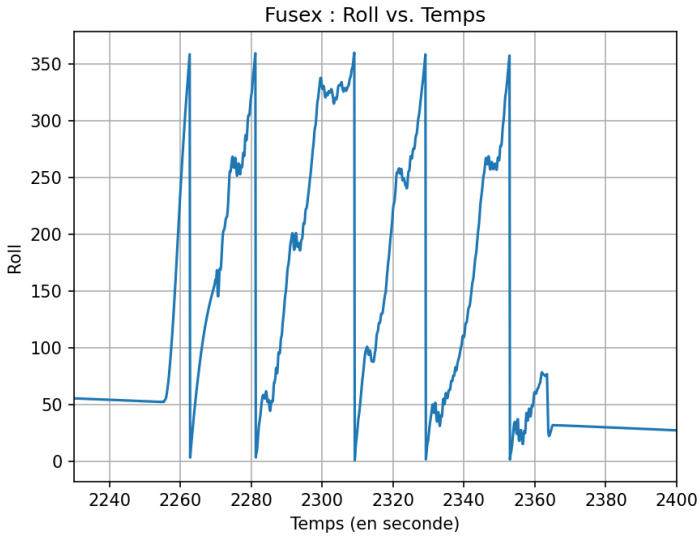
Pour le lacet (Yaw), il y a une descente très abrupte qui représente une erreur de code ramenant le lacet à 0 au-dessus de 360 degrés. On observe donc sur la Fusex et le CanSat un pic dans le lacet, et dans les deux cas, cela semble être le moment du décollage.

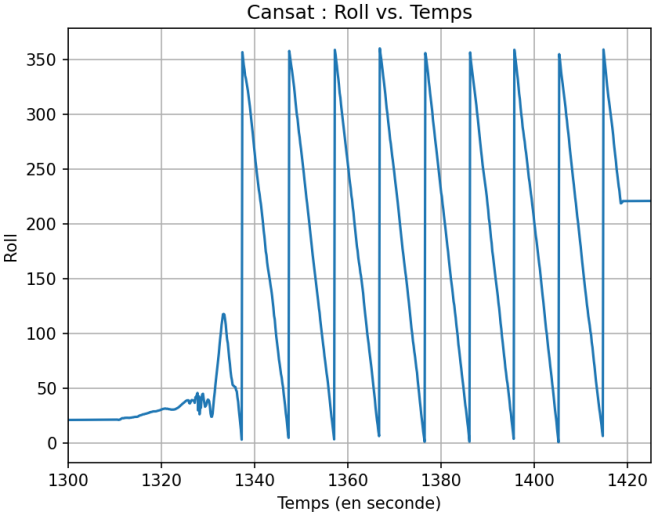




En ce qui concerne le roulis (roll), la Fusex et le CanSat ont montré un roulis élevé. On observe également un grand roulis pour la Fusex même après le déploiement du parachute.

Le CanSat a montré un roulis imprécis au tout début lors de l'éjection, mais on observe ensuite que le roulis se stabilise pendant la descente avec une période presque constante. Cela peut également être observé sur la vidéo prise par le CanSat.





7.3 Etat de la structure

7.3.1 Fusée DreamAir

Quelques morceaux de terre et quelques rayures, à part cela, rien d'autre n'a été noté sur le corps de la fusée. À l'intérieur, tout a tenu, et il n'y a rien de particulier à noter. Les ailerons et toute la structure sont encore droits. En changeant les piles, la fusée serait prête à voler à nouveau.

7.3.2 CanSat GreenAir

Le CanSat était encore en très bon état après le vol. Rien de particulier à signaler à l'extérieur. Cependant, la structure interne était sens dessus dessous, surtout les piles qui avaient bougé.

8. Conclusion

Au bout de 2 ans, l'aventure DreamAir arrive à son terme. Grâce à une collaboration déterminée, l'équipe a conçu, lancé et récupéré DreamAir ainsi que son CanSat avec succès. Sa structure mécanique a prouvé qu'elle était plus que suffisante et rigide pour le vol, l'électronique a tenu du début jusqu'à la fin, et les résultats obtenus, bien qu'imparfaits, laissent la voie pour de nouvelles opportunités d'amélioration.

Le projet a été l'occasion pour beaucoup de membres de toucher à de nouveaux domaines. Certains ont appris à réaliser des circuits imprimés tandis que d'autres ont appris à fabriquer des composites. DreamAir témoigne d'un projet qui a connu des moments difficiles, mais qui a été poussé par la persévérance et la ténacité de toute l'équipe. En espérant que le corps de la fusée devant l'association témoignera pour les années à venir du travail accompli et de l'engagement des membres de DreamAir pour AeroIpsa.

9. Annexe

9.1 Code Arduino CanSat

```
#include <Wire.h>
#include <SPI.h>
#include <Time.h>
#include <SD.h>
File myFile;

long int DEBUT_TEMPS;


int pin_buzzer = 5;
const int MPU = 0x68; // MPU6050 I2C address
float AccX, AccY, AccZ;
float GyroX, GyroY, GyroZ;
float accAngleX, accAngleY, gyroAngleX, gyroAngleY, gyroAngleZ;
float roll, pitch, yaw;
float AccErrorX, AccErrorY, GyroErrorX, GyroErrorY, GyroErrorZ;
float elapsedTime, currentTime, previousTime;
int c = 0;
int Led = 3;

void setup() {

  pinMode(pin_buzzer, OUTPUT);

  pinMode(Led, OUTPUT);
  digitalWrite(Led, HIGH);
  Serial.begin(19200);
  //SD BEGIN ----------------------------------------

  while (!Serial) {

  Serial.print("TEST  SD card...");
  }

  if (!SD.begin(4)) {
    Serial.println("initialization failed!");
    while (1);
  }
  Serial.println("initialization done.");


  //SD END ----------------------------------------



  Wire.begin();
  Wire.beginTransmission(MPU);
  Wire.write(0x6B);
  Wire.write(0x00);
  Wire.endTransmission(true);


  Wire.beginTransmission(MPU); // Sensi l'accelero
  Wire.write(0x1C);
  Wire.write(0x10);
  Wire.endTransmission(true);
  //Range de l'accelero
  Wire.beginTransmission(MPU);
  Wire.write(0x1B);
  Wire.write(0x10);
  Wire.endTransmission(true);
  delay(20);

  // Valeur d'erreur a corrigé
  calculate_IMU_error();
  delay(20);


myFile = SD.open("vRAI.txt", FILE_WRITE);

myFile.println("\n\n--------Debut donnes vol----------");
myFile.close();
```

```
}
void loop() {

while (millis() <= 5400000) {
  digitalWrite(pin_buzzer, HIGH);

    //SD BEGIN ----------------------------------------
myFile = SD.open("vRAI.txt", FILE_WRITE);
     //SD END ----------------------------------------



  //  Lecture de l'accelero
  delay(100);
  Wire.beginTransmission(MPU);
  Wire.write(0x3B); // Start with register 0x3B (ACCEL_XOUT_H)
  Wire.endTransmission(false);
  Wire.requestFrom(MPU, 6, true); // Read 6 registers total, each axis value is stored in 2 registers
  //For a range of +-2g, we need to divide the raw values by 16384, according to the datasheet
  AccX = (Wire.read() << 8 | Wire.read()) / 16384.0; // X-axis value
  AccY = (Wire.read() << 8 | Wire.read()) / 16384.0; // Y-axis value
  AccZ = (Wire.read() << 8 | Wire.read()) / 16384.0; // Z-axis value
  // Calcul Roll et Pitch (X et Y)
  accAngleX = (atan(AccY / sqrt(pow(AccX, 2) + pow(AccZ, 2))) * 180 / PI) - 0.58; // AccErrorX ~(0.58) See the calculate_IMU_error()custom function for more details
  accAngleY = (atan(-1 * AccX / sqrt(pow(AccY, 2) + pow(AccZ, 2))) * 180 / PI) + 1.58; // AccErrorY ~(-1.58)

  // Lecture Gyro //
  previousTime = currentTime;        // Previous time is stored before the actual time read
  currentTime = millis();            // Current time actual time read
  elapsedTime = (currentTime - previousTime) / 1000; // Divide by 1000 to get seconds
  Wire.beginTransmission(MPU);
  Wire.write(0x43); // Gyro data first register address 0x43
  Wire.endTransmission(false);
  Wire.requestFrom(MPU, 6, true); // Read 4 registers total, each axis value is stored in 2 registers
  GyroX = (Wire.read() << 8 | Wire.read()) / 131.0; // For a 250deg/s range we have to divide first the raw value by 131.0, according to the datasheet
  GyroY = (Wire.read() << 8 | Wire.read()) / 131.0;
  GyroZ = (Wire.read() << 8 | Wire.read()) / 131.0;
  // Corrige valeurs
  GyroX = GyroX - GyroErrorX; // GyroErrorX ~(-0.56)
  GyroY = GyroY - GyroErrorY; // GyroErrorY ~(2)
  GyroZ = GyroZ - GyroErrorZ; // GyroErrorZ ~ (-0.8)

  // Currently the raw values are in degrees per seconds, deg/s, so we need to multiply by sendonds (s) to get the angle in degrees
  gyroAngleX = gyroAngleX + GyroX * elapsedTime; // deg/s * s = deg
  gyroAngleY = gyroAngleY + GyroY * elapsedTime;
  yaw =  yaw + GyroZ * elapsedTime;

  // Filtre Complementaire - combine acceleromter and gyro angle values
  roll = 0.96 * gyroAngleX + 0.04 * accAngleX;
  pitch = 0.96 * gyroAngleY + 0.04 * accAngleY;


  if (roll > 360) {
    roll = roll - 360;
     }
  if (roll < 0) {
    roll = roll + 360;
     }


  if (pitch > 360) {
    pitch = pitch - 360;
     }
  if (pitch < 0) {
    pitch = pitch + 360;
     }


  if (yaw > 360) {
    yaw = yaw - 360  ;
     }
  if (yaw < 0) {
    yaw = yaw + 360  ;
     }



  // Print the values on the serial monitor
  Serial.print(AccX);
  Serial.print("/");
  Serial.print(AccY);
  Serial.print("/");
  Serial.println(AccZ);
```

```
          //SD BEGIN ----------------------------------------
          String TIME = String(millis(), 10);
        //  myFile.println("time : " + TIME + " ms");

          String YAW = String(yaw, 2);
          String PITCH = String(pitch, 2);
          String ROLL = String(roll, 2);
         // myFile.println("Yaw : " + YAW);
         // myFile.println("pitch : " + PITCH);
         // myFile.println("roll : " + ROLL);

          String ACCX = String(AccX, 2);
          String ACCY = String(AccY, 2);
          String ACCZ = String(AccZ, 2);

         // myFile.println("Acc_X : " + ACCX);
         // myFile.println("Acc_Y : " + ACCY);

          String ACCANGLEX = String(accAngleX, 2);
          String ACCANGLEY = String(accAngleY, 2);
         // myFile.println("Acc_Angle_X : " + ACCANGLEX);
        //  myFile.println("Acc_Angle_Y : " + ACCANGLEY);

          String gyroX = String(GyroX, 2);
          String gyroY = String(GyroY, 2);
          String gyroZ = String(GyroZ, 2);

         // myFile.println("gyro_Y : " + gyroY);
        //  myFile.println("gyro_Z : " + gyroZ);

          String GYROANGLEX = String(gyroAngleX, 2);
          String GYROANGLEY = String(gyroAngleY, 2);

         //  myFile.println("Angle_Gyro_X : " + GYROANGLEX);
         //  myFile.println("Angle_Gyro_Y : " + GYROANGLEY);

      myFile.println( TIME+ ","+YAW + "," +PITCH+ "," +ROLL+ "," +ACCX+ "," +ACCY + "," +ACCZ + "," +ACCANGLEX+ "," +ACCANGLEY+ "," +gyroX+ "," +gyroY+ "," +gyroZ+ "," +GYROANGLEX+ "," +GYROANGLEY );

      myFile.close();

          //SD END ----------------------------------------
          }

      while (millis() >= 10800000) { // 3 heures

        digitalWrite(pin_buzzer, HIGH);
        delay(3000);
        digitalWrite(pin_buzzer, LOW);
        delay(3000);
    }
    }

    void calculate_IMU_error() {
      // We can call this funtion in the setup section to calculate the accelerometer and gyro data error. From here we will get the error values used in the above equations printed on the Serial Monitor.
      // Note that we should place the IMU flat in order to get the proper values, so that we then can the correct values
      // Read accelerometer values 200 times
      while (c < 200) {
        Wire.beginTransmission(MPU);
        Wire.write(0x3B);
        Wire.endTransmission(false);
        Wire.requestFrom(MPU, 6, true);
        AccX = (Wire.read() << 8 | Wire.read()) / 16384.0 ;
        AccY = (Wire.read() << 8 | Wire.read()) / 16384.0 ;
        AccZ = (Wire.read() << 8 | Wire.read()) / 16384.0 ;
        // Sum all readings
        AccErrorX = AccErrorX + ((atan((AccY) / sqrt(pow((AccX), 2) + pow((AccZ), 2))) * 180 / PI));
        AccErrorY = AccErrorY + ((atan(-1 * (AccX) / sqrt(pow((AccY), 2) + pow((AccZ), 2))) * 180 / PI));
        c++;
      }
      //Divide the sum by 200 to get the error value
      AccErrorX = AccErrorX / 200;
      AccErrorY = AccErrorY / 200;
      c = 0;
      // Read gyro values 200 times
      while (c < 200) {
        Wire.beginTransmission(MPU);
        Wire.write(0x43);
        Wire.endTransmission(false);
        Wire.requestFrom(MPU, 6, true);
        GyroX = Wire.read() << 8 | Wire.read();
        GyroY = Wire.read() << 8 | Wire.read();
        GyroZ = Wire.read() << 8 | Wire.read();
        // Sum all readings
```

```
    GyroErrorX = GyroErrorX + (GyroX / 131.0);
    GyroErrorY = GyroErrorY + (GyroY / 131.0);
    GyroErrorZ = GyroErrorZ + (GyroZ / 131.0);
    c++;
   }
   //Divide the sum by 200 to get the error value
   GyroErrorX = GyroErrorX / 200;
   GyroErrorY = GyroErrorY / 200;
   GyroErrorZ = GyroErrorZ / 200;
   // Print the error values on the Serial Monitor
   Serial.print("AccErrorX: ");
   Serial.println(AccErrorX);
   Serial.print("AccErrorY: ");
   Serial.println(AccErrorY);
   Serial.print("GyroErrorX: ");
   Serial.println(GyroErrorX);
   Serial.print("GyroErrorY: ");
   Serial.println(GyroErrorY);
   Serial.print("GyroErrorZ: ");
   Serial.println(GyroErrorZ);
  }
```

9.2 Code Arduino EXPERIENCE

```
#include <Servo.h>
#include <Time.h>
#include <Wire.h>
#include <SPI.h>
#include <SD.h>
#include <RF24.h>
File myFile;

#define pinCE   9           // On associe la broche "CE" du NRF24L01 à la sortie digitale D7 de l'arduino
#define pinCSN  10          // On associe la broche "CSN" du NRF24L01 à la sortie digitale D8 de l'arduino
#define tunnel  "PIPE1"       // On définit un "nom de tunnel" (5 caractères), pour pouvoir communiquer d'un NRF24 à l'autre
RF24 radio(pinCE, pinCSN);    // Instanciation du NRF24L01
const byte adresse[6] = tunnel;               // Mise au format "byte array" du nom du tunnel
const char message[] = "oiu";     // Message à transmettre à l'autre NRF24 (32 caractères maxi, avec cette librairie)
int val;   // Variable
long int temp = 0;

const int MPU = 0x68; // MPU6050 I2C address
float AccX, AccY, AccZ;
float GyroX, GyroY, GyroZ;
float accAngleX, accAngleY, gyroAngleX, gyroAngleY, gyroAngleZ;
float roll, pitch, yaw;
float AccErrorX, AccErrorY, GyroErrorX, GyroErrorY, GyroErrorZ;
float elapsedTime, currentTime, previousTime;
int c = 0;
int Led = 3;


const int Jack_Pin = 7;
bool countdownActive = false;
long int Temps_Passe = 0;
long int DEBUT_TEMPS;
bool fini = false;

const int Pin_Led_Pret_A_Vole_2 = 4;
const int Pin_Led_motor_CubeSat = 5;
const int Pin_Led_motor_Roulis = 8;

const int Pin_SW_Pret_A_Voler_2 = A2;
const int Pin_SW_motor_CubeSat = A3;
const int Pin_SW_motor_Roulis = A1;

const int Pin_motor_CubeSat = 6;
const int Pin_motor_Roulis = 3;

Servo servo_CubeSat;
Servo servo_Roulis;


void setup() {
  radio.begin();                    // Initialisation du module NRF24
  radio.openWritingPipe(adresse);     // Ouverture du tunnel en ÉCRITURE, avec le "nom" qu'on lui a donné
  radio.setPALevel(RF24_PA_MIN);       // Sélection d'un niveau "MINIMAL" pour communiquer (pas besoin d'une forte puissance, pour nos
essais)
  radio.stopListening();              // Arrêt de l'écoute du NRF24 (signifiant qu'on va émettre, et non recevoir, ici)
  val = 0;
```

```
  //-----------------------
  Serial.begin(9600);

  pinMode(Jack_Pin, INPUT_PULLUP);

  pinMode(Pin_SW_Pret_A_Voler_2, INPUT);
  pinMode(Pin_SW_motor_CubeSat, INPUT);
  pinMode(Pin_SW_motor_Roulis, INPUT);

  pinMode(Pin_Led_motor_Roulis, INPUT);
  pinMode(Pin_Led_motor_CubeSat, OUTPUT);
  pinMode(Pin_Led_Pret_A_Vole_2, OUTPUT);

  servo_CubeSat.attach(Pin_motor_CubeSat);
  servo_Roulis.attach(Pin_motor_Roulis);


   //SD Begin ------------------------------------------
   while (!Serial) {


  Serial.print("TEST  SD card...");
  }

  if (!SD.begin(2)) {
    Serial.println("initialization failed!");
    while (1);
  }
  Serial.println("initialization done.");



  Wire.begin();
  Wire.beginTransmission(MPU);
  Wire.write(0x6B);
  Wire.write(0x00);
  Wire.endTransmission(true);


  Wire.beginTransmission(MPU); // Sensi l'accelero
  Wire.write(0x1C);
  Wire.write(0x10);
  Wire.endTransmission(true);
  //Range de l'accelero
  Wire.beginTransmission(MPU);
  Wire.write(0x1B);
  Wire.write(0x10);
  Wire.endTransmission(true);
  delay(20);

  // Valeur d'erreur a corrigé
  calculate_IMU_error();
  delay(20);


 myFile = SD.open("abc.txt", FILE_WRITE);

 myFile.println("\n\n--------Debut donnes vol----------");
 myFile.close();

   //SD END ----------------------------------------

 }

 void loop() {
   if (  digitalRead(Jack_Pin) != LOW && digitalRead(Pin_SW_Pret_A_Voler_2)   == LOW ) {

     digitalWrite(Pin_Led_Pret_A_Vole_2, LOW);
     delay(300);
     digitalWrite(Pin_Led_Pret_A_Vole_2, HIGH);
     delay(300);

   }
   else{

   if (digitalRead(Jack_Pin) != LOW && countdownActive == false && digitalRead(Pin_SW_Pret_A_Voler_2) == HIGH)  {  // Si Jack arrache ET
le compteur pas actif
     countdownActive = true;  // Start the countdown
     DEBUT_TEMPS = millis();    // Store the start tim

   }

    if (countdownActive == true) {
      Temps_Passe = millis() - DEBUT_TEMPS;  // Calculate the elapsed time
```

```
    if (Temps_Passe >= 2000 && Temps_Passe <= 4000) {
      digitalWrite(Pin_Led_Pret_A_Vole_2, LOW);
    }


     if (Temps_Passe >= 4000 && Temps_Passe <= 6000) {
     digitalWrite(Pin_Led_Pret_A_Vole_2, LOW);
    }
     if (Temps_Passe >= 6000 && Temps_Passe <= 8000 ){
     digitalWrite(Pin_Led_Pret_A_Vole_2, HIGH);
     }

     if (Temps_Passe > 8000 && Temps_Passe <= 10000) {
     digitalWrite(Pin_Led_Pret_A_Vole_2, LOW);
    }
     if (Temps_Passe > 10000 && Temps_Passe <= 12000) {
     digitalWrite(Pin_Led_Pret_A_Vole_2, HIGH);
    }
     if (Temps_Passe > 14000) {
     digitalWrite(Pin_Led_Pret_A_Vole_2, LOW);
    }

   }
   if (Temps_Passe >= 14000 && fini == false) {  // Countdown finished


     servo_CubeSat.write(180);

     fini = true;

    }

   // PRET A VOLER JACK
  if (digitalRead(Pin_SW_Pret_A_Voler_2) == HIGH && countdownActive != true) {
    digitalWrite(Pin_Led_Pret_A_Vole_2, HIGH);

  } if (digitalRead(Pin_SW_Pret_A_Voler_2) == LOW && countdownActive != true) {
    digitalWrite(Pin_Led_Pret_A_Vole_2, LOW);
  }

  // ROULIS
  if (digitalRead(Pin_SW_motor_Roulis) == HIGH && digitalRead(Pin_Led_motor_Roulis) != HIGH  ) {
    digitalWrite(Pin_Led_motor_Roulis, HIGH);

     if ((millis-temp) >= 1000) {
      radio.write(&val, sizeof(val));     // Envoi de notre message
    val++;
    temp = millis();
    }

  }

  if (digitalRead(Pin_SW_motor_Roulis) == LOW && digitalRead(Pin_Led_motor_Roulis) != LOW  )
 {
    digitalWrite(Pin_Led_motor_Roulis, LOW);
  }

 // CUBESAT

  if (digitalRead(Pin_SW_motor_CubeSat) == HIGH && digitalRead(Pin_Led_motor_CubeSat) != HIGH && countdownActive != true  ) {
    servo_CubeSat.write(180);
    digitalWrite(Pin_Led_motor_CubeSat, HIGH);
  }

  if (digitalRead(Pin_SW_motor_CubeSat) == LOW && digitalRead(Pin_Led_motor_CubeSat) != LOW && countdownActive != true   )
 {
    servo_CubeSat.write(90);

    digitalWrite(Pin_Led_motor_CubeSat, LOW);
  }


  //SD Debut ----------------------------------------

 myFile = SD.open("abc.txt", FILE_WRITE);



  //  Lecture de l'accelero
  delay(100);
  Wire.beginTransmission(MPU);
  Wire.write(0x3B); // Start with register 0x3B (ACCEL_XOUT_H)
  Wire.endTransmission(false);
  Wire.requestFrom(MPU, 6, true); // Read 6 registers total, each axis value is stored in 2 registers
  //For a range of +-2g, we need to divide the raw values by 16384, according to the datasheet
  AccX = (Wire.read() << 8 | Wire.read()) / 16384.0; // X-axis value
```

```
AccY = (Wire.read() << 8 | Wire.read()) / 16384.0; // Y-axis value
AccZ = (Wire.read() << 8 | Wire.read()) / 16384.0; // Z-axis value
// Calcul Roll et Pitch (X et Y)
accAngleX = (atan(AccY / sqrt(pow(AccX, 2) + pow(AccZ, 2))) * 180 / PI) - 0.58; // AccErrorX ~(0.58) See the calculate_IMU_error()custom function for more details
accAngleY = (atan(-1 * AccX / sqrt(pow(AccY, 2) + pow(AccZ, 2))) * 180 / PI) + 1.58; // AccErrorY ~(-1.58)

// Lecture Gyro //
previousTime = currentTime;        // Previous time is stored before the actual time read
currentTime = millis();            // Current time actual time read
elapsedTime = (currentTime - previousTime) / 1000; // Divide by 1000 to get seconds
Wire.beginTransmission(MPU);
Wire.write(0x43); // Gyro data first register address 0x43
Wire.endTransmission(false);
Wire.requestFrom(MPU, 6, true); // Read 4 registers total, each axis value is stored in 2 registers
GyroX = (Wire.read() << 8 | Wire.read()) / 131.0; // For a 250deg/s range we have to divide first the raw value by 131.0, according to the datasheet
GyroY = (Wire.read() << 8 | Wire.read()) / 131.0;
GyroZ = (Wire.read() << 8 | Wire.read()) / 131.0;
// Corrige valeurs
GyroX = GyroX - GyroErrorX; // GyroErrorX ~(-0.56)
GyroY = GyroY - GyroErrorY; // GyroErrorY ~(2)
GyroZ = GyroZ - GyroErrorZ; // GyroErrorZ ~ (-0.8)

// Currently the raw values are in degrees per seconds, deg/s, so we need to multiply by sendonds (s) to get the angle in degrees
gyroAngleX = gyroAngleX + GyroX * elapsedTime; // deg/s * s = deg
gyroAngleY = gyroAngleY + GyroY * elapsedTime;
yaw =  yaw + GyroZ * elapsedTime;

// Filtre Complementaire - combine acceleromter and gyro angle values
roll = 0.96 * gyroAngleX + 0.04 * accAngleX;
pitch = 0.96 * gyroAngleY + 0.04 * accAngleY;


if (roll > 360) {
  roll = roll - 360;
    }
if (roll < 0) {
  roll = roll + 360;
    }


if (pitch > 360) {
  pitch = pitch - 360;
    }
if (pitch < 0) {
  pitch = pitch + 360;
    }


if (yaw > 360) {
  yaw = yaw - 360  ;
    }
if (yaw < 0) {
  yaw = yaw + 360  ;
    }



// Print the values on the serial monitor
Serial.print(AccX);
Serial.print("/");
Serial.print(AccY);
Serial.print("/");
Serial.println(AccZ);

   String TIME = String(millis(), 10);
  //  myFile.println("time : " + TIME + " ms");

   String YAW = String(yaw, 2);
   String PITCH = String(pitch, 2);
   String ROLL = String(roll, 2);
  // myFile.println("Yaw : " + YAW);
  // myFile.println("pitch : " + PITCH);
  // myFile.println("roll : " + ROLL);

   String ACCX = String(AccX, 2);
   String ACCY = String(AccY, 2);
   String ACCZ = String(AccZ, 2);

  // myFile.println("Acc_X : " + ACCX);
  // myFile.println("Acc_Y : " + ACCY);

   String ACCANGLEX = String(accAngleX, 2);
   String ACCANGLEY = String(accAngleY, 2);
```

```
          // myFile.println("Acc_Angle_X : " + ACCANGLEX);
          //  myFile.println("Acc_Angle_Y : " + ACCANGLEY);

            String gyroX = String(GyroX, 2);
            String gyroY = String(GyroY, 2);
            String gyroZ = String(GyroZ, 2);

           // myFile.println("gyro_Y : " + gyroY);
          //  myFile.println("gyro_Z : " + gyroZ);

            String GYROANGLEX = String(gyroAngleX, 2);
            String GYROANGLEY = String(gyroAngleY, 2);

          //  myFile.println("Angle_Gyro_X : " + GYROANGLEX);
          //  myFile.println("Angle_Gyro_Y : " + GYROANGLEY);

        myFile.println( TIME+ ","+YAW + "," +PITCH+ "," +ROLL+ "," +ACCX+ "," +ACCY + "," +ACCZ + "," +ACCANGLEX+ "," +ACCANGLEY+ ","
+gyroX+ "," +gyroY+ "," +gyroZ+ "," +GYROANGLEX+ "," +GYROANGLEY );
        myFile.close();



        //SD END ----------------------------------------


        }
       }

       void calculate_IMU_error() {
        // We can call this funtion in the setup section to calculate the accelerometer and gyro data error. From here we will get the error values used
in the above equations printed on the Serial Monitor.
        // Note that we should place the IMU flat in order to get the proper values, so that we then can the correct values
        // Read accelerometer values 200 times
        while (c < 200) {
          Wire.beginTransmission(MPU);
          Wire.write(0x3B);
          Wire.endTransmission(false);
          Wire.requestFrom(MPU, 6, true);
          AccX = (Wire.read() << 8 | Wire.read()) / 16384.0 ;
          AccY = (Wire.read() << 8 | Wire.read()) / 16384.0 ;
          AccZ = (Wire.read() << 8 | Wire.read()) / 16384.0 ;
          // Sum all readings
          AccErrorX = AccErrorX + ((atan((AccY) / sqrt(pow((AccX), 2) + pow((AccZ), 2))) * 180 / PI));
          AccErrorY = AccErrorY + ((atan(-1 * (AccX) / sqrt(pow((AccY), 2) + pow((AccZ), 2))) * 180 / PI));
          c++;
        }
        //Divide the sum by 200 to get the error value
        AccErrorX = AccErrorX / 200;
        AccErrorY = AccErrorY / 200;
        c = 0;
        // Read gyro values 200 times
        while (c < 200) {
          Wire.beginTransmission(MPU);
          Wire.write(0x43);
          Wire.endTransmission(false);
          Wire.requestFrom(MPU, 6, true);
          GyroX = Wire.read() << 8 | Wire.read();
          GyroY = Wire.read() << 8 | Wire.read();
          GyroZ = Wire.read() << 8 | Wire.read();
          // Sum all readings
          GyroErrorX = GyroErrorX + (GyroX / 131.0);
          GyroErrorY = GyroErrorY + (GyroY / 131.0);
          GyroErrorZ = GyroErrorZ + (GyroZ / 131.0);
          c++;
        }
        //Divide the sum by 200 to get the error value
        GyroErrorX = GyroErrorX / 200;
        GyroErrorY = GyroErrorY / 200;
        GyroErrorZ = GyroErrorZ / 200;
        // Print the error values on the Serial Monitor
        Serial.print("AccErrorX: ");
        Serial.println(AccErrorX);
        Serial.print("AccErrorY: ");
        Serial.println(AccErrorY);
        Serial.print("GyroErrorX: ");
        Serial.println(GyroErrorX);
        Serial.print("GyroErrorY: ");
        Serial.println(GyroErrorY);
        Serial.print("GyroErrorZ: ");
        Serial.println(GyroErrorZ);
       }
```

9.3 Code Arduino Sequenceur

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

const int Jack_Pin = 10;
bool countdownActive = false;
long int Temps_Passe = 0;
long int DEBUT_TEMPS;
bool fini = false;

const int Pin_SW_Para = A2;
const int Pin_Led_Para = 8;
const int Pin_SW_Pret_A_Voler = A1;
const int Pin_Led_Pret_A_Vole = 9;
const int Pin_Servo_Para = 3;

Servo servo;

void setup() {
  pinMode(Jack_Pin, INPUT_PULLUP);

  pinMode(Pin_SW_Pret_A_Voler, INPUT);
  pinMode(Pin_SW_Para, INPUT);
  pinMode(Pin_Led_Para, OUTPUT);
  pinMode(Pin_Led_Pret_A_Vole, OUTPUT);

  servo.attach(Pin_Servo_Para);

}

void loop() {

    if (  digitalRead(Jack_Pin) != LOW && digitalRead(Pin_SW_Pret_A_Voler)   == LOW ) {

    digitalWrite(Pin_Led_Pret_A_Vole, LOW);
    delay(300);
    digitalWrite(Pin_Led_Pret_A_Vole, HIGH);
    delay(300);

  }
  else {

  if (digitalRead(Jack_Pin) != LOW && countdownActive == false && digitalRead(Pin_SW_Pret_A_Voler) == HIGH   ){  // Si Jack arrache ET le
compteur pas actif

    countdownActive = true;  // Start the countdown
    DEBUT_TEMPS = millis();    // Store the start time


  }

   if (countdownActive == true) {
     Temps_Passe = millis() - DEBUT_TEMPS;  // Calculate the elapsed time
     if (Temps_Passe >= 1000 && Temps_Passe <= 2000) {
       digitalWrite(Pin_Led_Pret_A_Vole, LOW);
     }
      if (Temps_Passe >= 2000 && Temps_Passe <= 4000) {
      digitalWrite(Pin_Led_Pret_A_Vole, HIGH);
     }
      if (Temps_Passe >= 4000 && Temps_Passe <= 6000) {
      digitalWrite(Pin_Led_Pret_A_Vole, LOW);
     }
      if (Temps_Passe >= 6000 && Temps_Passe <= 8000 ){
      digitalWrite(Pin_Led_Pret_A_Vole, HIGH);
      }

      if (Temps_Passe >= 8000 && Temps_Passe <= 12000) {
      digitalWrite(Pin_Led_Pret_A_Vole, LOW);
     }

      if (Temps_Passe >= 12000 && Temps_Passe <= 14000) {
      digitalWrite(Pin_Led_Pret_A_Vole, HIGH);
     }
}
    if (Temps_Passe >= 14000 && fini == false ) {  // Countdown finished


      servo.write(0);
      digitalWrite(13, HIGH);

     }

   if (digitalRead(Pin_SW_Pret_A_Voler) == HIGH && countdownActive != true) {
     digitalWrite(Pin_Led_Pret_A_Vole, HIGH);
```

```
  }
  if  (digitalRead(Pin_SW_Pret_A_Voler) == LOW && countdownActive != true) {
    digitalWrite(Pin_Led_Pret_A_Vole, LOW);
  }

  if (digitalRead(Pin_SW_Para) == LOW && countdownActive != true ) {



    servo.write(90);
    digitalWrite(Pin_Led_Para, LOW);


  }
  if (digitalRead(Pin_SW_Para) == HIGH && countdownActive != true)
   {

    servo.write(0);
    digitalWrite(Pin_Led_Para, HIGH);


}
  }
}
```