Rapport de vol

Mini-Fusée : MF22 StellarDart

Juillet 2024

Introduction

Le projet MF22 StellarDart de l'année 2023-2024 est une initiative de mini-fusée expérimentale menée par l'association Elisa SPACE, affiliée à Elisa Aerospace, une école d'ingénieurs spécialisée dans l'aéronautique et le spatial. L'association vise à promouvoir le domaine spatial au sein de l'école en proposant divers projets tels que des fusées à eau, des micro-fusées et des mini-fusées.

Lancé en novembre 2023 par cinq membres, ce projet a été un travail constant tout au long de l'année. Notre mini-fusée a été lancée lors de la campagne de lancement du C'Space 2024. Elle est équipée de cartes électroniques programmées permettant de déployer deux parachutes, de réaliser des mesures en vol et de séparer la fusée en deux parties. Avant le lancement (*Figure 1)*, chaque fusée doit démontrer sa capacité à effectuer un vol nominal (récupération intacte de la fusée) et à embarquer des expériences d'intérêt scientifique.

Notre expérience consiste à séparer la partie haute de la fusée grâce à un système de bague rotative utilisant des billes. De plus, nous avons décidé de collecter un maximum d'informations pendant le vol en embarquant plusieurs capteurs (accéléromètre, gyromètre, thermomètre, tube de Pitot) afin d'améliorer nos simulations et de concevoir des projets optimisés.

Ce rapport détaille d'abord le contexte de notre projet pour l'année 2023-2024, puis la planification des tâches, et enfin la réalisation de la mini-fusée, en décrivant les problèmes rencontrés et les résultats obtenus.

Figure 1 : Fusée en préparation pour son lancement

Table des matières

Contexte du projet StellarDart

Encadrement du projet

Planète Sciences

Les membres du projet collaborent avec l'association Planète Sciences, qui assure chaque année le suivi des projets étudiants similaires à travers la France. Cette association est également responsable de l'organisation du C'Space, la campagne nationale de lancement des projets étudiants dans le domaine spatial, visant à encourager l'intérêt et la pratique scientifique chez les jeunes. Le C'Space n'est qu'une des nombreuses activités proposées par Planète Sciences. En effet, l'association organise aussi des événements tels que la Nuit des étoiles, les Trophées de Robotique, CanSat, Float Lift & Fly Contest. Au cours des deux dernières années, Planète Sciences a accompagné notre équipe en organisant cinq Rencontres de Clubs Espace (RCE) pour nous guider et nous fournir les informations essentielles en vue de la campagne de lancement.

Figure 2 : Logo de l'association Planète Sciences

La campagne de lancement C'Space

Afin de tester nos travaux sur StellarDart, le projet a participé à la campagne nationale de lancement C'Space, qui s'est déroulé du 6 au 13 juillet 2024 à Tarbes. Organisé par Planète Sciences en partenariat avec le CNES, le C'Space représente l'aboutissement de notre projet. StellarDart y a été lancée et a effectué un vol nominal.

Le Centre National d'Études Spatiales (CNES)

Le CNES est le coorganisateur du C'Space avec Planète Sciences. En tant que la plus importante agence spatiale de l'Union européenne en termes de financement, cet établissement public est responsable du programme spatial français. Dans le cadre de notre projet, le CNES fournit les propulseurs à poudre nécessaires au lancement des fusées étudiantes, qui restent très dangereux à manipuler.

Figure 3 : Logo du CNES

Équipe du Projet

Le projet, mené par six étudiants d'ELISA AEROSPACE au cours de l'année scolaire, a été divisé en trois domaines techniques afin de faciliter la répartition du travail. Après avoir nommé Émilie PEREZ responsable de projet, les membres de l'équipe se sont répartis les rôles comme suit :

- Programmation : Arthur DEWASTE, Baptiste SCHWENDIMANN
- Électronique : Arthur DEWASTE, Baptiste SCHWENDIMANN, Aurélie LAMERAND
- Mécanique : Émilie PEREZ, Eva DILEO, Aurélie LAMERAND

Objectifs du Projet

En plus de sa conception, nous devons veiller à ce que la mini-fusée effectue un vol nominal, c'est-à-dire un vol qui se termine par la récupération intacte de la fusée, contrairement à un vol balistique. Le vol de la fusée comprend plusieurs phases (Figure 4) :

- La propulsion : Cette phase commence à la mise à feu du propulseur lorsque la fusée est encore sur la rampe de lancement et se termine lorsque tout le carburant a été consommé.
- La phase balistique : Immédiatement après la phase de propulsion, la fusée devient un corps libre soumis uniquement à son poids et aux forces de frottement.
- La récupération : Après avoir atteint l'apogée, le parachute se déploie pour permettre un atterrissage en douceur de la mini-fusée, réduisant considérablement sa vitesse.

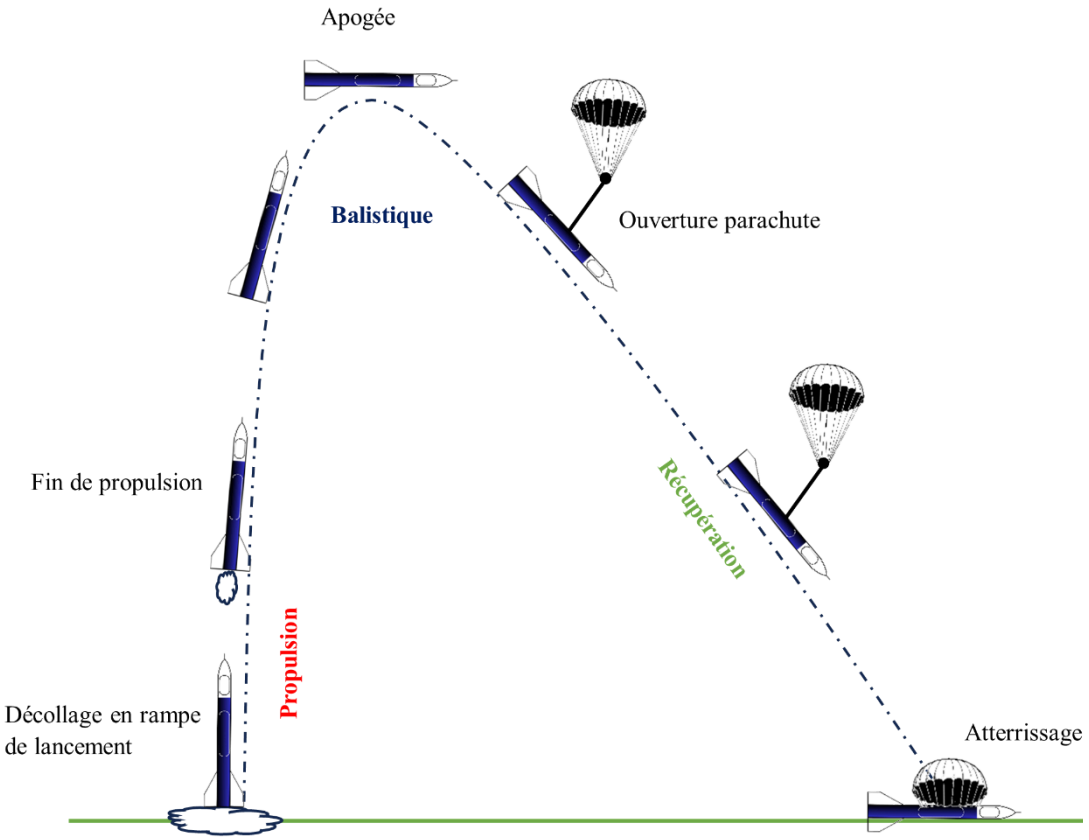


Figure 4 : Les différentes phases de vol de la mini-fusée

Expériences

Le projet StellarDart, qualifié de "mini-fusée", doit au moins permettre de réaliser une expérience à bord. Pour ce projet, plusieurs expériences ont été décidées, qui constitueront la charge utile du vol :

- Enregistrement des accélérations, de la vitesse angulaire et de la température.
- Tube de Pitot pour transmettre la vitesse
- Déploiement automatique du parachute à l'aide d'un minuteur.
- Séparation de la partie supérieure à l'aide d'un système rotatif avec des billes.
- Enregistrement des données sur une carte SD embarquée qui servira de boîte noire.
- Transmission des données en direct vers le sol à l'aide d'une télémétrie longue portée
- Système de récupération pour la partie séparée

L'idée derrière ces expériences est de prouver et valider leur fonctionnement pour les installer de manière récurrente sur les futurs projets plus évolués d'ELISA SPACE. Cela permettra d'augmenter chaque année le niveau de complexité ainsi que les performances des projets réalisés.

Cahier des Charges

Pour participer à la campagne de lancement du C'Space, la mini-fusée doit satisfaire le cahier des charges fourni par Planète Sciences, principalement axé sur les consignes de sécurité. Le site de lancement étant un camp militaire et les propulseurs étant dangereux à manipuler, tout projet doit respecter des normes très strictes. Voici le cahier des charges retenu :

Généralités

- GN1 : La fusée ne doit présenter aucun danger pour les personnes ou l'environnement. Sont interdits : les fumigènes, l'embarquement d'animaux morts ou vivants, les expériences dangereuses pour l'environnement, tout élément inflammable ou explosif, et tout dispositif pouvant modifier la stabilité de la fusée en phase ascensionnelle.
- GN2 : La fusée doit être compatible avec la rampe utilisée lors de la campagne de lancement.

Vol

- VL1 : L'équipe projet doit fournir une chronologie au responsable du lancement.
- VL2 : Le propulseur doit être centré sur le diamètre de la fusée et maintenu dans l'axe par la bague arrière. Un système doit empêcher le propulseur de sortir vers l'arrière, notamment lors du dépotage. Sa fixation doit pouvoir être effectuée rapidement, quelques minutes avant le lancement.
- VL3 : L'axe longitudinal de chaque aileron doit être parallèle à celui de la fusée et réparti symétriquement autour de son corps. Les ailerons doivent supporter les fortes contraintes aérodynamiques du vol.
- VL4 : La fusée doit avoir un vol stable. Si sa géométrie ou sa masse est modifiée durant la phase ascensionnelle, sa stabilité doit être vérifiée avant et après cette modification.
- VL5 : La flèche statique de la fusée doit être inférieure à 1% (entre l'extrémité supérieure de la fixation des ailerons et le bas de l'ogive).
- VL6 : Tous les éléments de la fusée doivent rester solidement fixés durant toute la durée du vol.

Récupération

- RC1 : La fusée doit être équipée d'un système de récupération permettant de rejoindre le sol à une vitesse verticale comprise entre 5 et 15 m/s.
- RC2 : La fusée doit disposer d'un système réglable en temps pour déployer un ralentisseur à la culmination, déclenché à ±2 secondes de l'apogée.

- RC3 : Le ralentisseur et ses fixations doivent résister au choc lors de l'ouverture.
- RC4 : Si une trappe latérale est utilisée, elle ne doit pas s'ouvrir sans être commandée mais doit pouvoir s'ouvrir malgré les contraintes du vol.
- RC5 : La portée balistique de la fusée doit être inférieure à 200 m lorsqu'elle est lancée à 80°.

Électronique

- EL1 : La fusée doit être équipée d'un système mécanique ou électronique pour la mise en œuvre du système de récupération.
- EL2 : Tous les éléments de commande doivent être accessibles lorsque la fusée est sur la rampe.
- EL3 : L'autonomie de l'alimentation électrique doit être d'au moins 15 minutes, avec un interrupteur marche/arrêt facilement accessible.
- EL4 : La fusée doit disposer d'indicateurs clairs pour connaître son état à tout moment.
- EL5 : Les interrupteurs doivent être positionnés de sorte que leur basculement s'opère perpendiculairement à l'axe de la fusée. Si cela n'est pas possible, la position ON ou VOL d'un interrupteur doit être dirigée vers le propulseur de la fusée.
- EL6 : Si une expérience autre que le test d'un système de récupération ou en lien avec celui-ci est intégrée à la fusée, elle doit respecter le cahier des charges de la fusée expérimentale.
-

Planification des Tâches

Gestion de Projet

La gestion de projet doit être un effort continu tout au long de sa durée. La répartition de notre équipe en différents pôles entraîne une perte de visibilité sur l'avancement global du projet. De plus, l'interdépendance des pôles signifie que tout retard dans un domaine peut entraîner des retards cumulés dans les autres. Ainsi, dès le début, nous avons soigneusement identifié les tâches prioritaires. Le déroulement du projet a été programmé, et les objectifs à court et long termes ont été fixés.

Rencontres Club Espace

Le projet a participé aux trois Rencontres Club Espace (RCE) de 2023-2024 organisées par Planète Sciences :

- Première RCE : Cette rencontre a permis une prise de contact entre le projet et les bénévoles de Planète Sciences.
- Deuxième RCE : Elle a servi de revue pour la validation du projet.
- Troisième RCE : Cette rencontre a permis la préqualification, validant la stabilité du projet et testant le système de récupération.

StabTraj et la Propulsion

Stabilité de la Mini-Fusée

La stabilité de la mini-fusée durant le vol, notamment lors du décollage, est calculée à l'aide d'une feuille Excel nommée StabTraj, fournie par le CNES. En entrant les dimensions de la fusée et les coordonnées de son centre de masse dans cette feuille de calcul, on peut confirmer si la fusée est stable ou non. La stabilité peut être contrôlée par la coiffe, les ailerons et le centre de masse, et ajustée en modifiant les dimensions des quatre ailerons. Des captures d'écran de StabTraj sont disponibles en Annexe 1.

Propulsion

La propulsion de la fusée est assurée par un propulseur à poudre de type Pandora, fourni par le CNES pour la campagne du C'Space. Ses dimensions sont normalisées par le CNES :

- Masse totale : 0,1599 kg
- Longueur totale : 228 mm
- Diamètre nominal : 24 mm
- Impulsion totale : 142,4 N.s

En termes de performance, la courbe de poussée du propulseur Pandora (Figure 5) est disponible sur le site du CNES.

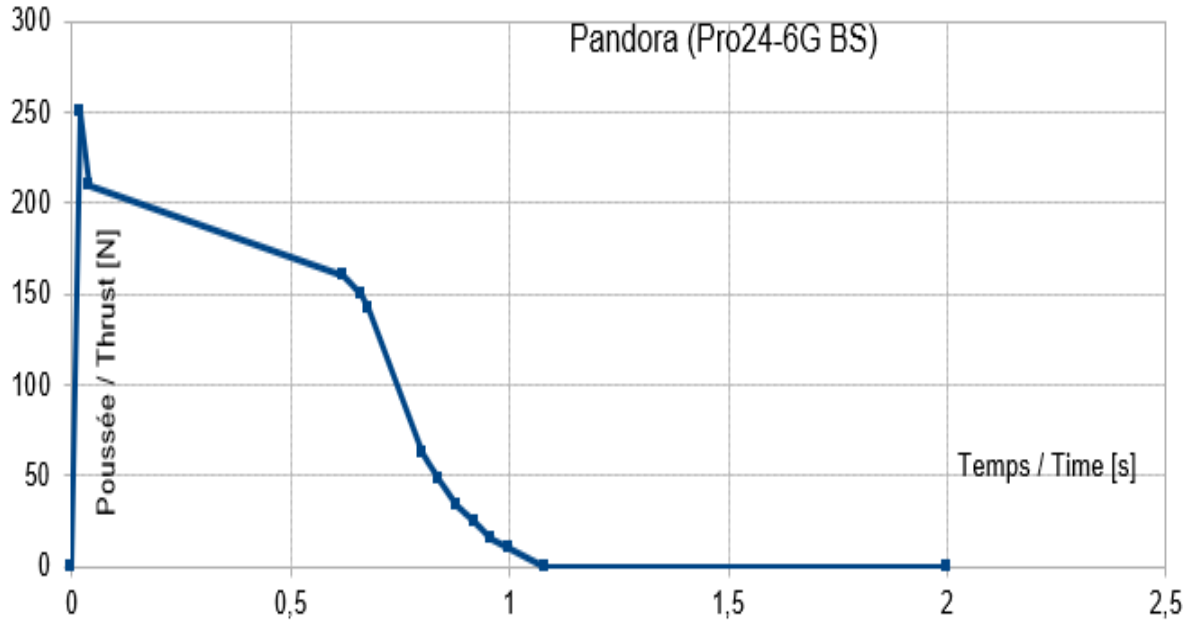


Figure 5 : Courbe de poussée du propulseur Pandora

La courbe indique une poussée maximale de 250 N et une poussée moyenne d'environ 100 N pendant plus d'une seconde, ce qui représente un bon compromis pour un propulseur de cette taille.

Réalisation du projet

Conception de l'électronique principal

Pour déclencher le lancement du parachute et assurer diverses fonctions de la mini-fusée, telles que communiquer avec le sol et effectuer des mesures en vol, il a été nécessaire de concevoir et de réaliser des cartes électroniques. Lorsque nous avons commencé ce projet, notre équipe avait une première expérience en Arduino, et en électronique pratique grâce à une mini-fusée réalisé durant l'année 2021-2022, nommée Yoctonium.

Nous avons opté pour un AtTiny en raison de ses performances adaptées à notre cahier des charges. De plus, sa petite taille est un atout majeur compte tenu de l'espace limité à bord de la fusée. En février 2024, nous avons créé une ébauche des premiers programmes. La figure 6 montre le câblage complet réalisé sur Fritzing.

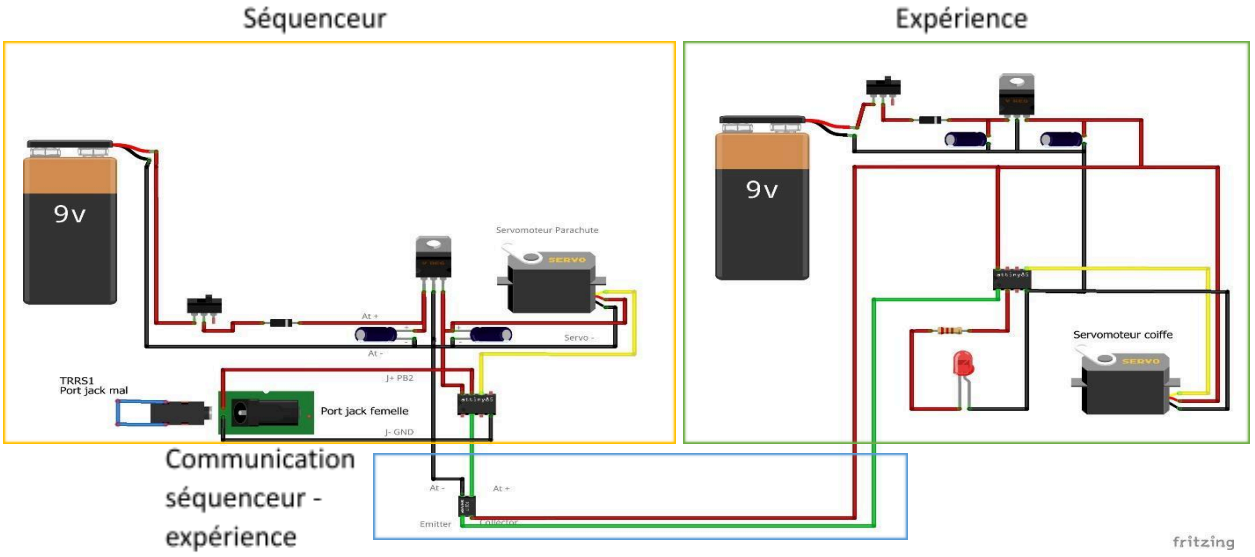


Figure 6 : Câblage principal du corps.

Le montage comporte deux parties principales :

- Partie Séquenceur : Cette carte détecte le départ de la fusée et déclenche l'ouverture du parachute.
- Partie Expérience : Elle active un servo-moteur au niveau de l'apogée permettant de séparer la partie haute de la fusée.

Il y a également un électronique secondaire qui est l'électronique de la coiffe. Il sera développer dans la suite de ce rapport.

Prise Jack de Déclenchement

Le signal de décollage est généré à l'aide d'une prise jack . Le côté femelle est fixé à la fusée, tandis que le côté mâle est connecté à la rampe de lancement. Lorsque la fusée décolle, la rupture du contact électrique est enregistrée par les deux microcontrôleurs séquenceurs, qui commencent alors leur comptage. Ce signal de déclenchement est réceptionné par le microcontrôleur principal. Le système jack de déclenchement est situé en bas de la case électronique.

En intégrant ces composants et systèmes, nous avons assuré une gestion précise et fiable de la sécurité du vol, garantissant que le parachute se déploie au moment opportun, même en cas de dysfonctionnement d'autres parties de l'électronique.

Partie Séquenceur

Le but de cette carte est de gérer l'enchaînement de toutes les phases du vol, en respectant les normes de sécurité imposées par le cahier des charges de Planète Sciences.

Le microcontrôleur joue un rôle crucial en tant que séquenceur. Il doit chronométrer depuis le décollage et déclencher l'ouverture du parachute au moment approprié. Pour des raisons de sécurité, une fenêtre temporelle est définie à partir de l'instant théorique de l'apogée, en dehors de laquelle le parachute ne pourra pas s'ouvrir. Son code est présenté en Annexe 2.

Partie Expérience

Selon le cahier des charges, le séquenceur parachute doit être isolé électriquement du reste de la fusée. Cette isolation est nécessaire pour garantir que le parachute se déploie même en cas de problème avec le reste de l'électronique. Pour réaliser cette isolation, nous utilisons des optocoupleurs ou photocoupleurs (Figure 7). Ces composants électroniques permettent de transmettre une tension entre l'entrée et la sortie sans contact électrique direct.

Figure 7 : Optocoupleur

Un optocoupleur se compose d'une diode électroluminescente (LED) qui émet de la lumière infrarouge, et d'un phototransistor qui ne laisse passer du courant que lorsqu'il est éclairé par la LED. Ce mécanisme permet de transmettre un signal sans aucune connexion électrique directe.

Nous avons intégré un optocoupleur pour la communication série entre la carte séquenceur et la carte parachute. Ce composant permet d'envoyer un signal une fois le temps de l'apogée atteinte pour faire tourner le servo-moteur permettant la séparation de la partie supérieure de la fusée.

Conception de l'électronique secondaire

Carte expérience coiffe

Le coeur du PCB expérience de la coiffe est une carte LILYGO TTGO LoRa32, basée sur un ESP32, cette carte offre donc toute la puissance d'un ESP32 en incorporant également un module LoRa (Long Range) RFM95 pour la communication à longue distance. La télémesure est un aspect important de l'expérience menée à bord de la fusée, puisque nous voulons suivre la trajectoire de la fusée en temps réel. Pour gérer la communication à longue portée, une carte émettrice (Figure 8) dont l'objectif est d'envoyer toutes les données et une carte réceptrice positionnée au sol doivent toutes les deux être programmées. Ces deux cartes sont des LILYGO. En cas de dysfonctionnement du système de télécommunication, la carte LILYGO est dotée d'un lecteur de carte micro SD, ceci nous permet d'enregistrer et de stocker les données de la fusée sur une carte SD récupérable après le vol.

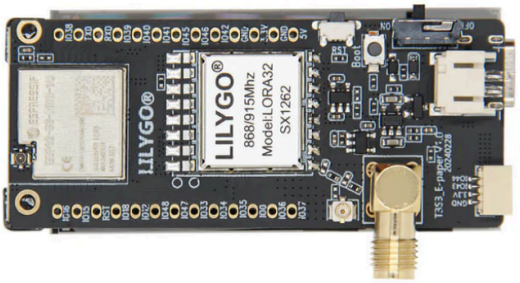


Figure 8 : Carte LILYGO

Lors de ce vol, les modules RFM95 des cartes LILYGO utilisent une fréquence de 868MHz.
Les cartes sont programmées en Arduino. Le programme de la carte LILYGO de la coiffe doit donc gérer la réalisation des différentes expériences, l'envoi des paquets de données au module RFM95, ainsi que le stockage des données sur la carte SD.

Carte de Télémétrie Réceptrice

La carte réceptrice reste au sol, connectée à un ordinateur portable et couplée à un logiciel adapté. Les données sont récupérées via une connexion série (Serial) à l'aide de ce logiciel. Son objectif est de capter toutes les données transmises par la carte de télémesure émettrice et de les transmettre à l'ordinateur.

Pour fonctionner correctement, la télémesure doit être configurée sur la même fréquence que l'émetteur. La carte réceptrice convertit les données en caractères, qui sont ensuite transmis au module. Ce dernier reçoit, stocke et affiche les transmissions sous forme de texte.

Le Gyroscope

Le MPU-6050 (Figure 9), présent sur la carte expérience de la coiffe, intègre à la fois un gyroscope à trois axes et un accéléromètre, permettant de réaliser des mesures indépendantes mais alignées sur les mêmes axes. Cette configuration élimine les erreurs transversales qui peuvent survenir lors de l'utilisation de dispositifs séparés.

Dans notre projet, nous avons utilisé le MPU-6050, qui offre une plage de mesure allant jusqu'à 16G d'accélération dans toutes les directions.

Figure 9 : MPU-6050

La communication entre la carte LILYGO et le MPU-6050 se fait via le protocole I2C. Ce protocole utilise les ports SCL (Serial Clock Line) et SDA (Serial Data Line) de la carte.

À chaque instant, les registres fournissent les informations suivantes :

- Vitesses angulaires

- Accélérations

Les données reçues sont des entiers sur 16 bits, qu'il faut convertir en m/s² pour les accélérations et en °/s pour les vitesses angulaires. Le facteur de conversion est indiqué dans la datasheet du MPU-6050 et est exprimé en LSB/unité. Ces valeurs varient en fonction des réglages de sensibilité du gyroscope et de l'accéléromètre, que l'on a configurés dans les registres.

La Carte SD

La carte SD utilisée pour le projet est un système de stockage de données flash, basée sur de la mémoire NAND. Cette technologie offre l'avantage d'être très rapide en écriture, ce qui est crucial pour enregistrer les données en temps réel pendant le vol.

Nous avons intégré une carte SD dans notre système sur la carte LILYGO. Dès l'allumage de la carte, les données enregistrées doivent inclure l'état de la fusée, à savoir :

Stockage des données :

- valeur lissée du différentiel de pression
- valeur de vitesse air calculée
- accélération selon x
- accélération selon y
- accélération selon z
- gyro selon x
- gyro selon y
- gyro selon z
- température

Conception et Fabrication des PCB

Un circuit imprimé (PCB) est constitué de plusieurs couches de cuivre isolées par un matériau non conducteur. Les composants électroniques sont intégrés à cette structure mécanique, créant ainsi un circuit électrique. Initialement, les circuits électroniques étaient montés sur une perfboard. Cependant, les défis liés à la soudure et le besoin d'un système plus flexible nous ont conduit à opter pour une autre solution.

Nous avons donc conçu le circuit sur Kikad, comme illustré par les Figures 10 et 11. Une fois le design finalisé, nous avons exporté le dessin du circuit et les détails des perçages dans un fichier Gerber. L'usinage a ensuite été préparé avec JLC PCB. Pour améliorer la facilité de montage et de démontage des composants, nous avons utilisé des connecteur JST et des tulipes.

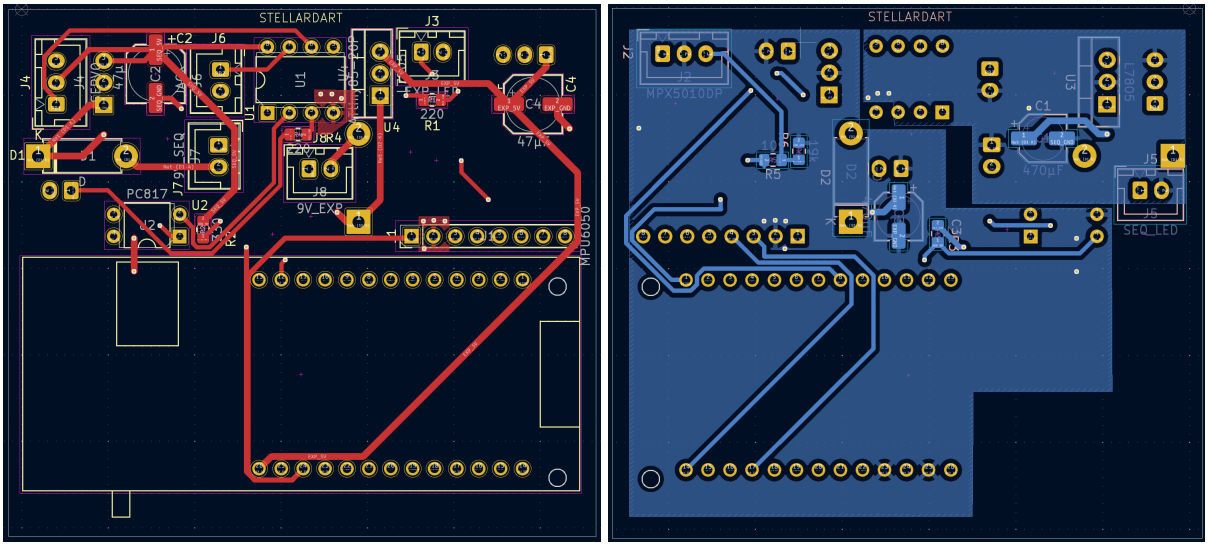


Figure 10 : Modélisation du PCB sur KIKAD (coiffe)

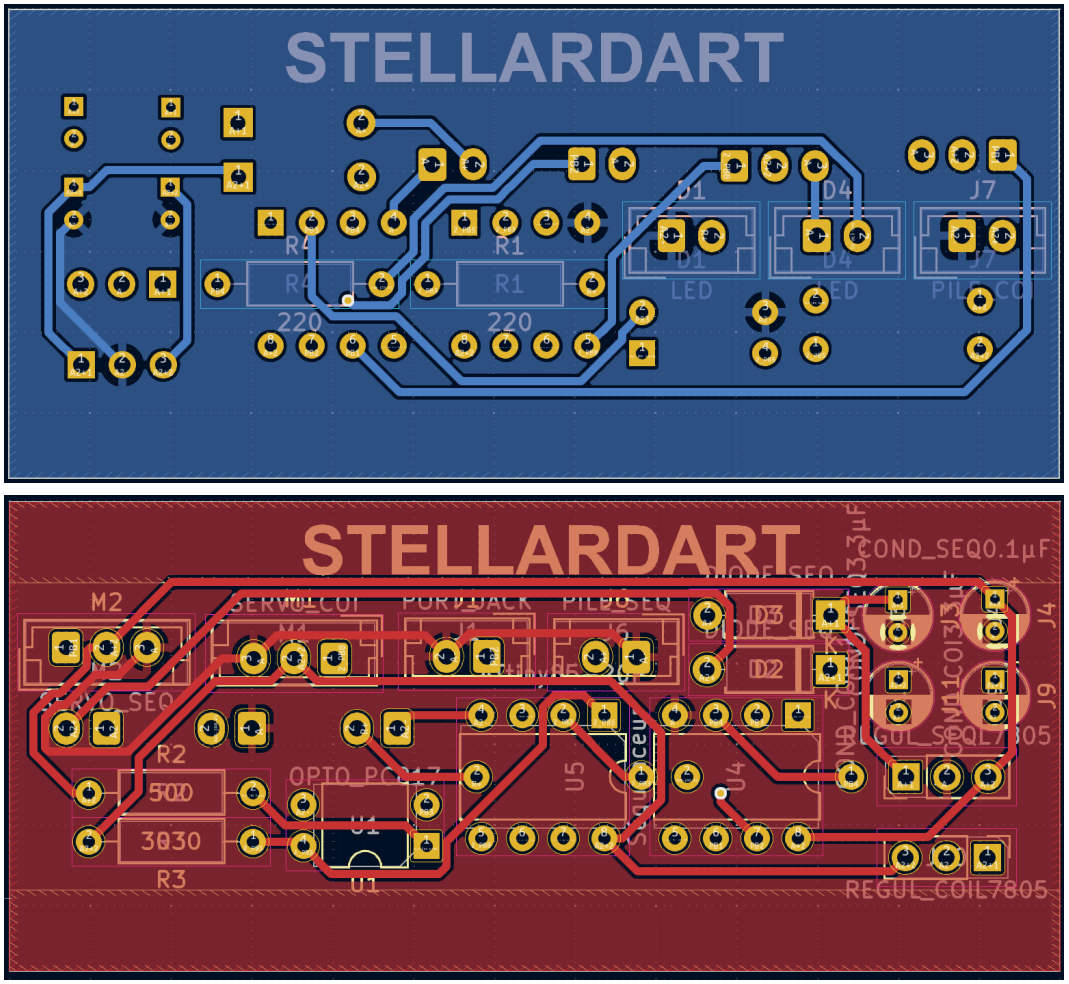


Figure 11 : Modélisation du PCB sur KIKAD (corps principal)

Gestion de la Sécurité du Vol

Système de Déclenchement du Parachute

L'électronique embarquée est responsable du déclenchement opportun de l'ouverture du parachute. Cette fonction est contrôlée par une carte située en bas de la case électronique.

Commande des Servomoteurs

Le déploiement du parachute est assuré par un servomoteur situé dans la case parachute. Ce servomoteur ouvre la porte de la case contenant le parachute. Le signal de contrôle des servomoteurs est un signal PWM (Pulse Width Modulation), géré par les microcontrôleurs.

Nous avons opté pour l'utilisation de servomoteurs plutôt que de moteurs à courant continu. En effet, avec un servomoteur, le signal de commande indique immédiatement l'angle que le servomoteur doit atteindre. À l'inverse, pour obtenir le même résultat avec des moteurs à courant continu, il aurait fallu calculer le temps nécessaire pour que le moteur effectue la rotation voulue et l'alimenter pendant cette durée. En cas de blocage, la rotation risquerait de s'arrêter avant d'atteindre l'angle désiré. En revanche, le servomoteur continue de travailler jusqu'à ce que l'angle correspondant à la commande soit atteint.

Partie Mécanique

La conception de la structure de la fusée a été pensée pour répondre à deux objectifs principaux : intégrer les expériences sélectionnées et se conformer aux exigences de sécurité du cahier des charges de Planète Sciences. La conception et la fabrication de la structure sont détaillées ci-dessous. Les différentes parties de la fusée ont été modélisées à l'aide de CATIA et FUSION. La construction proprement dite a commencé tardivement, en mai 2024, en raison de la priorité accordée à l'électronique avant la réalisation des autres composants. L'impression 3D a été envisagée pour la fabrication des pièces, car elle offre une solidité accrue et une facilité d'assemblage.

Structure Générale

La fusée est composée de cinq sous-modules (Figure 12) :

- Le Module Propulseur (au niveau des ailerons)
- Le Module Parachute
- Le Module Cartes
- La Coiffe

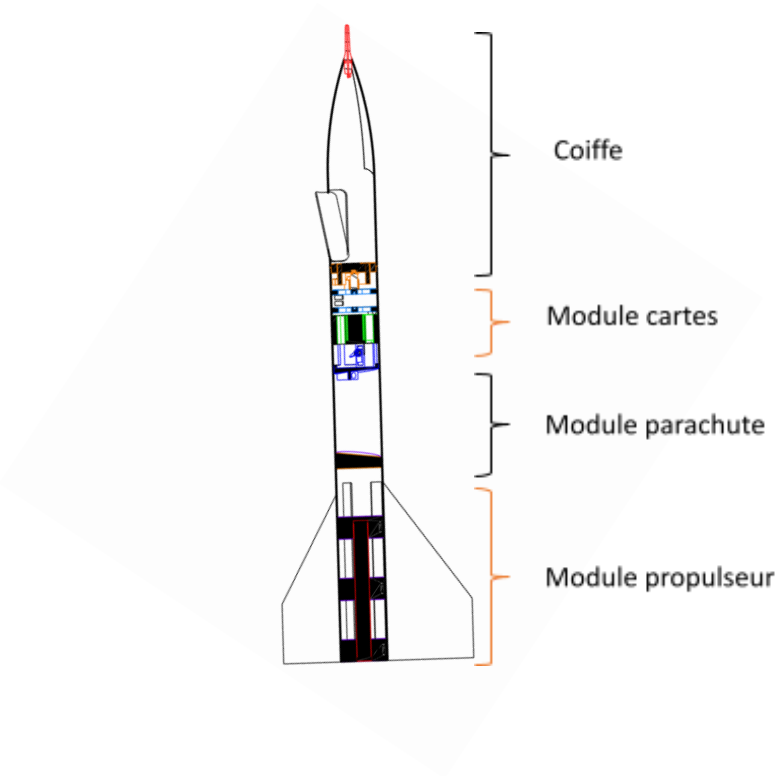


Figure 12 : Plan de la fusée

Module Propulseur

Le module propulseur a nécessité plusieurs impressions pour assurer une compatibilité optimale avec les ailerons. Il a fallu également retravailler des pièces en bois permettant le maintien du moteur. Son intégration dans la mini-fusée se fait à l'aide de :

- Une bague de poussée en bois, qui maintient la base du moteur et les ailerons à l'arrière.
- Une bagues de centrage fixe sur le centre des ailerons.
- Trois bagues de rétention en PLA permettant le maintien des ailerons

Ailerons

Les aileron (Figure 13) jouent un rôle crucial dans la stabilisation de la fusée pendant son vol. Leur fonction principale est d'empêcher la fusée de tourner sur elle-même, ce qui pourrait compromettre ses performances et entraîner une déviation de la trajectoire, avec des conséquences potentiellement dangereuses.

Pour garantir une trajectoire stable, les ailerons doivent être correctement dimensionnés. Des ailerons bien conçus assurent que la fusée revient à sa position d'équilibre en cas de déviation. À l'inverse, deux types de comportements peuvent survenir si les ailerons sont mal dimensionnés :

- Fusée instable : Si les ailerons sont trop petits, la fusée peut commencer à tourner de manière incontrôlée, réalisant des « loopings » avant de s'écraser.
- Fusée sur-stable : Si les ailerons sont trop grands, la fusée pourra osciller continuellement autour de sa position d'équilibre.

Les ailerons ont été fabriqués en plexiglass. Après leur réalisation, nous les accroché au bag en PLA.

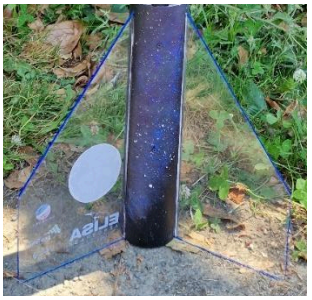


Figure 13 : Ailerons de la fusée

Le corps de la fusée

Le corps de la fusée constitue la plus grande surface en contact avec l'extérieur et joue un rôle double : il protège les éléments internes de la fusée et sert de squelette externe.

Le tube principal, fabriqué en PVC léger, présente des dimensions de 80 mm d'épaisseur extérieure et 76 mm d'épaisseur intérieure. Cette épaisseur garantit une solidité suffisante tout en minimisant le poids. L'usinage a été réalisé manuellement à l'aide d'une Dremel, complété par des opérations de lime, nécessitant une grande précision et patience.

Des trous ont été percés dans le tube pour permettre l'assemblage et la fixation des composants internes. Nous avons opté pour des vis M3, choisies pour leur bon compromis entre taille et résistance. Pour une finition propre, des chanfreins ont été utilisés pour dissimuler les vis à l'intérieur du corps de la fusée.

L'épaisseur du tube offre l'avantage de garantir une rigidité optimale, éliminant tout problème de flexion durant le vol.

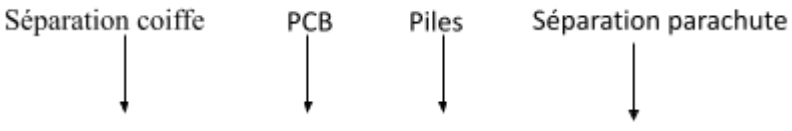
Le module parachute

Le module parachute est un élément crucial pour la récupération de la fusée. Après 1.5 seconde de l'apogée (fixée à 141 mètres selon StabTraj), le parachute est éjecté pour assurer une descente contrôlée. Une fois déployé, il permet de ralentir la fusée à une vitesse de descente d'environ 12 m/s jusqu'à l'atterrissage. Nous avons opté pour un parachute rond, fabriqué en nylon, un tissu réputé pour sa haute résistance au choc. Pour éviter que les câbles ne s'entrelacent, des bagues imprimées en 3D coulissent le long des cordes du parachute.

Le module cartes

Le module cartes (Figure 14) est la partie centrale de la fusée, car il porte la partie principal de l'électronique. Cette partie permet de déclencher deux servo-moteur : l'un pour l'ouverture du parachute et l'autre pour l'éjection de la coiffe.

Pour des raisons de simplicité et d'efficacité, la carte électronique a été placée horizontalement à l'intérieur de la fusée. La fixation de cette carte a été conçue pour être à la fois robuste (capable de supporter les fortes accélérations et les vibrations du vol) et pratique (permettant un montage et un démontage rapides pour d'éventuelles interventions ou reprogrammations).



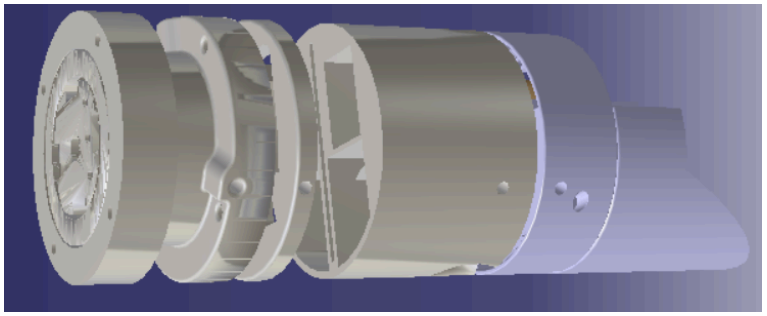


Figure 14 : CAO module cartes

Le module séparation coiffe

Le module de séparation (Figure 15) de la coiffe a été conçu par SCHWENNDIMANN Baptiste. Il se compose de trois parties : un rotor permettant de bloquer des billes, et deux pièces statiques, l'une pouvant loger les billes en position ouverte et l'autre en position fermée. L'avantage de ce système est la diminution des forces de frottement grâce à l'utilisation des billes.

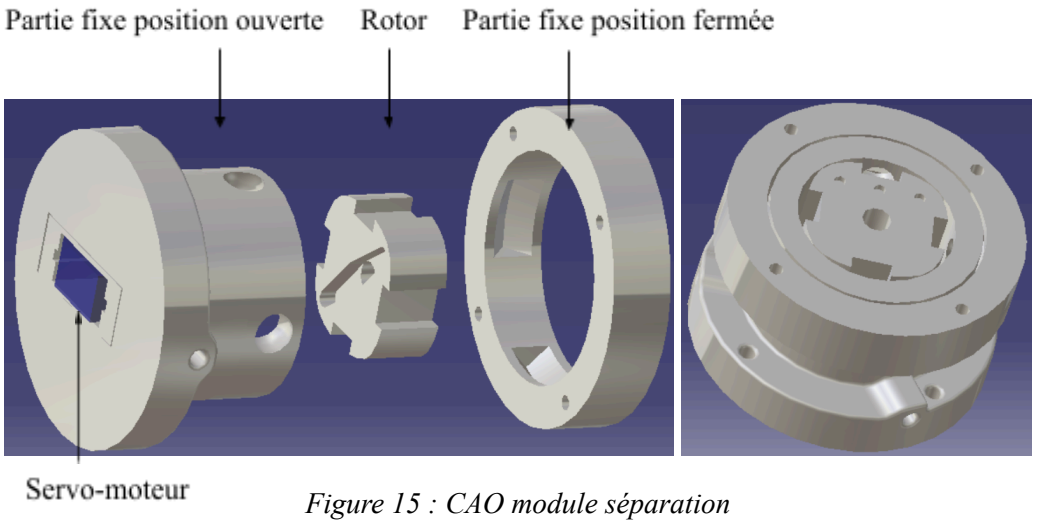


Figure 15 : CAO module séparation

La partie supérieur (partie fixe position fermée) correspond en réalité à la partie basse de la coiffe. Cette pièce est directement implémenté dans la coiffe contrairement à la représentation ci-dessus, il est possible de voir cette pièce sur la figure 16.

Le module coiffe

La coiffe de la fusée, de forme ogivale, a été réalisée en PLA . La coiffe comporte deux parties distinctes. La première est la case parachute et la seconde la case électronique (Figure 16).

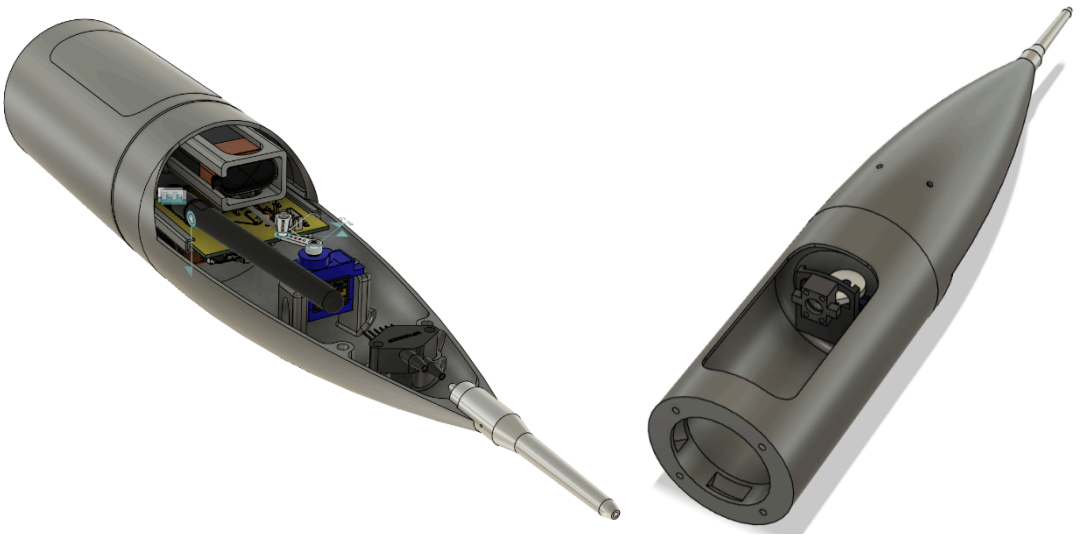


Figure 16 : CAO de la coiffe partie électronique et partie parachute

Cette partie du projet englobe toute l'électronique nécessaire à la réalisation des expériences embarquées. À l'extrémité de la coiffe, on trouve le tube de Pitot, qui est utilisé pour mesurer la vitesse de la fusée par un différentiel de pression.

Le module électronique comprend les composants suivants :

- Deux piles de 9V : pour l'alimentation électrique des divers dispositifs embarqués.
- la carte expérience coiffe : avec la carte LILYGO, ainsi que le MPU6050.
- Un servomoteur : utilisé pour l'ouverture de la trappe du parachute.
- Un capteur de pression MPX5010DP : qui récupère les données du tube de Pitot pour évaluer la vitesse de la fusée.

Cette configuration permet de collecter et de transmettre toutes les données nécessaires pour l'analyse du vol, tout en assurant le bon fonctionnement des mécanismes de récupération.

La peinture

La fusée a été baptisée StellarDart en hommage à la mission DART (Double Asteroid Redirection Test), un projet novateur visant à expérimenter des techniques de déviation d'astéroïdes. Ce nom reflète notre ambition de repousser les frontières de l'exploration spatiale et d'intégrer des technologies de pointe, à l'image de la mission DART qui cherche à démontrer des méthodes avancées pour protéger la Terre. La fusée StellarDart arbore un motif inspiré d'un ciel étoilé, en lien avec la partie « Stellar » de son nom. Cette décoration céleste symbolise notre désir d'explorer les confins de l'espace et de faire écho aux missions spatiales ambitieuses comme DART. De plus, la coiffe blanche représente l'étoile visée par la flèche, renforçant ainsi le thème de notre projet (Figure 17).

Figure 17 : Aperçu de la fusée

Campagne de lancement

Seulement trois membres de l'équipe (SCHWENDIMANN Baptiste, DEWASTE Arthur et PEREZ Emilie) ont été invités à venir qualifier et lancer le projet.

Le jour J

Pour obtenir une autorisation de vol, la mini-fusée doit passer une série de tests répertoriés dans un cahier des charges rédigé par les bénévoles de Planète Sciences. Ces tests, effectués en plusieurs étapes, doivent démontrer, entre autres, la stabilité de la fusée avec sa géométrie finale, le bon fonctionnement du séquenceur contrôlant l'ouverture du parachute, et la cohérence de la chronologie de vol spécifique à notre projet. L'intégralité de la fusée a été inspectée le soir du 9 juillet.

Déroulement du vol

La mini-fusée a décollé de la rampe de lancement le 10 juillet à 15h30. Après une ascension d'environ 10 secondes, la séparation s'est déclenchée à l'apogée, puis l'ouverture des parachutes ont eu lieu 1.5 seconde après la séparation, atteignant une altitude de 110 mètres au-dessus du sol. La descente s'est effectuée à une vitesse constante de 32 km/h, comme prévu. La mini-fusée s'est posée sans encombre sur l'herbe après 40 secondes de vol nominal. À l'arrivée, notre fusée était intacte. Les caméras et les données ont indiqué une ouverture du parachute au meilleur moment, et la détection de l'apogée a parfaitement fonctionné ! La fusée était censée subir une accélération de 10 G au décollage, mais n'a pris que 9 G en raison de la rampe qui l'a fortement ralentie.

Analyse des Résultats

Vitesse Maximale

La vitesse maximale enregistrée durant le vol est de 114.53 km/h (soit 31.81 m/s), ce qui est inférieur à la vitesse estimée de 183 km/h. Cet écart pourrait être dû à plusieurs facteurs, notamment les conditions météorologiques au moment du vol, ainsi qu'une éventuelle réduction de vitesse au décollage causée par les frottements avec la rampe de lancement. Il est également possible que cet écart soit lié au paramétrage des capteurs. La vitesse a été mesurée à l'aide d'un tube de Pitot, dont l'étalonnage a initialement été effectué en soufflerie. Cependant, comme la vitesse en soufflerie était inférieure à 100 km/h, nous avons également réalisé des étalonnages en voiture pour atteindre des vitesses allant jusqu'à 130 km/h. La différence entre la vitesse supposée et celle mesurée pourrait indiquer que la mesure n'a pas été parfaitement précise. Les données de vitesse obtenues ont également servi au calcul de l'altitude.

Altitude Maximale

L'altitude maximale atteinte est de 110.10 mètres, déterminée en intégrant les données de vitesse au fil du temps pour obtenir une estimation cumulative. Cette altitude est inférieure à l'altitude estimée de 140 mètres, ce qui est attendu étant donné que la vitesse mesurée est également inférieure à celle estimée. En l'absence d'altimètre, de baromètre ou d'autres capteurs dédiés à la mesure de l'altitude, nous n'avons pas de données directes pour valider cette estimation. Cette altitude à pu être calculé à partie de la durée totale du vol. Celle-ci est de 21.24 secondes. Ce temps est calculé en prenant en compte les périodes durant lesquelles la vitesse de l'air est non nulle.

Graphiques de Données

Les graphiques générés fournissent une vue visuelle détaillée des différentes variables mesurées au cours du vol dans la coiffe. Voici une synthèse des principaux graphiques (Figure 18, 19 et 21) :

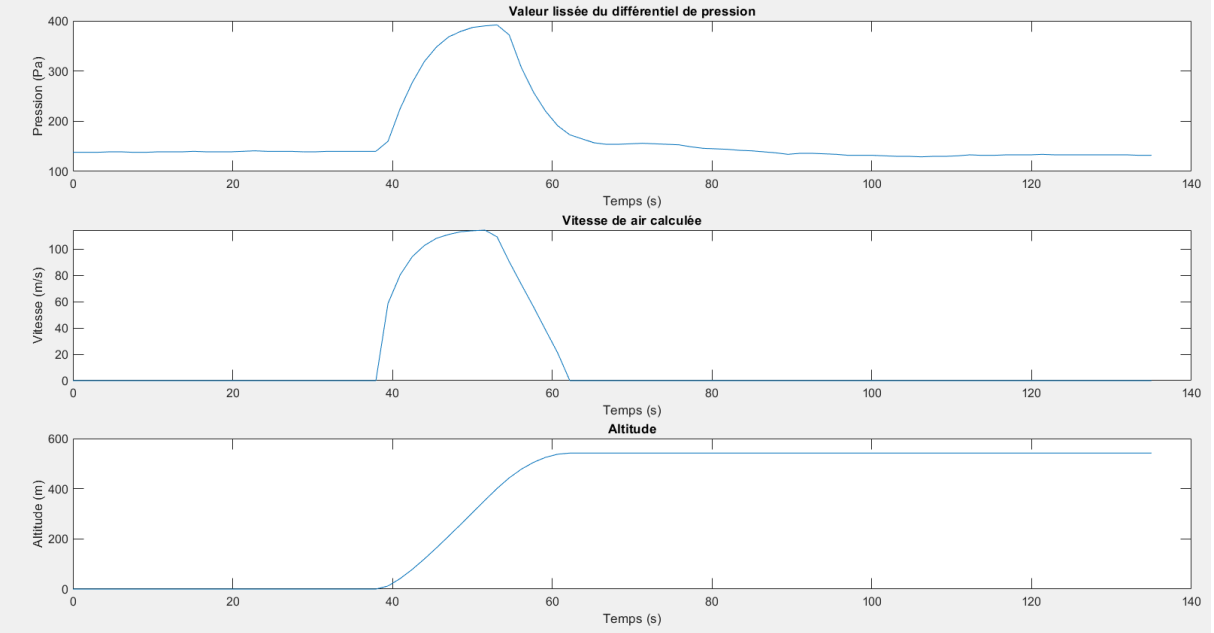


Figure 18 : Graphique du différentiel de pression, vitesse de l'air et de l'altitude

- Différentiel de Pression : Ce graphique montre comment la pression a évolué au fil du temps. Les variations peuvent indiquer des changements dans l'environnement de vol ou dans les conditions de fonctionnement de la fusée.
- Vitesse de l'Air : Le graphique de la vitesse illustre les variations de la vitesse de la fusée, montrant comment elle a changé au cours du vol. Les pics et les vallées dans ce graphique peuvent refléter des phases de montée, de stabilisation ou de descente.
- Altitude : Ce graphique présente l'altitude en fonction du temps, offrant une vue claire de la montée et de la descente de la fusée. Les changements dans l'altitude permettent de vérifier la précision du calcul d'altitude et la performance de la fusée.

On observe sur le graphique de l'altitude que cette dernière se stabilise à son maximum. Ce phénomène est dû à l'utilisation de la fonction « cumtrapz » dans MATLAB pour intégrer la vitesse et calculer l'altitude. Cette méthode d'intégration cumulative peut entraîner une stabilisation artificielle de l'altitude lorsqu'il n'y a pas de variations significatives dans les données de vitesse, ce qui peut donner l'impression que l'altitude atteint un plateau au lieu de continuer à varier et dans ce cas devenir négative (chute).

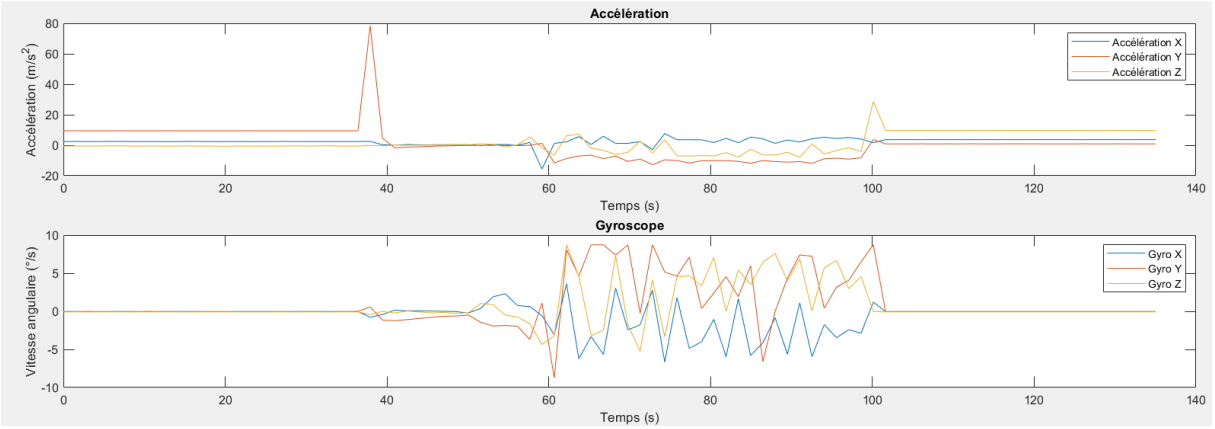


Figure 19 : Graphique du accélération et gyroscope 3 axes.

- Accélérations : Les graphiques des accélérations dans les directions X, Y, et Z montrent les forces auxquelles la fusée a été soumise. Cela peut aider à comprendre les forces d'accélération vécues pendant le vol.
- Gyroscope : Les variations des vitesses angulaires autour des axes X, Y, et Z fournissent des informations sur la stabilité et la dynamique de la fusée pendant le vol.

Le graphique du gyroscope (Figure 18) révèle les événements clés du vol, notamment le moment de la séparation de la coiffe. On observe un certain nombre de mouvements sur les différents axes, attribués à l'éjection par les ressorts. De plus, lorsque les graphiques montrent une agitation significative, cela correspond au déploiement du parachute. Ces variations indiquent des perturbations importantes dans la dynamique de la fusée à ces moments précis.

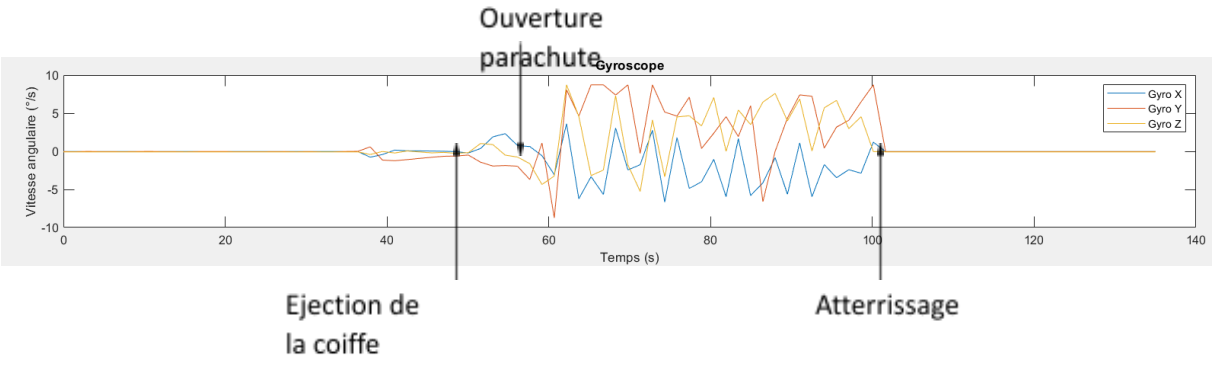


Figure 20 : Graphique gyroscope 3 axes

- Température : Le graphique de la température permet d'observer les variations thermiques au cours du vol, ce qui peut être utile pour analyser l'impact des conditions environnementales sur les performances de la fusée.

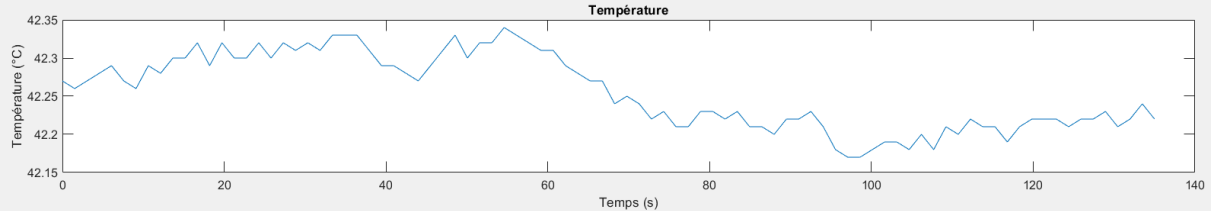


Figure 21 : Graphique température

On constate que la température est relativement élevée au départ, atteignant 43°C. Cette augmentation importante de la température peut être attribuée au fait que la fusée était exposée au plein soleil pendant son placement en rampe, ce qui a considérablement réchauffé l'intérieur de la coiffe. Avant la mise en rampe, la température intérieure était d'environ 36°C. Au cours du vol, la température a légèrement diminué, probablement en raison du refroidissement apporté par le vent généré. Enfin, lors de l'atterrissage dans l'herbe haute, légèrement plus fraîche, une diminution supplémentaire de la température a été observée.

Atterrissage et récupération

La fusée a atterri après 22 secondes de vol, durant lesquelles le vol s'est déroulé de manière nominale. Nous avons été rassurés de constater que la fusée a pu ouvrir son parachute et réaliser toutes ses expériences avec succès. De plus, les données de télémesure étaient identiques à celles stockées sur la carte SD, confirmant la précision du système de collecte de données.

Cependant, nous avons rencontré plusieurs difficultés sur le terrain, notamment lors des vols simulés, où les trappes (coiffe et corps) ne s'ouvraient pas dans les délais requis. Nous avons découvert des erreurs dans le code ainsi que des problèmes mécaniques qui ont entravé l'ouverture correcte des trappes. Ces ajustements ont été essentiels pour assurer le bon déroulement des expériences et la réussite du vol.

La récupération (Figure 22) de la fusée a eu lieu en fin d'après-midi, vers 18 heures. Nous avons été agréablement surpris de constater que la fusée était encore pleinement fonctionnelle et presque intacte, malgré le choc à l'atterrissage. De plus les deux parties avait atterrit relativement proche (1 mètre).

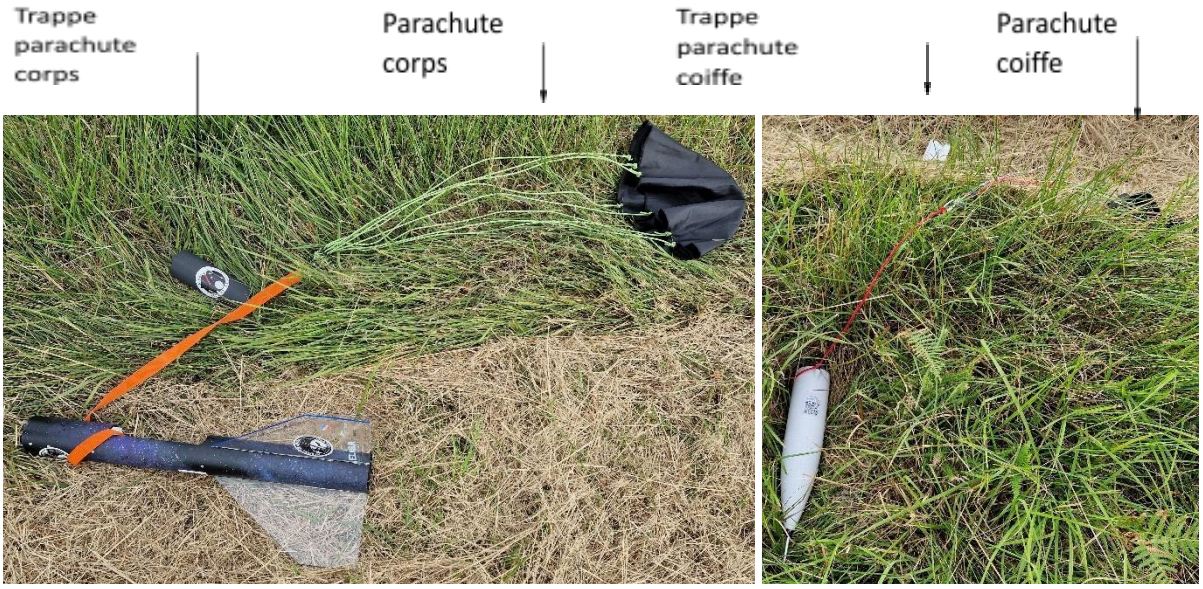


Figure 22 : Photo de la récupération

Difficultés rencontrées

Durant ce projet, nous avons rencontré de nombreuses difficultés que nous avons réussi à surmonter.

La première difficulté majeure a été l'organisation. L'absence prolongée de la chef de projet pendant une partie de l'année (jusqu'à février) a entraîné de nombreuses incompréhensions au sein du groupe. En conséquence, certaines personnes ont été amenées à effectuer des tâches en double, notamment en programmation. La partie mécanique a également démarré relativement tard. De plus, en raison de problèmes de communication et de motivation parmi certains membres du projet, nous avons terminé le projet avec seulement trois membres, au lieu des cinq initialement prévus. Cela a parfois représenté une charge de travail importante pour les trois membres restants.

Le deuxième problème concernait l'ouverture de la trappe du parachute du corps. Notre premier système utilisait une bille pour verrouiller la trappe en la forçant dans une cavité. Bien que ce système ait répondu aux critères de torsion de la trappe, il ne satisfaisait pas les critères de traction, la trappe pouvant être retirée à la main. Ce problème a été découvert lors des tests au C'Space. Nous avons donc dû remplacer tout le mécanisme. Nous avons opté pour une solution avec une tige en métal qui s'insère dans un trou de la trappe. Bien que ces modifications aient pris beaucoup de temps et que nous ayons eu des doutes sur le résultat final, nous avons réussi à mettre en place ce nouveau système d'ouverture à temps, garantissant ainsi le bon fonctionnement de la trappe tout en respectant les critères de sécurité.

Le dernier problème rencontré concernait les membres restants. En effet, les trois membres présents au C'Space étaient également impliqués dans un autre projet (FuseX Elytra FX 33). La gestion simultanée de deux projets a parfois causé des retards pour l'un ou l'autre des projets. À l'avenir, il serait préférable d'éviter de participer à plusieurs projets en parallèle pour éviter ces problèmes de gestion du temps et de ressources.

Nous sommes tout de même très fiers du résultat et de ce que nous avons accompli (Figure 23).

Figure 23 : Photo des membres du projet avec la fusée avant son lancement

Conclusions

Le projet MF22 StellarDart, mené par l'association Elisa SPACE, a représenté une expérience enrichissante et instructive tout au long de l'année 2023-2024. Ce projet ambitieux a permis de tester et d'affiner des technologies et des méthodes dans le domaine de l'aéronautique et du spatial, tout en offrant une occasion précieuse d'apprentissage pratique pour les membres de l'association.

L'objectif principal de ce projet était de concevoir et de lancer une mini-fusée capable de réaliser un vol nominal, avec des phases de propulsion, balistique et de récupération, tout en intégrant divers systèmes expérimentaux pour collecter des données précieuses. Le projet a vu la mise en place de plusieurs innovations, notamment un système de séparation rotatif, une télémétrie longue portée, et une variété de capteurs pour mesurer les accélérations, les vitesses angulaires, et la température. Ces éléments ont été essentiels pour valider et optimiser les performances de la fusée.

La phase de conception a été marquée par la création de cartes électroniques sophistiquées, la gestion de la télémétrie, et l'intégration de divers capteurs. La réalisation des PCB a nécessité une attention particulière aux détails et une précision technique, illustrant la complexité et les défis associés à la conception de systèmes embarqués pour des applications spatiales. La fabrication de la fusée a également impliqué une approche méthodique pour garantir la solidité et la fonctionnalité des différentes parties, allant des ailerons aux modules de parachute.

Le lancement de la mini-fusée le 10 juillet a été un moment culminant du projet. Malgré quelques défis techniques, tels que des écarts entre la vitesse et l'altitude estimées et les valeurs mesurées, le vol a été considéré comme un succès global. La récupération intacte de la fusée et la bonne performance des systèmes embarqués ont validé de nombreux aspects de notre conception. Les graphiques de données obtenus ont fourni des informations précieuses sur la dynamique du vol, bien que des ajustements futurs soient nécessaires pour améliorer la précision des mesures.

Les résultats de ce projet ouvrent la voie à des améliorations futures. La collecte et l'analyse des données ont révélé des pistes d'optimisation pour les projets suivants, notamment en affinant les méthodes de mesure et en améliorant la fiabilité des systèmes électroniques. Les retours d'expérience nous permettront de renforcer les capacités de nos futurs projets, en intégrant les leçons apprises de cette expérience.

En conclusion, le projet StellarDart a non seulement démontré les compétences techniques et la capacité d'innovation de l'équipe Elisa SPACE, mais a également renforcé l'engagement de l'association à promouvoir l'exploration spatiale et à repousser les limites de la technologie aérospatiale. La réussite de ce projet témoigne du dévouement et du travail acharné des membres de l'association, ainsi que de l'importance de la collaboration et de l'expérimentation dans le domaine de l'aéronautique et du spatial. Nous attendons avec impatience les futures initiatives qui s'appuieront sur les fondations posées par StellarDart, contribuant ainsi à l'avancement de la recherche et de l'innovation dans ce domaine passionnant.

Remerciements

Nous tenons à exprimer notre profonde gratitude à nos partenaires pour leur soutien inestimable tout au long de ce projet. Leur collaboration et leur engagement ont été essentiels à la réussite de cette initiative, et nous les remercions sincèrement pour leur confiance et leur précieuse contribution.

Annexe

Annexe 1 : Stabtraj stabilité et trajectoire

STABILITO
Stabilité de fusée à ailerons

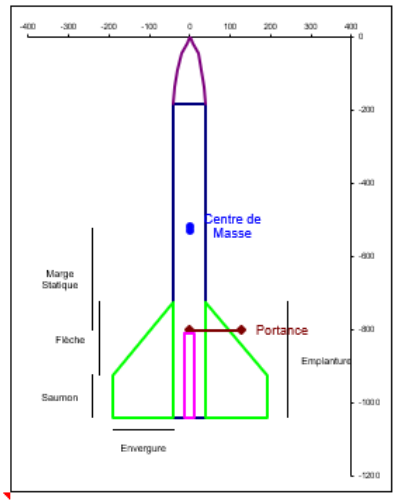
Remplir les cases jaunes

Fusée	
Nom	StellarDart
Club	Elisa Space
Matricule	MF 22
Type	Minifusée
Masse	2200 g sans propu
Centre de Masse	500 mm sans propu
Longueur totale	1038 mm

Propulseur	
Type	Pandora (Pro24-6G BS)
Position du bas	1038 mm

Coiffe	
Forme	Ogivale (pointue)
Hauteur	184 mm
Diamètre	80 mm

Ailerons	Mono-empennage
Emplanture 'm'	315 mm
Saumon 'n'	115 mm
Flèche 'p'	200 mm
Envergure 'E'	150 mm
Epaisseur 'ep'	3 mm
Nombre	3
Position du bas	1038 mm



########	Min	Résultats		Max
Finesse	10	13,0		20
Portance	15	24,2	24,2	30
MargeStat.	1,5 D	3,40 D	3,56 D	6 D
Couple	30	82,1	86,0	100
XCp		801 mm	801 mm	
MS /L		26% L	27% L	

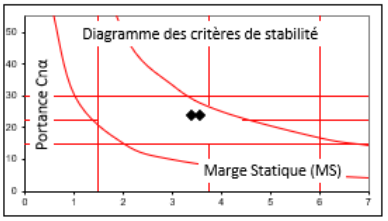
STABLE

Language/Langue Français

Fusée mono-diamètre,

	Propu plein	Propu vide	Sans propu
Masse propu	0,16 kg	0,08 kg	-
CdM propu	114 mm	114 mm	-
Masse fusée	2,36 kg	2,28 kg	2,2 kg
CdM fusée	529 mm	516 mm	500 mm

	XCp	Cnα
Coiffe	86 mm	2,0
Ailerons	865 mm	22,2



Commentaire libre :

Checksum : propu OK v3.4.5

TRAJECTO

Trajectographie de fusée

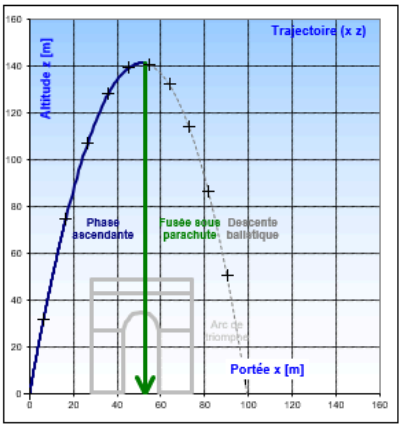
Remplir les cases jaunes

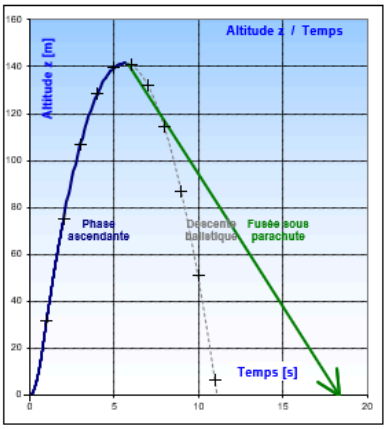
	Fusée
Nom	StellarDart
Club	Elisa Space
Matricule	MF22
Masse totale	2,3599 kg
Propulseur	Pandora (Pro24-6G BS)

	Traînée Aérdynamique
Surface Réf.	0,006377 m²
Cx	0,6

	Rampe de Lancement
Longueur	2,5 m
Élévation	80 °
Altitude	0 m

	DescenteSousParachute	
	Fusée	0 satellite
Masse	2,2843 kg	
Dépotage	N/A	
Ouverture para	5,8 s	
Type de para	Rond	
Surface para	0,29 m²	
Cx parachute	1	
Vitesse du vent	5 m/s	
Vitesse descente	11,2 m/s	
Durée descente	13 s	
Durée du vol	18 s	
Déport latéral	± 63 m	





02/08/2024	Temps	Altitude z	Portée x	Vitesse	Accélération	Efforts
Sortie de Rampe				19,3 m/s		
Vit max & Acc max				51 m/s	92 m/s²	
Culmination, Apogée	5,6 s	141 m	51 m	9 m/s		
Ouverture parachute fusée	5,8 s	141 m	53 m	9 m/s		15,7 N
Impact balistique	11,2 s	~0 m	100 m	50,2 m/s		2876 J

	Pour localiser la fusée
Couleur fuselage/coiffe	Galaxie / Coiffe blanche
Couleur parachute fusée	Noir

propu Cl
v3.4.

Commentaire libre :

Annexe 2 : Code séquenceur corps principal

```
/* Programme du séquenceur de StellarDart

Pin 0 : pas utilisé
Pin 1 : Output -> Ouverture trappe
pin 2 : Input_PULLUP -> port jack
Pin 3 : Output -> Envoi signal Exeprience
Pin 4 : Output -> Indicateur vol
Pin 5 : pas utilisé

Fait par : SCHWENDIMANN Baptiste
*/


//-----------Importation---------------------------------//

#include <SoftwareServo.h>

//------------Définition des pins-----------------------//

#define OpenSignal 1            // Pin ouverture trappe
#define IndicatorFlight 4       //Pin de la LED qui indique si la fusée est en vol
#define SignalJack 2            // Pin recevant le signal du port jack
#define SignalEnvoi 3           //Pin envoie le signal d'ouverture de trappre à la carte expérience

SoftwareServo servo;

//--------------------Initialisation des variables globales-------------------------//


const double ApoTime = 5.8;       // Temps que met la fusée pour atteindre l'apogée
const short delai = 1.5;          // plage d'ouverture du parachute [delai-ApoTime;delai+ApoTime]
unsigned long long GNDTime;               // T0 est le temps initial. Il sera modifier dans le programme par la suite.
unsigned long cli = millis();   // unsigned long => permet d'avoir une variable avec un stockage immense.
const short AngleF = 54;       // Angle que fait le servo lorsque la trappe est fermé
const short AngleO = 24;         // Angle que fait le servo lordque la trappe est ouvert


void setup() {
  servo.attach(OpenSignal);
  pinMode(OpenSignal, OUTPUT);
  pinMode(IndicatorFlight, OUTPUT);
  pinMode(SignalJack, INPUT_PULLUP);
  pinMode(SignalEnvoi, OUTPUT);


  while (digitalRead(SignalJack) == HIGH){//|| digitalRead(SignalExperience) == LOW // Clinotement de la LED (IndicatorFlight). Indique que le jack n'est pas branché
    digitalWrite(SignalEnvoi, LOW);      // HIGH pour dire "prêt" au découpleur
    clignotement(IndicatorFlight, 500, 250);
    servo.write(AngleO);
    SoftwareServo::refresh();
    delay(15);
  }

   servo.write(AngleF);
   SoftwareServo::refresh();
```

```
    delay(100);


  digitalWrite(IndicatorFlight, HIGH);
  digitalWrite(OpenSignal, LOW);
}

void loop() {
  GNDTime = millis();
  while(digitalRead(SignalJack) == HIGH) { // Si le jack est débranché la fusée est en mode "En vole".
    enVol();
  }
  while(digitalRead(SignalJack) == LOW){
    auSol();
  }

}

void enVol(){

  unsigned long currentTime = millis();
  if(currentTime >= GNDTime + (ApoTime*1000 + delai*1000)) {                // Temps apres le delai
    Ouverture();
  }else if(currentTime >= GNDTime + (ApoTime*1000 - delai*1000)) {
    Decoupleur();
  }

}


void Decoupleur(){
  digitalWrite(SignalEnvoi, HIGH); // LOW pour décrocher le découpleur
}

void Ouverture(){
  servo.write(AngleO);
  SoftwareServo::refresh();
  digitalWrite(SignalEnvoi, HIGH);     // LOW pour décrocher le découpleur
  clignotement(IndicatorFlight, 200, 100);
}

void auSol(){  // Si le jack est rebranché après le "décollage", le programme se résinialise et peut à nouveau fonctionné.
  //digitalWrite(OpenSignal, LOW);
  //clignotement(IndicatorFlight, 200, 30);


  servo.write(AngleF);
  SoftwareServo::refresh();

  while (digitalRead(SignalJack) == LOW);

  GNDTime = millis();

}

void clignotement(int pin, int on, int off){
    unsigned long t = millis();
    if(cli > t - on && cli < t - off){
      digitalWrite(pin, HIGH);
    }else if(cli > t - off){
      digitalWrite(pin, LOW);
    }else{
      cli = millis();
    }
}
```

Annexe 3 : Programme de la carte expérience coiffe

```
//////////Programme de la carte expérience coiffe de la fusée STELLARDART réalisé par Arthur DEWASTE//////////


#include <mySD.h> // Bibliothèque pour l'uilisation du lecteur µSD de la carte Lilygo

#include <SPI.h>

#include <LoRa.h>

#include <Adafruit_MPU6050.h>

#include <Adafruit_Sensor.h>

#include <Wire.h>


// Pins utilisés pour la LoRa

#define ss 18

#define rst 14

#define dio0 26


File root;

Adafruit_MPU6050 mpu;
```

```
int SPREADING_FACTOR = 7;

int TX_POWER = 14; // Valeur maax admise pour FuseX (ne pas dépasser)



int valueCount = 0; // Compteur pour le nombre de valeurs écrites dans le fichier actuel

String currentFileName = "data0.txt"; // Nom du fichier actuel



int counter = 0;



const double R = 8.314; // Constante universelle de gaz parfaits

const double rho = 1.204; // Masse volumique de l'air

const double voltageMax = 3.3; // voltage maximal (correspondant à la plage du pin analogique de l'ESP32)

const double kpaRangeTopVoltage = 3.33; // valeur maximale renvoyée par le capteur, par l'intermédiaire d'un pont diviseur de tension.



const int ledPin = 13; // Constante pour le pin de la LED expérience



const int numReadings = 10; // Nombre de lectures pour la moyenne glissante

const int dataNumber = 300; // Nombre de valeurs par fichier sur la carte SD

int readings[numReadings] = {0}; // Tableau pour stocker les lectures des capteurs

int currentIndex = 0; // Index actuel pour le tableau des lectures

int total = 0; // Somme des lectures pour le calcul de la moyenne

int sensorPin = 36; // Pin analogique où le capteur est connecté

int smoothedSensorValue; // Valeur lissée du capteur de pression différentielle



int sensorValue = 0, sensorMax = 4096, sensorOffset = 0; // Variables pour la valeur du capteur, valeur max du capteur, et l'offset

double voltage = 0, kpa = 0, speed = 0; // Variables pour la tension, pression en kPa

bool calibrationDone = false; // Booléen pour vérifier si le calibrage est effectué



////////////// fonctions utilisées pour ce programme ////////////



int adjustOffset(int sensorPin, int sensorOffset) { // Fonction "adjustOffset", permettant de calibrer le capteur pitot

  Serial.println("Commencement de l'etalonnage"); // Affichage pour le développement

  while (true) {

    sensorValue = analogRead(sensorPin) - sensorOffset; // Lecture de la valeur du capteur, ajustée par l'offset

    voltage = sensorValue * (voltageMax / sensorMax); // Convertion de la valeur analogique en tension

    kpa = ((voltage / kpaRangeTopVoltage) - 0.04) / 0.09; // Convertir la tension en pression (kPa) selon le datasheet du capteur MPX5010DP :
https://www.nxp.com/docs/en/data-sheet/MPX5010.pdf



    if (kpa > 0.05) { // si la pression est supérieure à 0.05 kPa...

      sensorOffset++; // ...Augmentation de l'offset

    } else if (kpa < -0.05) { // si la pression est inférieure à -0.05 kPa...

      sensorOffset--; // ... Diminution de l'offset

    } else {

      Serial.println("Etalonnage termine"); // Affichage d'un message de fin d'étalonnage
```

```
      break;
    }
    Serial.println("Etalonnage en cours..."); // Affichage d'un message pendant l'étalonnage
  }
  return sensorOffset; // Retourner la nouvelle valeur d'offset ajustée
}



int getSmoothedSensorValue() { // Fonction pour obtenir la valeur moyenne lissée
  return total / numReadings; // Calcul de la moyenne des lectures
}


void calculateAndPrintValues() {
  total -= readings[currentIndex];
  readings[currentIndex] = sensorValue;
  total += readings[currentIndex];
  currentIndex = (currentIndex + 1) % numReadings;

  // Calculer la moyenne mobile
  int smoothedSensorValue = total / numReadings;

  voltage = smoothedSensorValue * (voltageMax / sensorMax); // Conversion de la valeur du capteur en tension
  kpa = ((voltage / kpaRangeTopVoltage) - 0.04) / 0.09; // Conversion de la tension en pression en kPa

  speed = 3.6 * sqrt((2 * kpa * 1000) / rho); // Calcul de la vitesse en utilisant la conversion kPa en Pa
   if (isnan(speed)) {
    speed = 0;  // Si la vitesse est "nan", on définit mach à 0
  }



  /*Serial.print("Voltage: ");
  Serial.println(voltage, 3);
  Serial.print("Kpa: ");
  Serial.println(kpa, 1);
  Serial.print("vitesse :");
  Serial.println(speed);*/
}




// Fonction pour écrire les données sur la SD
void writeDataToSD(float data) {
```

```
  // Vérifie si le fichier actuel a atteint le nombre dataNumber
  if (valueCount >= dataNumber) {
    // Réinitialiser le compteur et incrémenter le nom du fichier
    valueCount = 0;
    incrementFileName();
  }

  // Ouverture du fichier pour écriture
  root = SD.open(currentFileName.c_str(), FILE_WRITE);
  if (root) {
    root.println(data);
    root.close();
  } else {
    Serial.print("Error opening ");
    Serial.println(currentFileName);
  }

  // Incrémenter le compteur de valeurs
  valueCount++;
}

// Fonction pour incrémenter le nom du fichier
void incrementFileName() {
  static int fileIndex = 1; // Index pour générer de nouveaux noms de fichier
  currentFileName = "data" + String(fileIndex++) + ".txt";
}

void printDirectory(File dir, int numTabs) {
  while (true) {
    File entry = dir.openNextFile();
    if (!entry) {
      break;
    }
    for (uint8_t i = 0; i < numTabs; i++) {
      Serial.print('\t'); // Indentation pour une meilleure lisibilité
    }
    Serial.print(entry.name());
    if (entry.isDirectory()) {
      Serial.println("/");
      printDirectory(entry, numTabs + 1);
    } else {
      Serial.print("\t\t");
      Serial.println(entry.size());
```

```
    }
    entry.close();
  }
}



void setup() {


  delay(1000);
  pinMode(ledPin, OUTPUT); // Initialisation de la LED expérience
  sensorOffset = adjustOffset(sensorPin, sensorOffset);
  Serial.begin(115200);
  Serial.print("Initializing SD card...");


  // Initialize SD library with SPI pins
  if (!SD.begin(13, 15, 2, 14)) {
    Serial.println("initialization failed!");
    return;
  }
  Serial.println("initialization done.");
    //initialize Serial Monitor
  Serial.begin(115200);
  while (!Serial);
  Serial.println("LoRa Sender");


  //setup LoRa transceiver module
  LoRa.setPins(ss, rst, dio0);



  while (!LoRa.begin(868188E3)) {
    Serial.println(".");
    delay(500);
  }
   // Change sync word (0xF3) to match the receiver
  // The sync word assures you don't get LoRa messages from other LoRa transceivers
  // ranges from 0-0xFF
  LoRa.setSyncWord(0xF3);


  SPREADING_FACTOR = constrain(SPREADING_FACTOR, 6, 12); // Limits of SPREADING_FACTOR
  TX_POWER = constrain(TX_POWER, -3, 17); // Limmits of TX_POWER dB
```

```
  LoRa.setSpreadingFactor(SPREADING_FACTOR);

  LoRa.setTxPower(TX_POWER);

  LoRa.setSignalBandwidth(125E3);           // 500 Hz = 0.0005 MHz Plus étaler = plus de vitesse de transmission et moins de portée (car plus de bruit). 7.8E3, 10.4E3, 15.6E3, 20.8E3, 31.25E3, 41.7E3, 62.5E3, 125E3, 250E3, and 500E3

  //LoRa.setCodingRate4(5);                  // entre 5 et 8 plus la valeurs est elevée plus les erreurs sont faible (portée augmente) débit est réduit 4/5 4/8

  //LoRa.setPreambleLength(8);            //Taille des données anti erreurs plus le chiffre est grand plus la portée augmente (et la vitesse diminue)



  Serial.println("LoRa Initializing OK!");



  Serial.begin(115200);

  while (!Serial)

    delay(10); // will pause Zero, Leonardo, etc until serial console opens



  Serial.println("Adafruit MPU6050 test!");



  // Try to initialize!

  if (!mpu.begin()) {

    Serial.println("Failed to find MPU6050 chip");

    while (1) {

      delay(10);

    }

  }

  Serial.println("MPU6050 Found!");



  mpu.setAccelerometerRange(MPU6050_RANGE_8_G);

  Serial.print("Accelerometer range set to: ");

  switch (mpu.getAccelerometerRange()) {

  case MPU6050_RANGE_2_G:

    Serial.println("+-2G");

    break;

  case MPU6050_RANGE_4_G:

    Serial.println("+-4G");

    break;

  case MPU6050_RANGE_8_G:

    Serial.println("+-8G");

    break;

  case MPU6050_RANGE_16_G:

    Serial.println("+-16G");

    break;

  }

  mpu.setGyroRange(MPU6050_RANGE_500_DEG);

  Serial.print("Gyro range set to: ");
```

```
 switch (mpu.getGyroRange()) {

 case MPU6050_RANGE_250_DEG:

   Serial.println("+- 250 deg/s");

   break;

 case MPU6050_RANGE_500_DEG:

   Serial.println("+- 500 deg/s");

   break;

 case MPU6050_RANGE_1000_DEG:

   Serial.println("+- 1000 deg/s");

   break;

 case MPU6050_RANGE_2000_DEG:

   Serial.println("+- 2000 deg/s");

   break;

 }


 mpu.setFilterBandwidth(MPU6050_BAND_21_HZ);

 Serial.print("Filter bandwidth set to: ");

 switch (mpu.getFilterBandwidth()) {

 case MPU6050_BAND_260_HZ:

   Serial.println("260 Hz");

   break;

 case MPU6050_BAND_184_HZ:

   Serial.println("184 Hz");

   break;

 case MPU6050_BAND_94_HZ:

   Serial.println("94 Hz");

   break;

 case MPU6050_BAND_44_HZ:

   Serial.println("44 Hz");

   break;

 case MPU6050_BAND_21_HZ:

   Serial.println("21 Hz");

   break;

 case MPU6050_BAND_10_HZ:

   Serial.println("10 Hz");

   break;

 case MPU6050_BAND_5_HZ:

   Serial.println("5 Hz");

   break;

 }


 Serial.println("");

 delay(100);
```

```
}



////////////////////// Boucle loop ////////////////////////////

void loop() {

  digitalWrite(ledPin, HIGH); // Allumage de la LED expérience, signifiant que les experiences sont fonctionnelles et initialisés.

  sensorValue = analogRead(sensorPin) - sensorOffset; // Calcul de la valeur renvoyée par le capteur après calibrage

  smoothedSensorValue = getSmoothedSensorValue(); // Obtention de la valeur lissée du capteur

  calculateAndPrintValues(); // Calcul et affichage des valeurs

  // Optensions des valeurs de l'accéléromètre et gyroscope
  sensors_event_t a, g, temp;
  mpu.getEvent(&a, &g, &temp);

////////////// Transmission et stockage des données /////////////

 // Écriture des données sur la carte SD

 /*
Architechture d'envoi et stockage des données :
- valeur lissée du différentiel de pression
- valeur de vitesse air calculée
- accélération selon x
- accélération selon y
- accélération selon z
- gyro selon x
- gyro selon y
- gyro selon z
- température
 */

 // Ecriture sur la carte µSD
  writeDataToSD(smoothedSensorValue);
  writeDataToSD(speed);
  writeDataToSD(a.acceleration.x);
  writeDataToSD(a.acceleration.y);
  writeDataToSD(a.acceleration.z);
  writeDataToSD(g.gyro.x);
```

```
  writeDataToSD(g.gyro.y);

  writeDataToSD(g.gyro.z);

  writeDataToSD(temp.temperature);

  writeDataToSD(888);



  //Transmission des données via télémesure LoRa

  LoRa.beginPacket();

  LoRa.println(smoothedSensorValue);

  LoRa.println(speed);

  LoRa.println(a.acceleration.x);

  LoRa.println(a.acceleration.y);

  LoRa.println(a.acceleration.z);

  LoRa.println(g.gyro.x);

  LoRa.println(g.gyro.y);

  LoRa.println(g.gyro.z);

  LoRa.println(temp.temperature);

  LoRa.println("fin");

  LoRa.endPacket();

  counter++;

  delay(150);

}
```