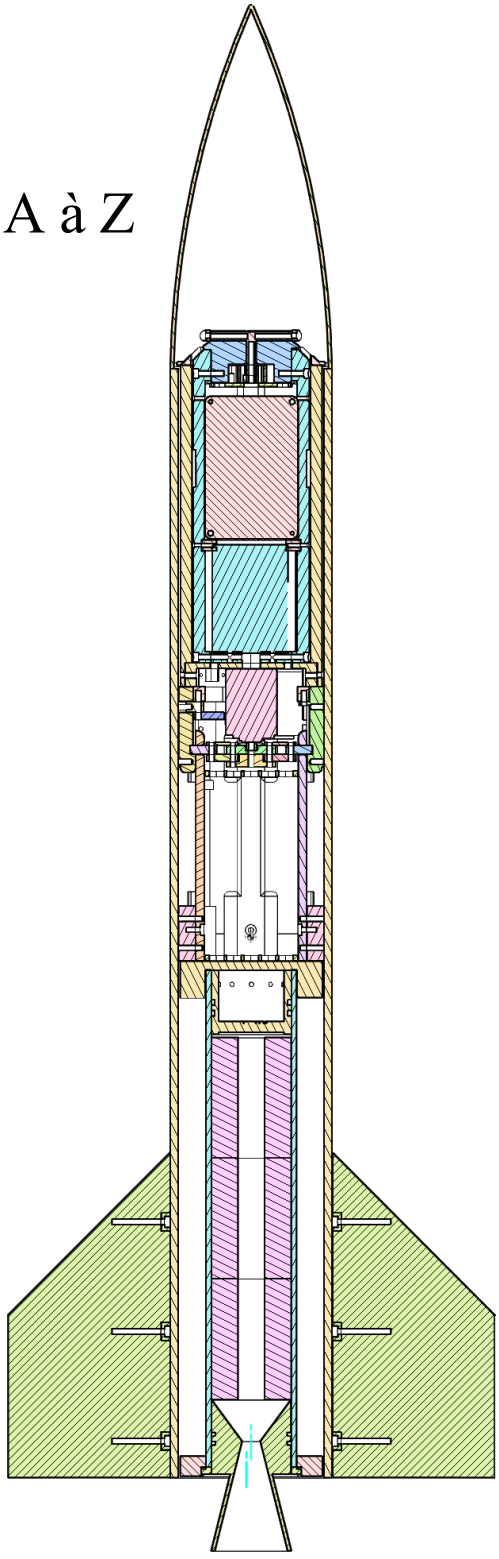
Mémoire

Titre du projet :

Construction d'une fusée de A à Z



Élaboré par :

Enzo Sanchez Sartori

&

Elliot Matton–Marie

Avec l'aide du professeur :

Nicolas Debons

Sommaire

1. Introduction

Introduction générale

Qui n'a jamais eu cette sensation spontanée de vouloir créer quelque chose, de matériel ou non. Ce qui caractérise l'humanité à notre sens, c'est cette inventivité, au début pour survivre, puis vivre et une fois installé, pour explorer de nouvelles envies. Ce cycle éternel montre à quel point l'être humain est déterminé et est prêt à tout pour faire l'impensable : traverser les océans pendant des mois en bateau pour découvrir de nouveaux continents ou encore, amener des hommes sur la Lune. Les moyens employés sont juste démesurés.

Aujourd'hui la fusée est l'un des engins les plus complexes que notre société n'ait jamais construits. Il regroupe un ensemble de différentes technologies dans l'unique but de transférer une charge utile d'un point A à un point B.

Avant le projet NOVA nous avons tous les deux construit et imaginé bien d'autres choses. Notre précédent projet commun concerne une voiture radiocommandée (RC) à haute vitesse de taille 1/10. Chacun de nous deux a donc fabriqué sa voiture : Enzo a voulu réaliser une voiture compacte tandis qu'Elliot a voulu agrandir le châssis pour accueillir plus de composants. L'objectif en vitesse est 100 km/h. Récemment, nous avons mesuré une vitesse maximale de 70 km/h (voir *Annexe 1.1 et 1.2*).

Explication du logo du projet NOVA EE

Notre logo est circulaire, en noir et blanc, porte le mot **NOVA** (le nom de notre fusée) en lettres majuscules. Il y a une fusée simpliste au centre, elle est entourée d'éléments rappelant l'espace, comme une étoile et une trajectoire. Les noms **PULSAR X AEN** en haut à gauche (voir *Annexe 1.3*), dans la courbe du cercle. Le design est futuriste, avec une thématique spatiale (nos compétences de graphisme ne sont pas assez poussées pour être capables de réaliser ce logo donc on a demandé à Bing IA).

C'est dans une volonté d'apprentissage et de curiosité que nous avons démarré ce projet il y a plus d'un an et demi. L'objectif principal est d'imaginer, de construire puis de procéder aux vols d'une fusée expérimentale* pour en analyser les données et les confronter à nos calculs théoriques.

*Ce type de classification de fusée provient des normes de plusieurs clubs et associations à l'international comme en France.

Objectifs du mémoire

Dans un premier temps, nous nous sommes intéressés à la fabrication d'un ordinateur de bord capable d'enregistrer les données de vol de notre fusée, telles que le temps de vol, l'inclinaison, l'altitude, la température, l'accélération, et les moments clés du vol (décollage, apogée, atterrissage) (parties 4.3.1. et 4.3.2.). Pour analyser et rendre ces informations plus lisibles, nous avons besoin d'un logiciel dédié. Celui-ci devra également calculer des données supplémentaires comme la vitesse verticale et la vitesse moyenne (partie 4.3.3.).

L'ordinateur de bord est également chargé de gérer l'ouverture du système de récupération au moment adéquat. Pour assurer le retour intact de la fusée, nous avons conçu un système de récupération correspondant à un parachute, maintenu par un mécanisme jusqu'à son déploiement (Partie 4.1.).

Par ailleurs, pour faciliter le rangement et l'extraction de l'ordinateur de bord, nous avons pensé à un système de tiroir (Partie 4.2.), qui contient également la charge utile.

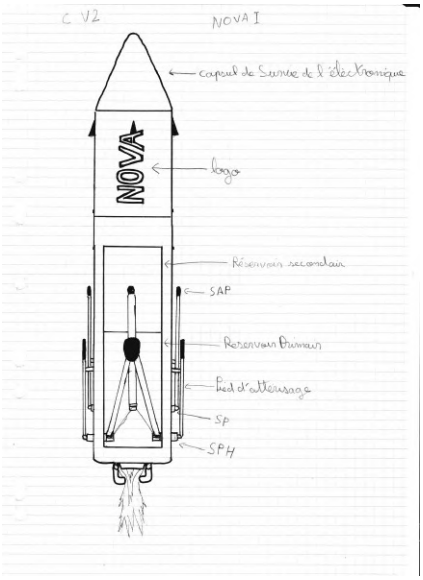
Enfin, pour protéger les composants internes et améliorer l'aérodynamisme, la fusée a un fuselage et une ogive. Ces éléments contribuent également à sa stabilité en vol, en complément des ailerons (Partie 4.4.).

Par ailleurs, d'un point de vue expérimental, nous avons testé plusieurs aspects de notre fusée, comme l'ordinateur de bord, notamment au niveau du bon fonctionnement de ses capteurs. Nous avons pour cela procédé à l'étalonnage du capteur barométrique (Partie 4.3.2.). Nous avons aussi testé le déploiement du parachute (Partie 4.1.2.).

Pour détailler l'avancée de notre projet au cours des 18 derniers mois, nous avons décidé de la présenter sous la forme d'une chronologie.

2. Chronologie du projet

Mai 2023 :
Nous commençons à réfléchir à construire une fusée. Les idées sont là, mais trop ambitieuses : les objectifs principaux étaient le décollage puis l'atterrissage grâce à deux réservoirs et des pieds déployables. Le projet est finalement mis de côté, au profit de celui concernant les voitures radiocommandées. Un des premiers schémas date de mai 2023 (voir *Figure 1*)



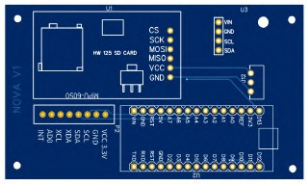
<u>Figure 1 :</u> Un des premiers schémas de fusée du projet NOVA.

Mai 2024 :

Un an après, l'idée nous revient, avec une vision plus réaliste et donc faisable. Nos nouveaux objectifs : créer une fusée capable de décoller, analyser les données de vol comme l'inclinaison, l'altitude, la pression ou bien la température, puis programmer la fusée pour qu'elle puisse d'elle-même ouvrir le parachute à l'apogée grâce à l'ordinateur de bord.

Fin mai 2024 :

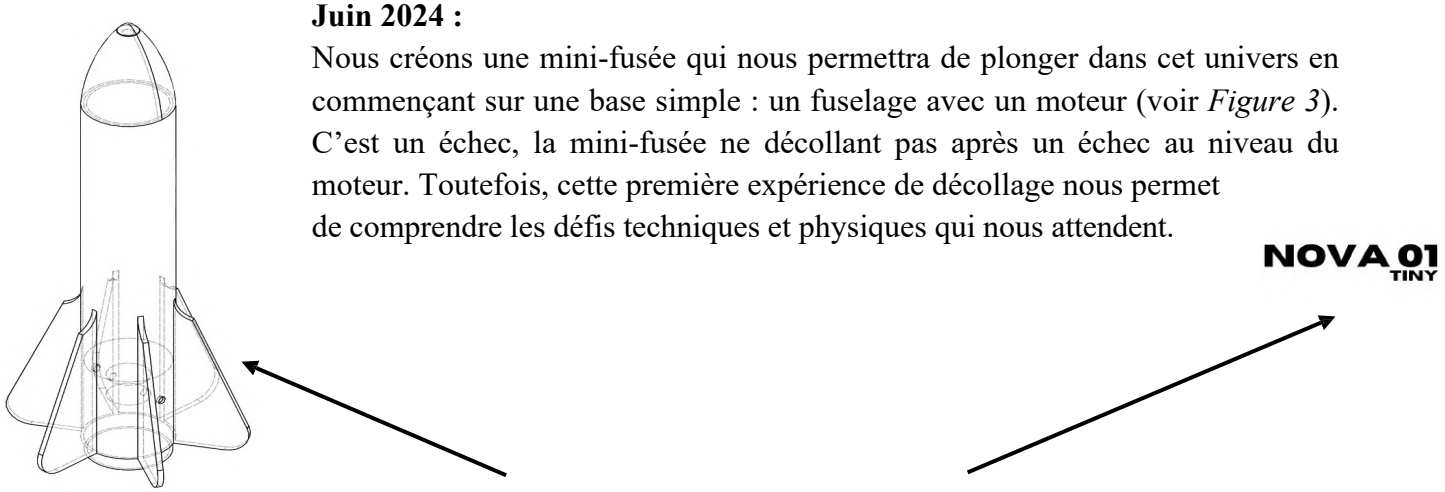
Elliot commence à s'intéresser à la microélectronique et crée la première version de l'ordinateur de bord. À cette date, cet ordinateur n'a pas de nom et est composé d'un circuit très simple. On le teste mais il est malheureusement incomplet (une première version de l'ordinateur est donnée en *Figure 2*).



<u>Figure 2 :</u> Première version de l'ordinateur

Juin 2024 :

Nous créons une mini-fusée qui nous permettra de plonger dans cet univers en commençant sur une base simple : un fuselage avec un moteur (voir *Figure 3*). C'est un échec, la mini-fusée ne décollant pas après un échec au niveau du moteur. Toutefois, cette première expérience de décollage nous permet de comprendre les défis techniques et physiques qui nous attendent.

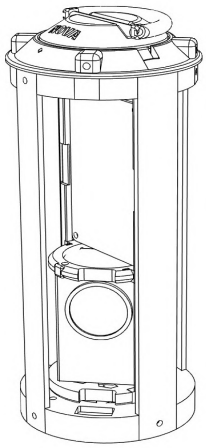


<u>Figure 3 :</u> Notre premier modèle 3D de fusée et le logo associé au projet « Nova 01 tiny » (voir *Annexe 2* pour une photo de la fusée).

Été 2024 :

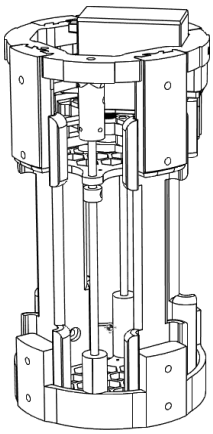
Elliot, toujours intéressé par la microélectronique, décide de développer l'ordinateur. Il aura fallu pour sa conception plusieurs versions de l'ordinateur, chacune des versions étant réalisée sur le logiciel "easy EDA". Finalement, après la 7ème itération, nous obtenons l'ordinateur que

nous nommons ARC pour "**A**dvanced **R**ocketry **C**omputer" (on trouvait que ça sonnait moins bien en français…). Pour en savoir plus sur son développement, voir (partie 5.3.).



Fin août 2024 :
Elliot modélise un système de tiroir qui servira de réservoir pour la charge utile et donc qui maintiendra ARC (l'ordinateur de bord) en place dans la fusée et facilitera l'extraction à l'atterrissage. Le système de tiroir est modélisé en 3D en *Figure 4* et est détaillé dans la partie 5.2.).

Figure 4 : Modélisation 3D du tiroir.



Fin septembre 2024 :
Enzo modélise le système de récupération qui permettra de libérer un parachute à l'apogée de la trajectoire à la demande de l'ordinateur. Cette partie cruciale est nécessaire si l'on veut récupérer la fusée en entier. Le système de récupération modélisé en 3D figure 5 et la suite partie 5.1.)

igure 5 : Modélisation 3D du système de récupération.

Mi-novembre 2024 :
On a toutes les parties de la fusée : ogive, tiroir + charge utile (ordinateur de bord et caméra), système de récupération, le moteur (une version imprimée en 3D qui nous permet de le visualiser) et les ailerons. La prochaine page présente un schéma pour visualiser toutes ces parties (voir *Figure 6*).

Attention : le mémoire n'est pas écrit dans l'ordre chronologique de développement de notre projet.

3. Schéma complet de la fusée

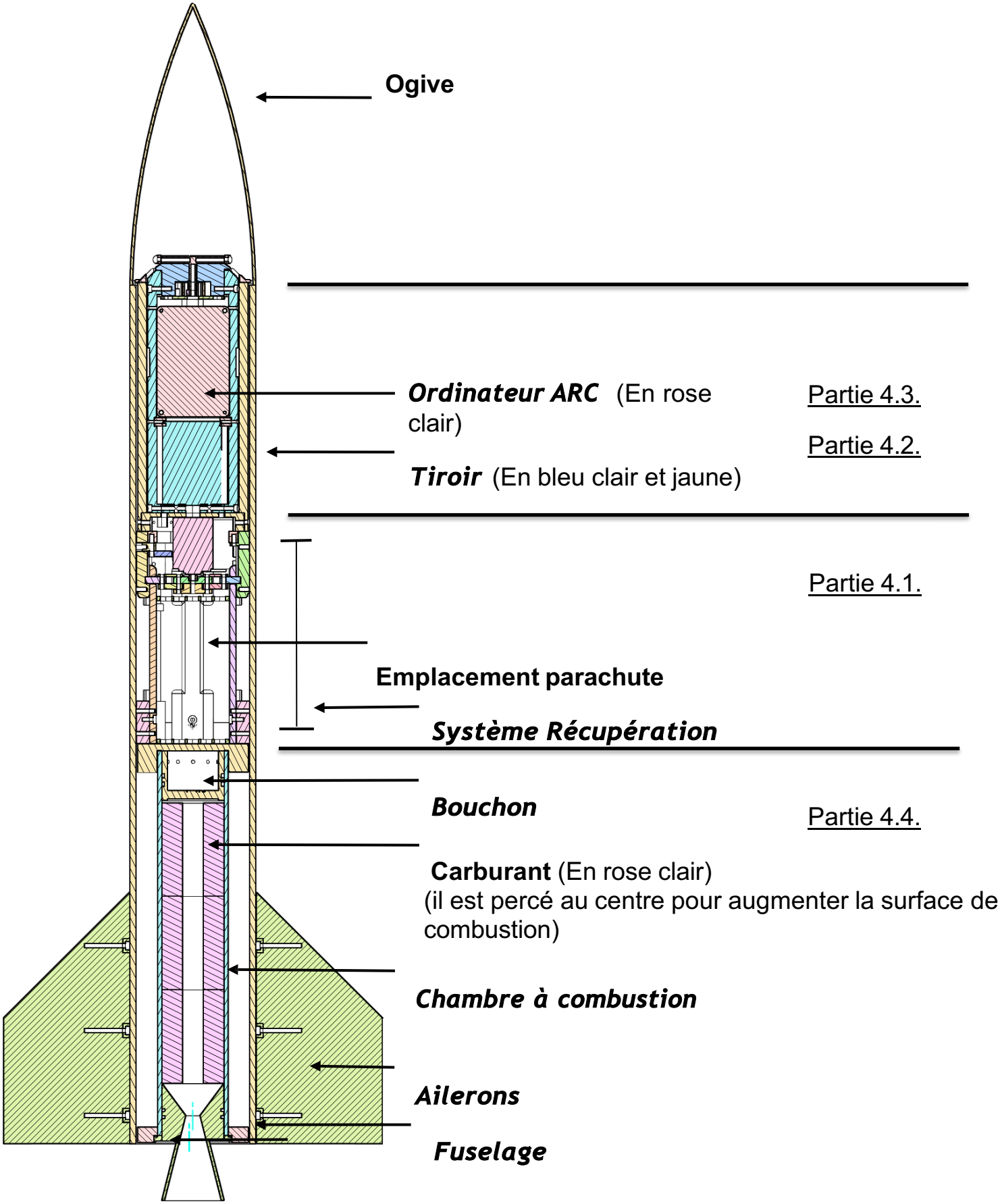


Figure 6 : Schéma complet de la fusée.

4. Résultats et discussion

4.1. Système de récupération

4.1.1. Système

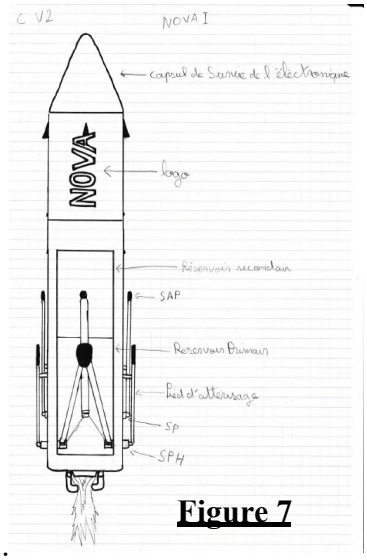


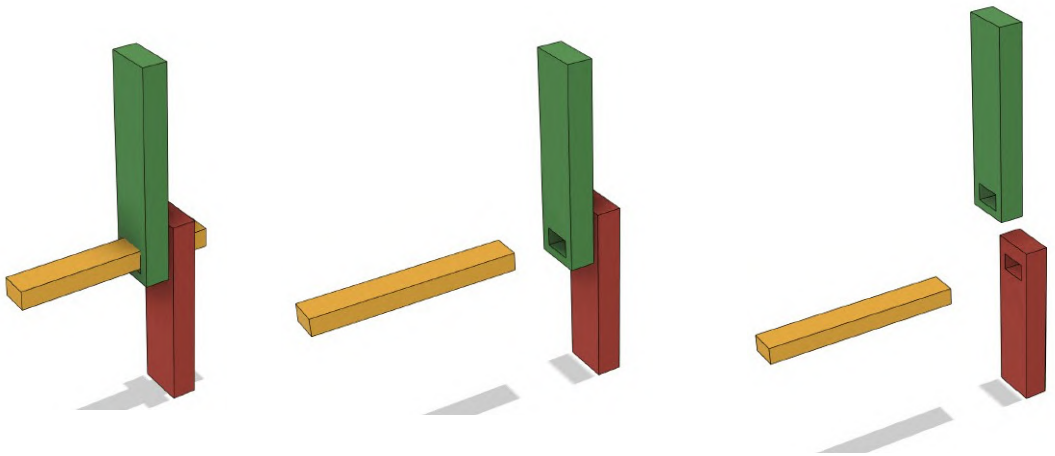
Figure 7

Comme mentionné en page 3, en **mai 2023**, nous voulons faire **atterrir la fusée** à la manière des boosters de *SpaceX* grâce à **des pieds** déployables (voir *Figure 7* ci-contre avec les pieds d'atterrissage).
En septembre 2024, avec une vision plus réaliste du projet, Enzo se penche sur la réalisation du système de récupération mais cette fois-ci grâce à un parachute se déployant à l'apogée.
Enzo pense à 4 systèmes différents, regroupés en deux idées phares :

- Première idée : la fusée se sépare en deux et le parachute sort du milieu,
- Deuxième idée : une trappe se détache et laisse sortir le parachute sans modifier complètement la structure de la fusée.

Pour notre système de récupération

Enzo a opté pour la première idée et a commencé par modéliser un prototype du système en se concentrant sur les parties mouvantes.
Le système est basé sur le principe où la partie rouge (Pôle Base) et la base verte (Pôle Amovible) sont maintenus ensemble par ce qu'on peut nommer un "*trigger*" (gâchette en français, jaune dans le schéma). Il maintient les deux parties ensemble et, une fois enlevé, les deux pôles sont libres de mouvement (voir *Figure 8*).



Le système est bloqué. Le *trigger* s'enlève. Les deux partie rouge et verte se libèrent

Figure 8 : Évolution du système en trois étapes.

Ce système est utilisé dans de nombreux systèmes à relâchement rapide.
Ici, les pôles Base (rouge) et Amovible (vert) seront sous tension constante à l'aide de ressorts fixés de part-et-d'autre des pôles pour un relâchement forcé quand le *trigger* sera retiré.

Ensuite Enzo a prototypé le système des *triggers* qui permet un relâchement rapide au moment voulu (*Figure 9*).

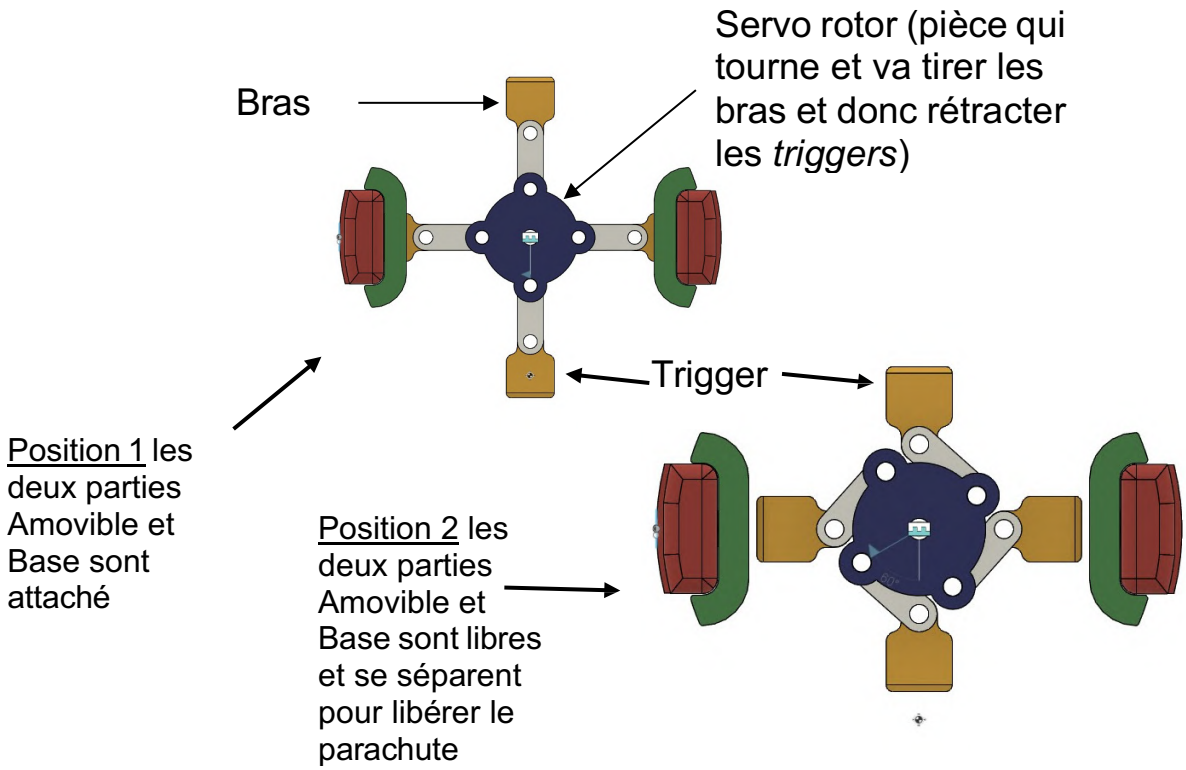
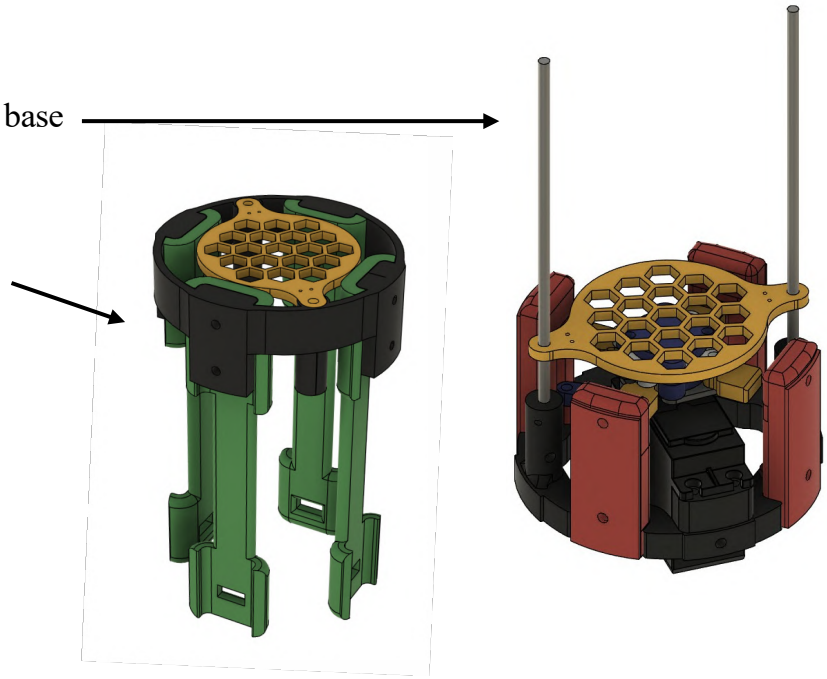


Figure 9 : Système des *triggers* permettant un relâchement rapide.

Les *triggers* sont tirés grâce à des bras représentés en blanc et attachés à un cercle bleu nommé *Servo rotor* car c'est un servo moteur qui fera tourner, (Notre servo moteur est un moteur avec un couple de 294 N (valeur affichée sur le moteur non testé de 30 kilogrammes)) et il permet grâce à l'ordinateur de bord de tourner à un angle précis avec un capteur interne.

Une fois le principe compris et validé, Enzo est passé à la réalisation du modèle 3D en gardant les spécificités données précédemment mais en prenant en compte la place pour le parachute et la résistance des matériaux (voir *Figure 10*). Cette dernière, adaptée au matériau utilisé lors d'une impression 3D (PETG, polyéthylène glycol), a été évaluée de manière empirique à la suite de nombreuses impressions 3D lors de précédents projets. Pour en savoir plus sur les problèmes rencontrés avec les système de récupération, voir *Annexe 3.1.*

1. J'ai commencé par la base
2. Puis la partie mobile



<u>Figure 10 :</u> Assemblage final.

4.1.2. Le parachute

Pour le patron du parachute, Enzo est allé sur un site de la NASA qui donne une formule pour l'équation de la traînée (*Figure 11*). En considérant le mouvement rectiligne et uniforme durant la chute du système {parachute + moitié de la fusée}, et en supposant un bilan des forces composé uniquement du poids du système $\vec{P}$ et de sa force de traînée $\vec{D}$, on obtient grâce à la 1re loi de Newton la vitesse du système pendant sa chute (*Figure 11*).

Equation de la traîner:

$$D = Cd \cdot \frac{\rho \cdot V^2}{2} \cdot S$$

Equation de la vitesse:

$$V = \sqrt{\frac{2 \cdot P}{Cd \cdot \rho \cdot S}}$$

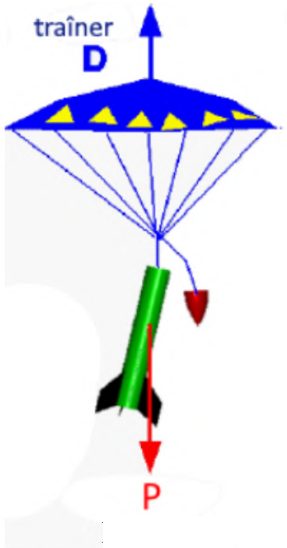
ρ = Densité de l'air $\left(1.229\ kg/m^3\right)$

S = Surface parachute

V = Vitesse

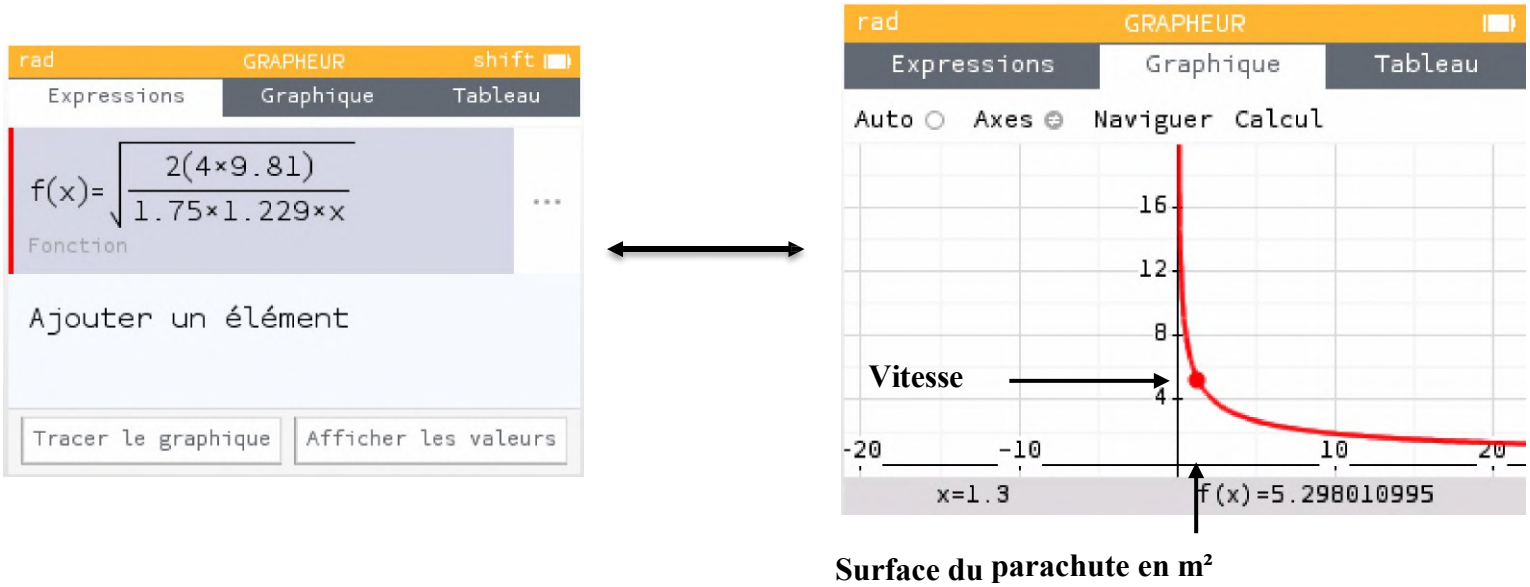
Cd = Coefficient de traînée

D = traîner



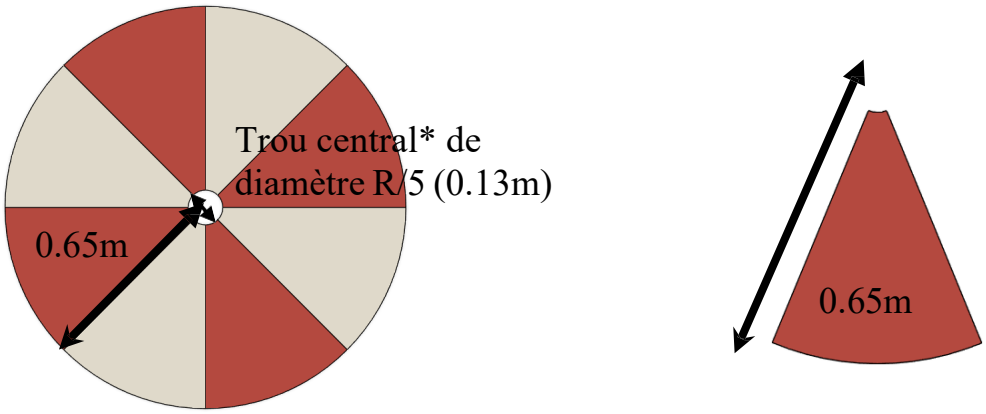
<u>Figure 11 :</u> Équation de la force de traînée et obtention de la vitesse du parachute en régime permanent.

Enzo a tiré de l'équation de la vitesse une fonction pour avoir la vitesse en fonction de la surface du parachute (*Figure 12*).



<u>Figure 12 :</u> Tracé de la fonction de la vitesse en fonction de la surface de parachute sur une calculatrice Numworks.

Pour une vitesse de 6 m/s environ, on trouve une surface de parachute de 1,3 m^2. Comme (0,65 m)^2 x $\pi \approx$ 1,3 m^2, il faut donc choisir un cercle de rayon 0,65 m. On dessine donc le patron dessiné en *Figure 13* qu'il faudra répéter 8 fois pour donner le parachute *in fine*.



<u>Figure 13 :</u> Dessin du patron (à droite) et du parachute assemblé (à gauche).
***Il faut un trou au centre du patron pour stabiliser le parachute lors de la descente.**

Nous avons testé le bon déploiement du parachute au lycée en le lançant à une hauteur d'environ 5 mètres et en le photographiant une fois déployé (voir *Annexe 4*).

4.2. Tiroir

La **partie "tiroir"** est la première partie de la fusée qui a été réalisée, son but étant de stocker la charge utile qui comprend un ordinateur de bord (ARC) et une caméra.

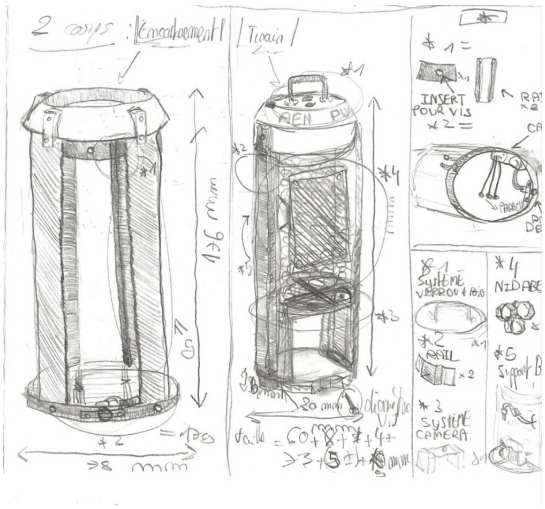


Figure 14 : premiers croquis de la partie "tiroir".

Cette **partie "tiroir"** est donc composée d'un cadre qui va rester fixe dans le fuselage de la fusée, et d'un tiroir avec la charge utile, qu'on va pouvoir retirer et remettre facilement grâce à une poignée.

Il y a eu très peu de modifications importantes sur le modèle car en général les problèmes étaient des mauvaises mesures qui étaient donc facile à régler.

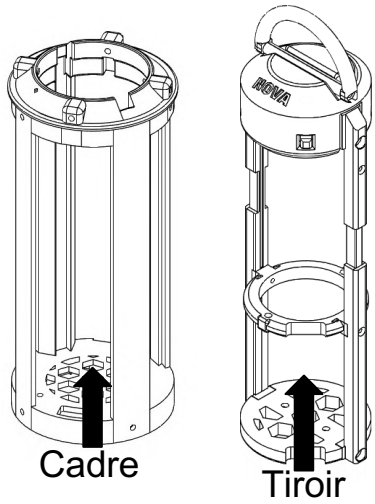


Figure 15 : Cadre + tiroir dessinés sur le logiciel Fusion

Enfin, le système de fermeture (voir *Figure 16*) permet au tiroir de rester en place lors des vols. Du fait d'une anticipation erronée des mesures, deux tentatives ont suffi à développer un prototype fonctionnel pour ce modèle (voir *Annexe 3.2* pour les problèmes rencontrés avec le tiroir).

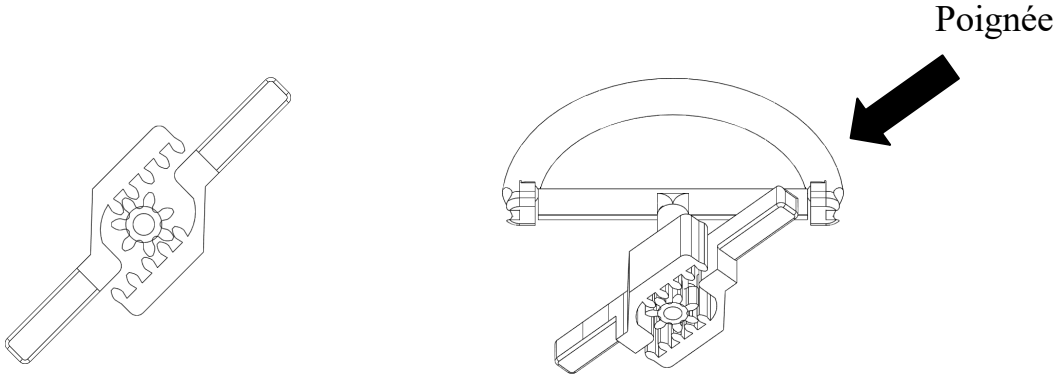


Figure 16 : Poignée et système de fermeture.

Le cadre permet aussi de fixer l'ogive/coiffe avec 4 trous.
Le fonctionnement du tiroir est le suivant : tout d'abord, l'ordinateur de bord et la caméra sont placés à l'intérieur du tiroir. Ensuite, le tiroir est partiellement inséré dans le cadre pour permettre de connecter les câbles du cadre à l'ordinateur de bord (voir *Figure 17*). Une fois cette étape réalisée, le tiroir est complètement inséré, puis la poignée est tournée pour verrouiller sa position.

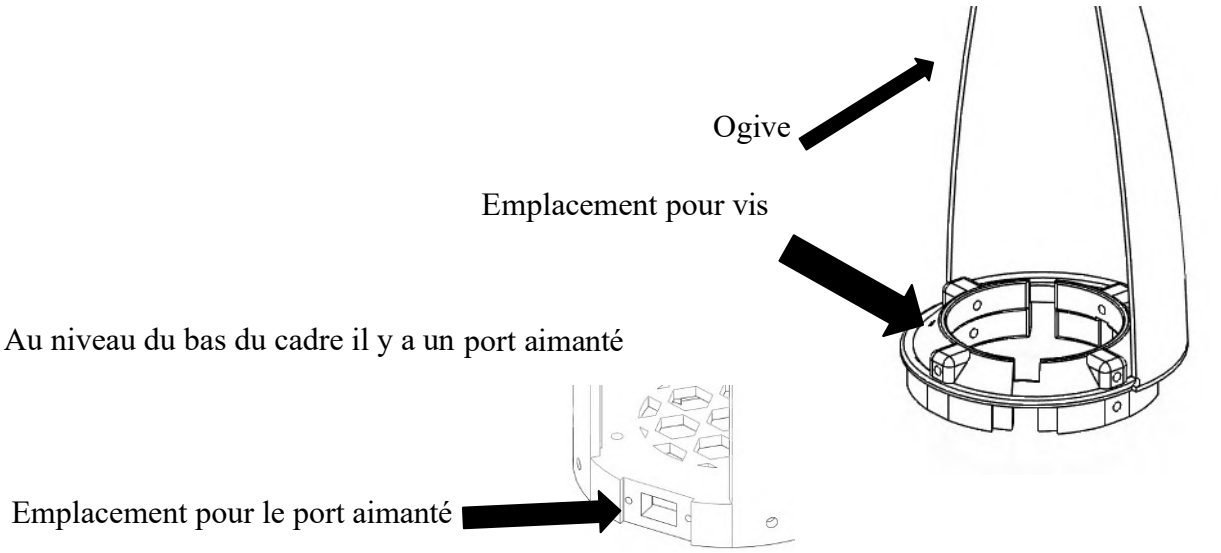


Figure 17 : Fonctionnement du tiroir.

4.3. Ordinateur de bord (ARC)

4.3.1. Ordinateur

Attention : l'ensemble du matériel et des consommables utilisés dans ce projet est présenté en *Annexe 5*.

ARC est un ordinateur de bord (circuit imprimé) conçu spécifiquement pour le projet Nova. Son rôle est double :

1. Activer le parachute au moment de l'apogée, c'est-à-dire au point le plus haut du vol.
2. Enregistrer les données de vol sur une carte micro SD pour les analyser par la suite.

Pour concevoir ARC, nous avons dû réfléchir au choix des composants, en commençant par le microcontrôleur. Initialement, nous avons opté pour une Arduino Nano, car elle est compacte et facile à programmer.

L'ordinateur doit collecter un maximum d'informations sur la fusée pendant son vol, notamment :

- le temps de vol,
- la hauteur atteinte,

- l'orientation de la fusée,
- l’accélération,
- la température ambiante.

Pour répondre à ces besoins, nous avons sélectionné plusieurs capteurs adaptés:

- Un MPU6050 pour mesurer l’orientation et l’accélération.
- Un BMP280 pour mesurer la hauteur (via la pression atmosphérique) et la température.
- L’Arduino Nano pour gérer les calculs de détermination du temps de vol.

Nous devions également enregistrer toutes ces informations. Initialement, nous avons envisagé d’utiliser la mémoire interne de l’Arduino Nano, mais elle s’est révélée insuffisante. Nous avons donc choisi une solution plus adaptée : une carte micro SD, accompagnée d’un lecteur pour enregistrer toutes les données.

Pour rappel, le déploiement du parachute nécessite un servomoteur, ce qui a motivé l’ajout d’une prise.

La première version de l’ordinateur était relativement simple et peu adaptée aux besoins expérimentaux.

Nous avons donc dû repenser entièrement l’ordinateur de bord.

Nous avons tout d’abord changé le microcontrôleur qui n’était pas assez puissant pour faire fonctionner le code. Nous avons donc opté pour un ESP32. Puis nous avons ajouté des composants tels qu’un bouton pour allumer et éteindre l’ordinateur, un buzzer, des LEDs, des résistances, des condensateurs, des transistors, une deuxième prise et un emplacement pour un module LoRa (Long Range, qui permettra *in fine* de communiquer avec la fusée lors des vols).

Le buzzer et les LEDs servent à indiquer que les autres composants fonctionnent correctement.

Les résistances permettent de stabiliser le courant pour ne pas abîmer les LEDs et le buzzer. Les condensateurs servent à filtrer les variations de tension pour le servomoteur et les transistors.

Nous avons ajouté une autre prise pour un câble de détection aimanté qui se déconnectera lors du décollage. Nous avons également laissé des connexions pour un module LoRa que nous n’allons pas installer pour l’instant car nous voulons nous concentrer sur l’objectif principal du projet.

Nous avons donc fait un schéma papier de toutes les connexions nécessaires (*Annexe 6.1.*).
Nous avons par la suite fait le même schéma de connexions (*Figure 18*) sur le logiciel EASY EDA.

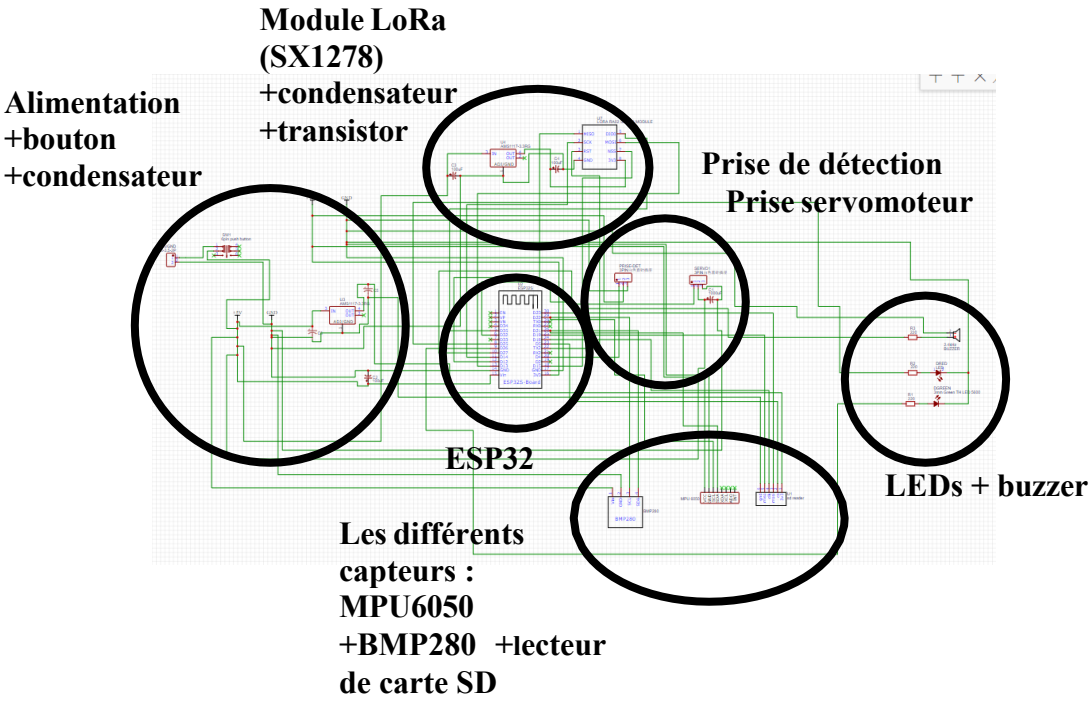


Figure 18 : Schéma de connexions de l'ordinateur de bord.

La suite de la conception consiste à placer chaque composants sur la carte et donc par définitions définir la taille de l'ordinateur de bord, puis de connecter chaque composant entre eux avec des routes.
Il y en tout 6 itérations (*Figure 19*).

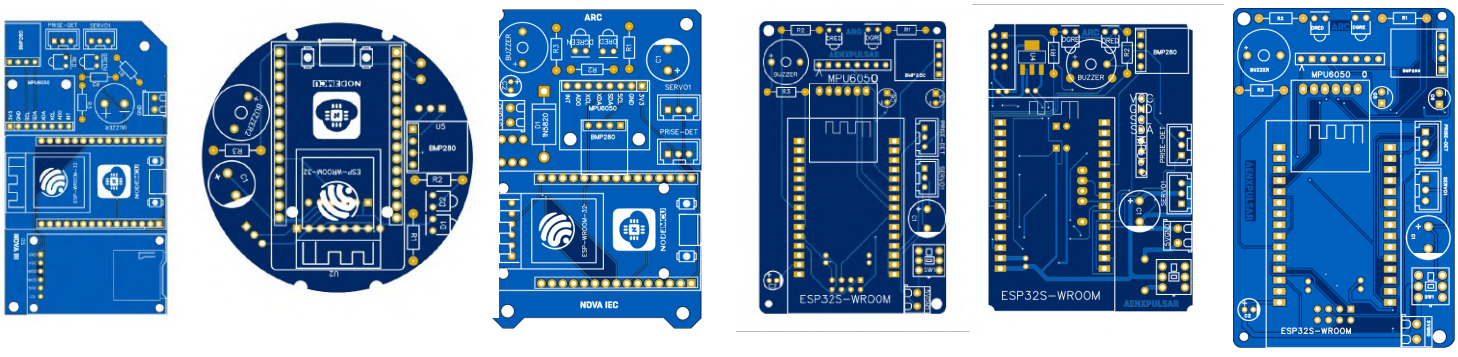


Figure 19 : Les 6 itérations pour ARC (tout à droite : le circuit actuel).

Cette dernière itération a donc pu être commandée sur un site de production de circuit imprimé. Par la suite, nous avons effectué des tests pour s'assurer que l'ordinateur fonctionne. Ils se sont avérés longs mais concluants.
Nous présentons dans la partie suivante le fonctionnement des capteurs MPU6050 et BMP280, pour répondre en particulier à la question de la mesure de l'altitude de la fusée.

4.3.2. Quel capteur utiliser pour mesurer l'altitude ?

Au début du projet nous avons fait face à la problématique de savoir par quel moyen nous pouvons calculer l'altitude. Deux solutions s'offraient à nous, soit utiliser un baromètre, soit un accéléromètre.
Nous avons donc voulu comparer ces deux capteurs sur plusieurs points : précision et puissance de calcul adaptée à l'ordinateur de bord. Pour ce faire, nous voulions tester les deux capteurs puis comparer les données obtenues pour conclure au choix du meilleur capteur.
Nous commençons par présenter le fonctionnement de chacun des capteurs, en commençant par le capteur barométrique dont une image est représentée en *Figure 20*.

<u>Figure 20 :</u> Le capteur barométrique BMP280.

Le BMP280 est un capteur qui mesure la pression, l'altitude et aussi la température. Ce capteur mesure des données barométriques pour ensuite donner une valeur approchée de l'altitude. Pour obtenir l'altitude, une loi plus proche de la réalité que la loi fondamentale de la statique des fluides (pour laquelle la température est constante) et celle du nivellement barométrique pour une variation linéaire de température :

Cette loi nous donne la Pression P (en hPa) en fonction de : $P = P_0 \cdot \left(1 - \frac{(L \cdot h)}{T_0}\right)^{\frac{g \cdot M}{R \cdot L}}$

- P_0 = pression de départ en hPa (au sol ou au niveau de la mer)
- L = gradient thermique (diminution de température par rapport à une distance)
  = 0.0065 K·m
- h = altitude en m
- T_0 = température initiale en K (au sol ou au niveau de la mer)
- g = intensité du champ de la pesanteur
- M = masse molaire de l'air (environ 0,029 kg·mol^{-1})
- R = constante des gaz parfaits (8,314 J·mol^{-1}K^{-1})

Tout d'abord on va calculer le quotient (constant) apparaissant en exposant dans la formule : $\frac{(9.81 \cdot 0.029)}{8,314 \cdot 0.0065} \approx 5.255$

Ensuite on déplace les termes de l'équation pour obtenir la hauteur en fonction de la pression :

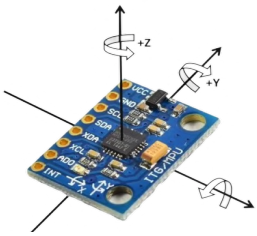
$$\frac{P}{P_0} = \left(1 - \frac{L \cdot h}{T_0}\right)^{5.255} <=> \left(\frac{P}{P_0}\right)^{\frac{1}{5.255}} = 1 - \frac{L \cdot h}{T_0} <=> \frac{L \cdot h}{T_0} = 1 - \left(\frac{P}{P_0}\right)^{\frac{1}{5.255}} <=> h = \frac{T_0}{L} \cdot \left[1 - \left(\frac{P}{P_0}\right)^{\frac{1}{5.255}}\right]$$

Maintenant que l'on a la formule pour la hauteur en fonction de la pression il nous suffit grâce au capteur d'obtenir la pression pour calculer la hauteur tous les Δt avec l'ordinateur de bord. On peut alors exploiter ces données durant le vol pour notamment commander l'ouverture du parachute à l'apogée.

Concernant la précision du capteur, le constructeur du composant indique une précision à ±

0.12 hPa (± 1m), ce qui représente de très grandes incertitudes de mesure. À l'échelle de la fusée, ce manque de précision peut être acceptable dans la mesure où ce que compte à nos yeux est que l'apogée surtout soit détectée pour amorcer l'ouverture du parachute.

Figure 21 : Le capteur MPU6050.



Le MPU6050 (représenté ci-dessus en *Figure 21*) est un capteur qui peut jouer les rôles d'accéléromètre et de gyroscope. Il permet donc de mesurer l'accélération sur 3 axes et la vitesse de rotation autour de 3 axes (ces axes sont représentés sur la *Figure 21*).
Nous pouvons donc l'utiliser en tant qu'accéléromètre pour mesurer l'altitude.
Tout d'abord il faut utiliser le capteur de façon à ce que l'axe (*Oz*) soit parfaitement perpendiculaire au sol pour que l'exploitation des données soit fiable.

Pour obtenir l'altitude on sépare les calculs en trois étapes :

Premièrement nous mesurons la composante verticale du vecteur accélération (que nous noterons a_Z), à laquelle on soustrait g, l'accélération de la pesanteur. On obtient alors l'accélération notée $a_{verticale}$: $a_{verticale} = a_Z - g$
Puis on calcule la vitesse verticale en cherchant une primitive de l'accélération $a_{verticale}$.
Pour finir, on cherche une primitive de la vitesse verticale pour obtenir la position verticale z et donc l'altitude.

Test des capteurs

Nous voulions d'abord rédiger un code pour tester les deux capteurs, en extraire la pression ou l'accélération, et obtenir *in fine* l'altitude. Cependant, après un test rapide du MPU6050, les données affichées étaient inexploitables du fait de la valeur constante affichée de l'accélération verticale. Sans variation d'accélération durant le mouvement du capteur, il nous a été impossible de détecter son altitude.

Nous nous sommes donc focalisés sur le BMP280. Nous avons procédé à son étalonnage grâce à une poulie attachée à 5 mètres de hauteur avec un câble long de plus de 10 mètres. Ce câble nous a permis depuis le sol de faire décoller la boîte contenant le BMP280. Un mètre déployé verticalement nous a permis de mesurer l'altitude réelle de la boîte pendant son mouvement (voir photos et tableau de mesures en *Annexe 6.2*).

Nous avons rassemblé les mesures de pression P et d'altitude h dans un tableau, puis tracé sur Excel le graphique $h = f(P)$ (*Annexe 6.2*). Nous nous sommes contentés par souci de simplicité de la pression en abscisse (*cf.* loi fondamentale de la statique des fluides) et non de l'expression développée en page 15. Nous observons une variation de pression environ égale à l'erreur de mesure du BMP280 pour une variation d'altitude de 1 m, ce qui amène à de très grandes incertitudes (non représentées sur le graphique). Cependant, la répartition des mesures d'altitude au mètre parmi celles issues du capteur (notamment à partir de $h > 2$ m) nous semble très encourageante pour la suite. De plus, qualitativement, on observe une pression qui diminue quand l'altitude augmente, ce qui est cohérent.

Ce sont donc les données issues de ce capteur que nous avons décidé d'utiliser au sein de l'ordinateur de bord pour mesurer l'altitude de la fusée durant son mouvement.

4.3.3. Code de l'ordinateur (original en *Annexe 6.3.*)

Voici le cahier des charges pour le code de l'ordinateur

Début des enregistrements :

- Démarrer l'enregistrement des données sur la carte SD lorsque le décollage est détecté.

Calculs et mesures en vol :

- Calculer et enregistrer la hauteur de la fusée en mètres.
- Calculer et enregistrer la position de la fusée sur différents axes (en degrés).
- Enregistrer la température ambiante.
- Calculer la force G subie par la fusée et l'enregistrer.
- Déterminer et enregistrer le temps de vol.
- Identifier et enregistrer :
  - Le point de décollage.
  - L'apogée (altitude maximale).
  - Le point d'arrêt (fin du vol).

Contrôle et gestion du servomoteur :

- Activer le servomoteur à 180° :
- Normalement, 4 secondes après l'apogée.
- Par sécurité si :
  - La fusée n'est plus à la verticale (trajectoire horizontale).
  - La hauteur ne change pas pendant 10 secondes après le décollage confirmé.

Enregistrer sur la carte SD :

- Le type d'activation du servomoteur (normale ou par sécurité).
- Le moment exact de son activation.

Fin des enregistrements :

- Arrêter l'enregistrement des données 10 secondes après la détection de la fin du vol.

4.4. Aérodynamisme

4.4.1. Ogive

Pour qu'une fusée atteigne la plus grande hauteur possible, elle doit contrer la force d'attraction gravitationnelle de la Terre en générant une poussée suffisante grâce à son moteur. Cependant, en montant dans l'atmosphère à grande vitesse, la fusée rencontre une autre force opposée : la résistance de l'air. Celle-ci est due aux molécules présentes dans l'atmosphère que la fusée rencontre en progressant rapidement, ce qui freine son ascension.

Pour maximiser l'efficacité de la fusée face à la résistance de l'air, on doit concevoir un design aérodynamique. Le fuselage, qui est un tube lisse, pose peu de problème, mais la forme de la coiffe (le nez de la fusée) est cruciale. Pour réduire la résistance face à l'air, la coiffe doit avoir une forme profilée, comme celle d'une ogive. Cette forme permet de fendre l'air de manière plus efficace, réduisant la force qui freine la fusée.

On a opté pour les dimensions présentées en *Figure 22*.

Nous avons choisi des valeurs qui nous semblent logiques, mais nous nous réservons le droit de modifier les dimensions si les résultat ultérieurs de nos tests le requièrent par exemple.

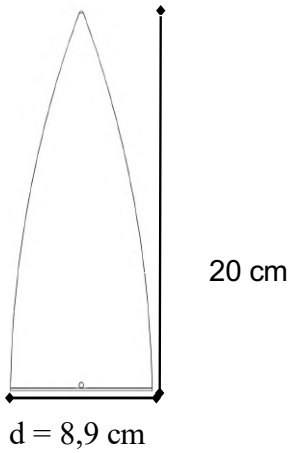


Figure 22 : Dimensions de l'ogive.

4.4.2. Ailerons

Nous avons décidé d'intégrer des ailerons à notre fusée, car ils jouent un rôle essentiel dans la stabilité et la direction en vol. Leur position et leurs dimensions ont été choisies de manière logique pour optimiser l'aérodynamisme (*Figure 23*). Cependant, nous nous réservons comme pour l'ogive le droit d'ajuster leur taille ou leur orientation si les tests ou une analyse approfondie montrent qu'une amélioration est nécessaire.

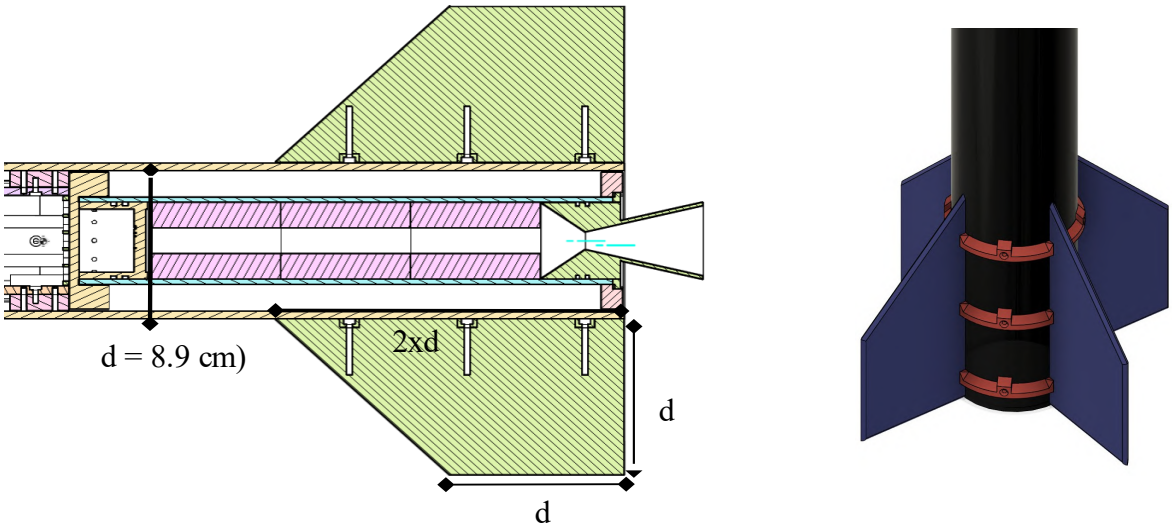


Figure 23 : Détail des dimensions des ailerons (à gauche) et représentation simulée en 3D des 4 ailerons à la base de la fusée.

5. Conclusion et perspectives

5.1. Conclusion

Nous sommes très fiers de présenter notre projet NOVA en version papier. Nous ne sommes qu'à une étape du projet, qui n'est pas près de s'arrêter.
Nous n'avons en effet pas encore fait décoller la fusée. Nous misons sur de prochains événements tels que le C'space qui se déroule en juillet prochain pour réaliser le décollage.

En terme d'expérience, ce projet nous a permis d'en acquérir beaucoup et dans de nombreux domaines, notamment dans la microélectronique, l'aérodynamisme, l'ingénierie, l'utilisation de différents logiciels (conception 3D, conception de circuits imprimés...).
Comme expliqué précédemment nous n'avons pas encore terminé le projet. Les objectifs de ce mémoire ne sont évidemment pas ceux du projet en entier. En effet nous avons d'autres objectifs comme :

- Créer un moteur original avec un carburant solide. Nous sommes conscients du danger associé à la construction de ce genre de moteur (flammes, pression). C'est pourquoi nous avons préféré ne pas le faire pour ce concours. Nous y avons quand même songé, le problème étant le temps de réalisation et de compréhension du sujet.
- Si le système de récupération ne fonctionne pas, nous avons comme idée d'ajouter des volets rétractables ou alors revenir à une idée de trappe qui s'ouvre sur le côté.
- Repenser ARC pour corriger les erreurs actuelles (qui ne compromettent pas le bon fonctionnement de l'ordinateur) ou alors de le miniaturiser en repensant entièrement l'ordinateur.

5.2. Perspectives professionnelles

Nous sommes très heureux de faire ce projet qui décrit bien nos aspirations professionnelles. Pouvoir mettre un cadre autour de tout ce que l'on a fait récemment, professionnalise d'une certaine façon ce projet et permet de le marquer à vie comme un réel projet d'ingénierie à notre sens.

6. Sitographie

https://www.astromodelisme.com/ailettes/ (consulté le 23/11/2024, 15h00)
Quentin, Ailerons, 13 mars 2015
https://www.grc.nasa.gov/www/k-12/VirtualAero/BottleRocket/airplane/rktvrecv.html
(consulté le 23/11/2024, 15h00)
Tom Benson, Parachute, "Velocity during recovery"
http://clap54b.free.fr/microfusees/4parachute.htm (consulté le 23/11/2024, 15h00)
https://www.nakka-rocketry.net/ (consulté le 23/11/2024, 15h00)
On a utilisé de nombreuse pages de ce site
Richard Nakka, *Experimental Rocketry* Website, July 1997
Tout au long du projet (pour le code surtout)
Open AI, IA generative, 30 novembre 2022
https://openai.com/index/gpt-4/
Au début du projet, Microsoft IA de génération d'image ;
https://www.bing.com/images/create
Site de Circuit imprimé :
https://jlcpcb.com/
Info sur le C'space ;
https://cnes.fr/actualites/cspace-un-nombre-de-lancements-record-ledition-2024

OLYMPIADES DE PHYSIQUE FRANCE

Annexe du Mémoire

Titre du projet :

Construction d'une fusée de A à Z

Élaboré par :

Enzo Sanchez Sartori

&

Elliot Matton–Marie

Avec l'aide du professeur :

Nicolas Debons

Sommaire

Annexe 1 – Projets personnels

Annexe 1.1 – Projets d'Enzo

Le projet de voiture radiocommandée d'Enzo a pour nom « Pulsar ». Plusieurs photos de ce projet ainsi que le logo du nom du projet sont présentés en *Figure 1*.

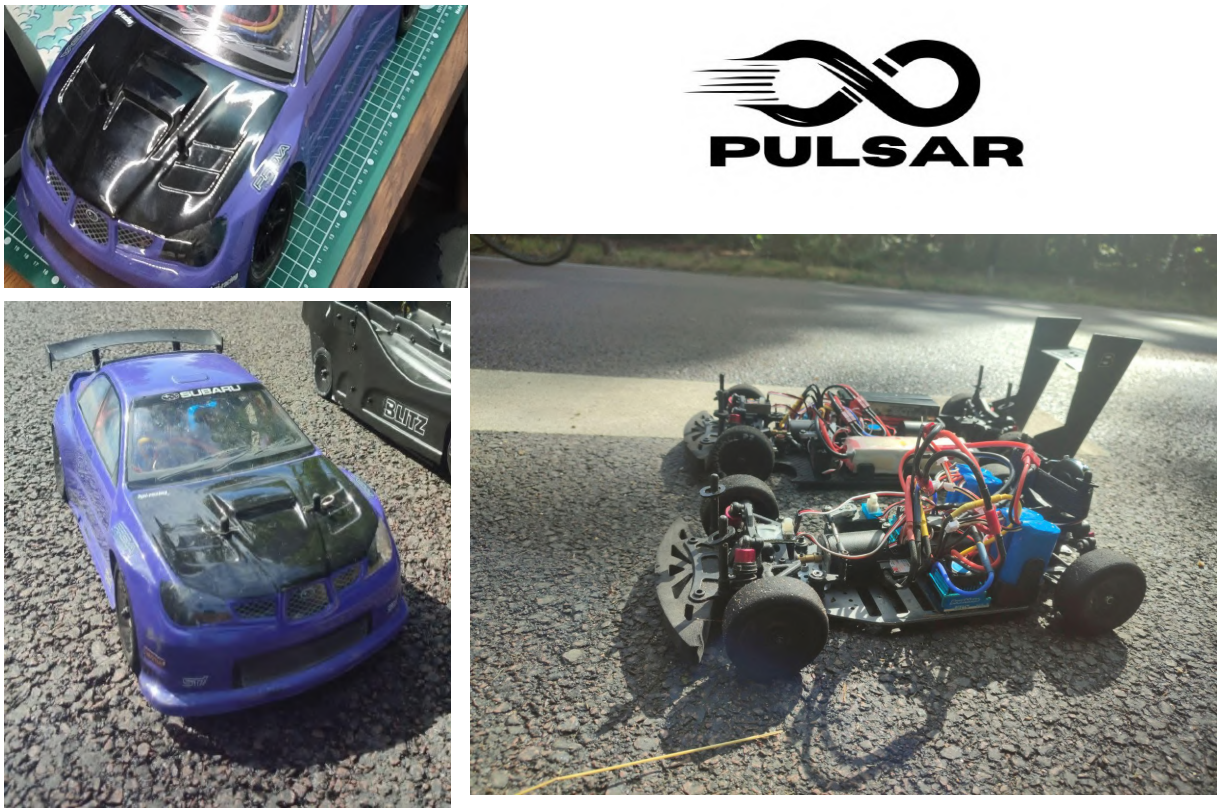


Figure 1 : Trois photos de la voiture radiocommandée développée par Enzo, ainsi que le logo du projet « Pulsar ».

La vitesse maximale atteinte par cette voiture à ce jour est de 70 km/h. Le but est d'atteindre la vitesse symbolique des 100 km/h, en optimisant l'espace occupé et en conservant les dimensions d'origine.

Le circuit électrique est composé de 8 cellules électriques destinées à donner 14.8V. Il y a également deux moteurs qui tournent à 50.000 tours minute théoriquement. La vitesse de 100 km/h est *a priori* dépassée mais les tests sont compliqués à réaliser à de telles vitesses.

Enzo a également réalisé plusieurs petits projets trop courts pour être expliqués :
- Modélisation 3D d'un bras de robot (première image de la *Figure 2*) ;
- Restauration de voiture 1/24 ;
- Réaménagement de chambre en la modélisant en 3D (deuxième image de la *Figure 2*) ;

- Construction d'une propre tour PC (avec l'aide d'Elliot) ;
- Réalisation d'une « Trokart », mélange entre une voiture de karting et une trottinette électrique (troisième image de la *Figure 2*) ;
- Restauration d'une Pocket bike (une très petite moto).

<u>Figure 2 :</u> Trois images de mini-projets menés par Enzo. Image de gauche : modélisation 3D d'un bras de robot. Image centrale : modélisation 3D d'une chambre avec notamment le bureau. Image de droite : photo d'une « Trokart ».

<u>Annexe 1.2 – Projet d'Elliot</u>

Le projet de voiture radiocommandée d'Elliot a pour nom « Endurance » car il y a besoin d'endurance financière et d'endurance mentale dans ce genre de projet.

Elliot a réalisé sa voiture en agrandissant le châssis pour accueillir plus de composants. Il a dû modéliser un nouveau châssis en fibres de carbone. Il a par la suite commandé une plaque en fibres de carbone qu'il a découpée et percée.

Nous sommes tous les deux en pause sur ce projet de voiture radiocommandée. Nous reprendrons les tests une fois les conditions expérimentales plus favorables, donc en printemps/été (il faut notamment un sol un minimum chaud pour améliorer l'adhérence).

<u>Figure 3 :</u> Image du projet d'avion motorisé en bois mené par Elliot.

Annexe 1.3 – Explication du logo de notre projet

Enzo utilise le nom de « Pulsar » car c'est une étoile à neutrons qui tourne extrêmement rapidement sur elle-même. Ses projets ont souvent un lien avec la vitesse et Enzo aime beaucoup le symbole comme le nom (voir *Figure 4*).

Elliot utilise le nom de « AEN » qui est l'acronyme de « Air, Endurance, Nova ». Ce sont les noms de ses trois plus gros projets à ce jour. A pour Air, un avion motorisé en balsa (type de bois très léger) de 2 mètres d'envergure ; E pour Endurance, la voiture radiocommandée ; N pour NOVA (voir *Figure 4*).

Figure 4 : Logo de notre projet NOVA : « Construction d'une fusée de A à Z ».

Annexe 2 – Nova 01 tiny

Cette mini-fusée entièrement réalisée par impression 3D a été réalisée dans un premier temps pour nous initier au domaine de la fuséologie. Elle s'appelle Nova 01 tiny (voir *Figure 5*). C'est pourquoi nous souhaitions un moment que la fusée qui est l'objet de ce projet soit nommée Nova 2E ou Nova EE (deuxième tentative de réalisation du projet de construction de fusée).

<u>Figure 5 :</u> Photo de notre première fusée NOVA 01 parée au décollage.

Annexe 3 – Problèmes rencontrés

Annexe 3.1 – Avec le système de récupération

Nous avons rencontré les problèmes suivants durant la modélisation du système de récupération :

- La Base une fois imprimée, nous nous sommes rendu compte que ses dimensions avaient des valeurs ne permettant pas le bon emboîtement des pièces au niveau du rail en métal. Nous avons donc dû apporter des modifications et la réimprimer.
- la résistance mécanique de la partie rouge de la Base était faible donc nous avons dû apporter de nouveau des modifications avant de la réimprimer.
- La partie Amovible (en vert) était sujette à la flexion vers l'intérieur ce qui empêchait les *triggers* de bien s'enlever. Nous avons donc dû créer une nouvelle pièce pour empêcher le mouvement des pièces de la partie Amovible.

Nous avons fait des tests de résistance mécanique en accrochant la totalité du système assemblé à une masse. Cela nous a permis de tester avec succès la bonne jointure du système (voir *Figure 6*).

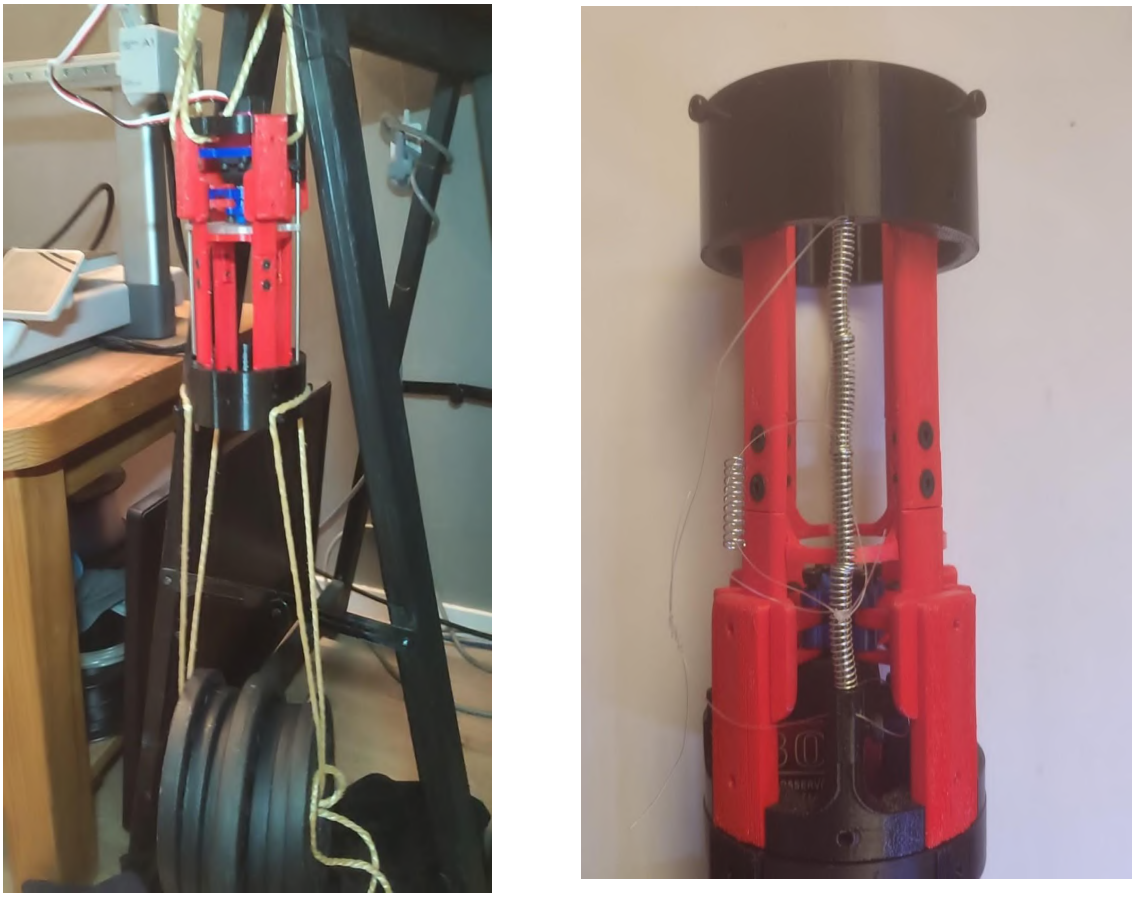


Figure 6 : Photos du système de récupération final.

Annexe 3.2 – Avec le tiroir

La première version de fermeture du tiroir comprenait un ressort qui permettait à la poignée de rester collée à son socle. Lorsqu'il fallait tourner la poignée, on devait d'abord la tirer vers le haut, ce qui compressait le ressort. Ce système apportait des problèmes au niveau de l'ouverture car on s'est aperçus que le tiroir ne s'ouvrait plus.

Cette idée a donc été écartée pour finalement conserver le système sans le ressort.

Annexe 4 - Expérience du lancer de parachute

Figure 7 : Image extraite de la vidéo obtenue lors de l'expérience de lancer de parachute.

Après avoir construit notre parachute nous avons testé sa capacité à se déployer lors de la phase de chute pour assurer le freinage de la fusée et éviter de casser la fusée.

C'est dans cette optique que l'on a réalisé une expérience dans le gymnase de notre lycée. C'est dans ce lieu que nous avons trouvé la plus grande hauteur (7 m) pour réaliser la chute. Cette expérience nous a permis de valider l'ouverture du parachute et sa forme hemisphérique lors de la chute (voir *Figure 7*).

Nous aurions aimé pouvoir exploiter la vidéo et obtenir à l'aide d'une chronophotographie la vitesse maximale atteinte par le système {parachute + masse} au cours de la chute. Malheureusement la hauteur n'était pas suffisante pour que le système atteigne une vitesse constante. Nous aimerions prochainement pouvoir réaliser à nouveau cette expérience avec une hauteur de chute plus grande.

Annexe 5 – Matériel, consommables et logiciels

Annexe 5.1 – Matériel

Outils :
Pied à coulisse / règle/ tournevis / clé à laine / balance / calculatrice / papier de verre / visseuse / cutter / scie / multimètre / fer à souder.

Machines :
Ordinateur fixe et portable / BambuLab A1 (imprimante 3D) / imprimante papier.

Annexe 5.2 – Consommables

Quincaillerie :
Vis M3 (de 5mm à 30mm) / ressort.

Matériaux :
PETG / carton / aluminium / composants électroniques.

Annexe 5.3 – Logiciels

Pour la 3D “Fusion 360” :
https://www.autodesk.com/ca-fr/products/fusion-360/overview?term=1-YEAR&tab=subscription

Pour l’ordinateur (ARC) “Easy EDA” :
https://easyeda.com/fr

Pour créer les fichier lisibles par l'imprimante 3D “Bambu studio” :
https://bambulab.com/en/download/studio

Pour conception théorique d’une fusée “Openrocket” :
https://openrocket.info/

Pour le code “Visual studio” :
https://code.visualstudio.com

Annexe 6 – Ordinateur de bord

Annexe 6.1 – Schéma papier des connexions

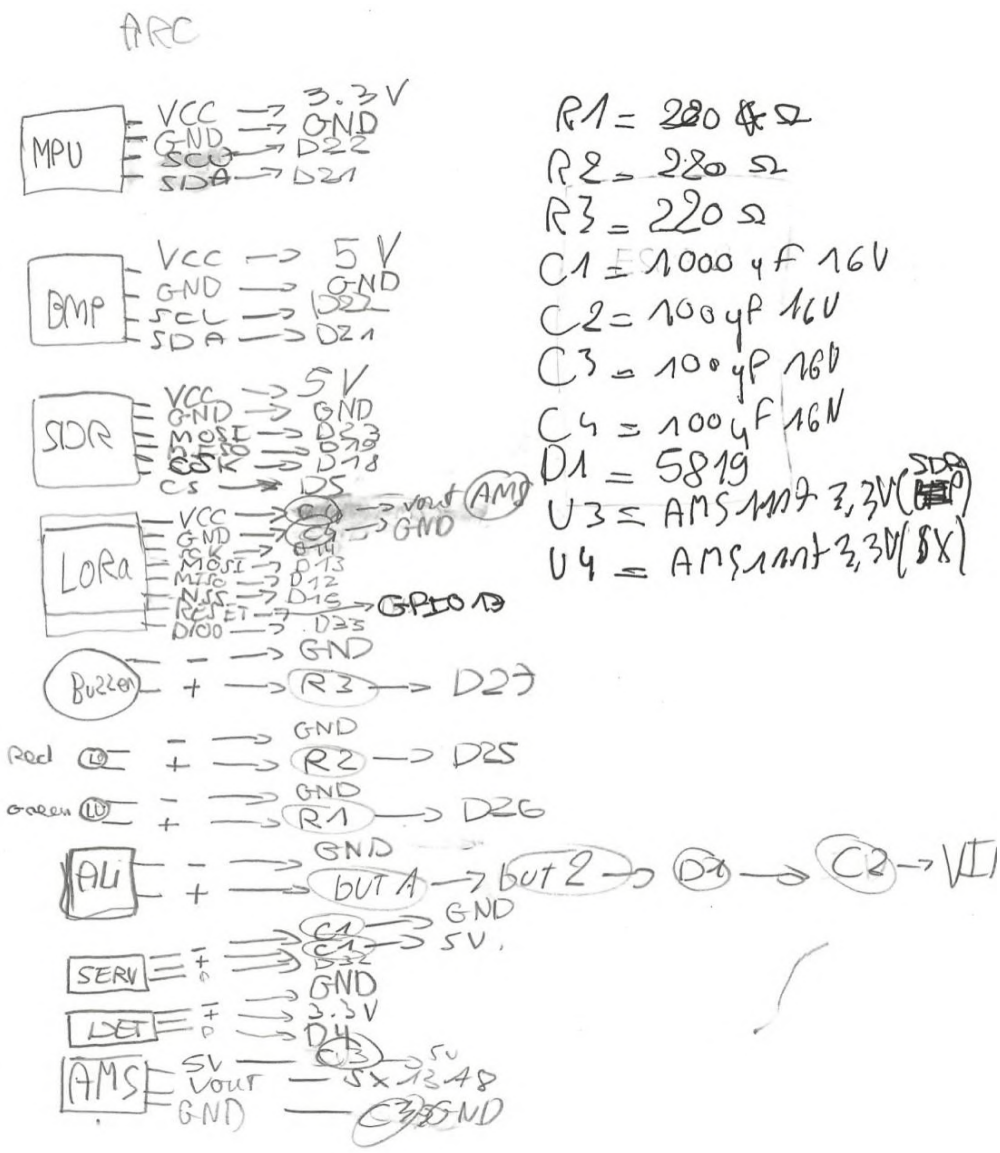
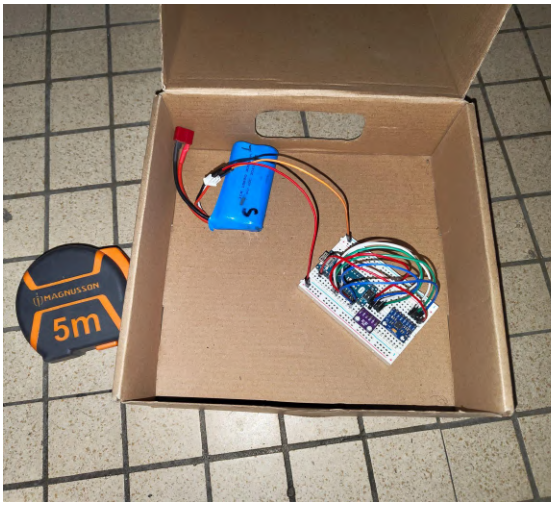
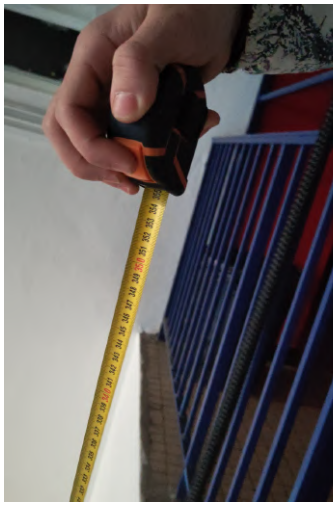


Figure 8 : Schéma papier des connexions de l'ordinateur de bord ARC.

Annexe 6.2 - Expérience de l'étalonnage du BMP280





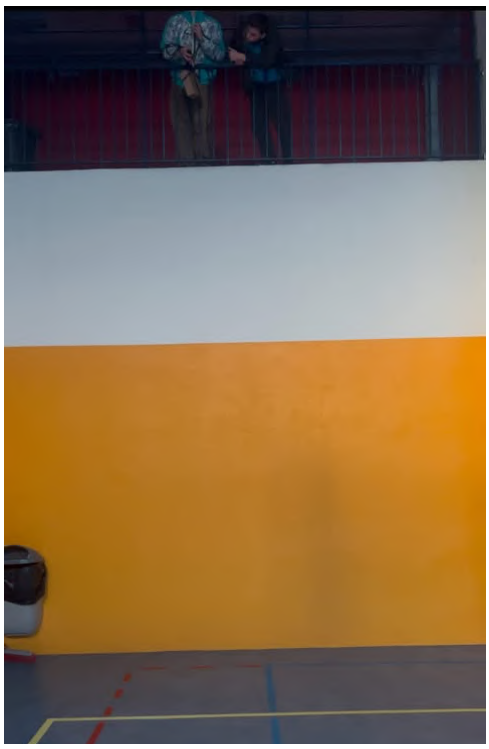
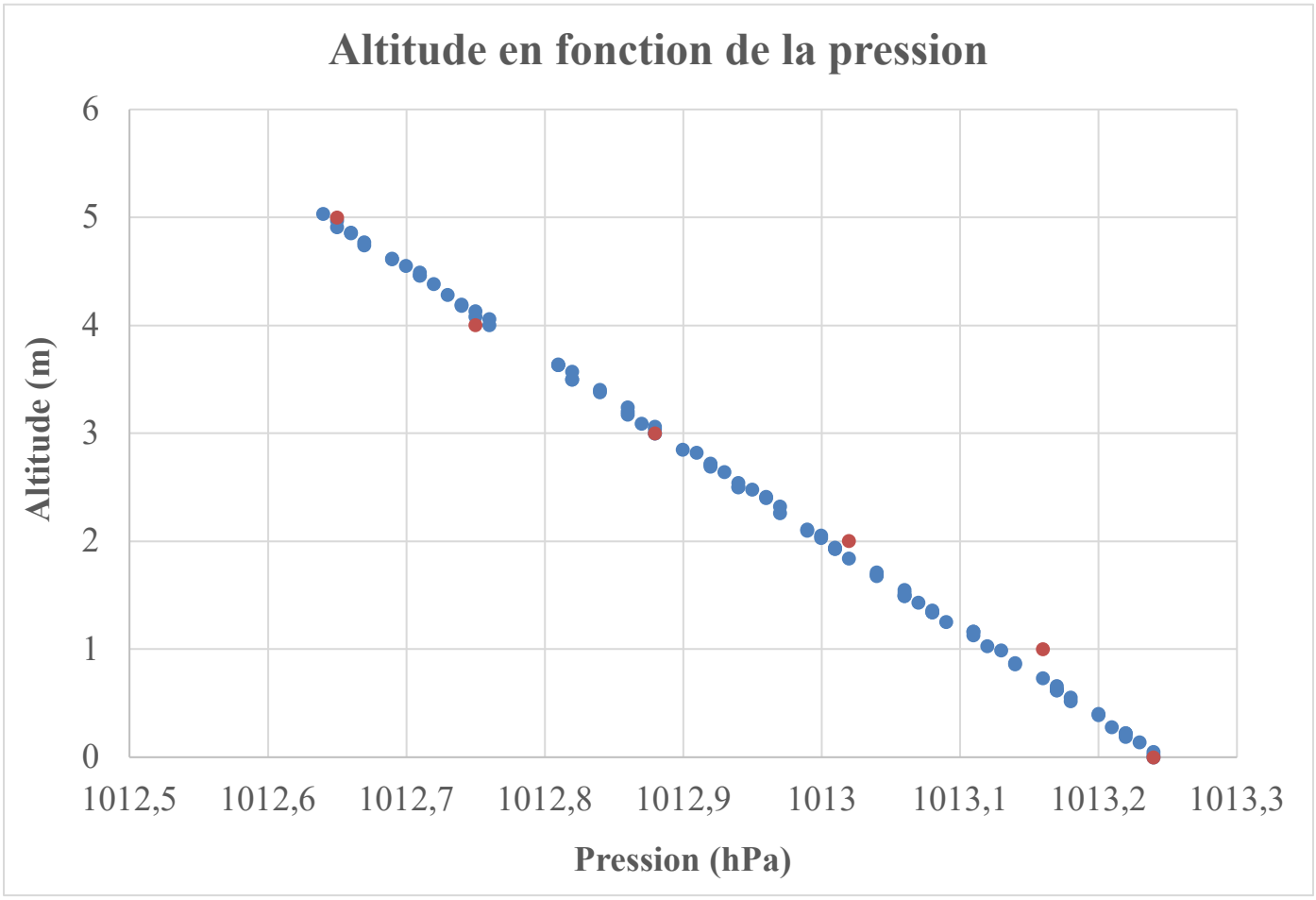


Figure 9 : Photos prises lors de l'étalonnage du capteur barométrique, montrant la boîte contenant le BMP280, le mètre et la hauteur considérée au gymnase du lycée.

Temps (s)	Pression (hPa)	Altitude (m)	Altitude reel
0.0	1013.24	0.02	0
0.1	1013.24	0.05	
0.2	1013.24	0.0	
0.3	1013.22	0.19	
0.4	1013.22	0.22	
0.5	1013.22	0.22	
0.6	1013.2	0.4	
0.7	1013.21	0.28	
0.8	1013.23	0.14	
0.9	1013.2	0.39	
1.0	1013.18	0.52	
1.1	1013.18	0.55	
1.2	1013.17	0.63	
1.3	1013.17	0.66	
1.4	1013.17	0.62	
1.5	1013.17	0.62	
1.6	1013.16	0.73	1
1.7	1013.13	0.99	
1.8	1013.14	0.86	
1.9	1013.12	1.03	
2.0	1013.14	0.87	
2.1	1013.11	1.16	
2.2	1013.11	1.13	
2.3	1013.11	1.16	
2.4	1013.09	1.25	
2.5	1013.08	1.34	
2.6	1013.08	1.34	
2.7	1013.08	1.36	
2.8	1013.07	1.43	
2.9	1013.06	1.49	
3.0	1013.06	1.52	
3.1	1013.06	1.49	
3.2	1013.06	1.5	
3.3	1013.06	1.55	
3.4	1013.04	1.71	
3.5	1013.04	1.69	
3.6	1013.04	1.68	
3.7	1013.02	1.84	2
3.8	1013.01	1.93	
3.9	1013.01	1.93	
4.0	1013.01	1.94	
4.1	1013.0	2.03	
4.2	1012.99	2.1	
4.3	1013.0	2.05	
4.4	1012.97	2.32	
4.5	1012.99	2.11	
4.6	1012.96	2.41	
4.7	1012.97	2.26	
4.8	1012.94	2.5	
4.9	1012.96	2.4	
5.0	1012.95	2.48	
5.1	1012.94	2.54	
5.2	1012.94	2.5	
5.3	1012.93	2.64	
5.4	1012.92	2.71	
5.5	1012.92	2.69	
5.6	1012.92	2.72	
5.7	1012.91	2.82	
5.8	1012.9	2.85	
5.9	1012.88	3.0	
6.0	1012.88	3.06	
6.1	1012.88	3.04	
6.2	1012.86	3.2	
6.3	1012.88	3.0	3
6.4	1012.87	3.09	
6.5	1012.86	3.24	
6.6	1012.86	3.17	
6.7	1012.84	3.38	
6.8	1012.82	3.5	
6.9	1012.84	3.4	
7.0	1012.82	3.5	
7.1	1012.81	3.63	
7.2	1012.81	3.63	
7.3	1012.82	3.57	
7.4	1012.81	3.64	
8.0	1012.76	4.06	
8.1	1012.75	4.13	4
8.2	1012.75	4.08	
8.3	1012.74	4.18	
8.4	1012.76	4.0	
8.5	1012.74	4.19	
8.6	1012.71	4.46	
8.7	1012.73	4.28	
8.8	1012.72	4.38	
8.9	1012.71	4.49	
9.0	1012.7	4.55	
9.1	1012.71	4.46	
9.2	1012.69	4.61	
9.3	1012.69	4.62	
9.4	1012.67	4.77	
9.5	1012.67	4.74	
9.6	1012.66	4.85	
9.7	1012.66	4.86	
9.8	1012.65	4.91	
9.9	1012.65	4.97	5
10.1	1012.64	5.03	
10.2	1012.66	5.1	
10.3	1012.63	5.05	
10.4	1012.64	5.12	

Figure 10 : Mesures (temps, pression, altitude) obtenues *via* le BMP280 lors de l'étalonnage du capteur barométrique. Sont également figurées les mesures réelles d'altitude réalisées à l'aide du mètre.



<u>Figure 11 :</u> Graphique montrant l'évolution de l'altitude en fonction de la pression. Les points bleus correspondent aux mesures expérimentales réalisées avec le capteur barométrique, et ceux rouges correspondent aux altitudes mesurées à l'aide du mètre.

Annexe 6.3 – Code C++

```
#include <Wire.h>
#include <SD.h>
#include <SPI.h>
#include <ESP32Servo.h> // Utiliser la bibliothèque ESP32Servo
#include <Adafruit_MPU6050.h>
#include <Adafruit_BMP280.h>
#include <Adafruit_Sensor.h>

// Définitions des broches
#define LED_ROUGE_PIN 25
#define LED_VERTE_PIN 26
#define BUZZER_PIN 27
#define SERVO_PIN 32
#define SD_CS_PIN 5
#define USB_DETECT_PIN 4

// Initialisation des capteurs et composants
Adafruit_MPU6050 mpu;
Adafruit_BMP280 bmp;
Servo myServo;
File dataFile;

void detecterProblemes() {
// Vérifier la carte SD
if (!SD.begin(SD_CS_PIN)) {
Serial.println("Erreur de carte SD !");
digitalWrite(LED_ROUGE_PIN, HIGH);
digitalWrite(LED_VERTE_PIN, LOW); while
(1);
}

// Vérifier le capteur BMP280
if (!bmp.begin()) {
Serial.println("Erreur de capteur BMP280 !");
digitalWrite(LED_ROUGE_PIN, HIGH);
digitalWrite(LED_VERTE_PIN, LOW);
while (1);
}

// Vérifier le capteur MPU6050
if (!mpu.begin()) {
Serial.println("Erreur de capteur MPU6050 !");
digitalWrite(LED_ROUGE_PIN, HIGH);
digitalWrite(LED_VERTE_PIN, LOW);
while (1);
}

// Si tout va bien
digitalWrite(LED_VERTE_PIN, HIGH);
digitalWrite(LED_ROUGE_PIN, LOW);
}

void enregistrerDonneesDeVol() {
unsigned long startTime = millis();
bool volTermine = false;
bool apogeeAtteinte = false;
float hauteurMax = 0;
```

```
                while (!volTermine) {
            unsigned long currentTime = millis();
             float hauteur = bmp.readAltitude();
             float temp = bmp.readTemperature();
              sensors_event_t a, g, tempEvent;
             mpu.getEvent(&a, &g, &tempEvent);

                // Enregistrement des données
          dataFile.print(currentTime - startTime);
                   dataFile.print(",");
                 dataFile.print(hauteur);
                   dataFile.print(",");
           dataFile.print(a.orientation.pitch);
                   dataFile.print(",");
           dataFile.print(a.orientation.roll);
                   dataFile.print(",");
         dataFile.print(a.orientation.heading);
                   dataFile.print(",");
                  dataFile.print(temp);
                   dataFile.print(",");

               // Détection de l'apogée if
                 (hauteur > hauteurMax) {
                    hauteurMax = hauteur;
                  apogeeAtteinte = false;
} else if (!apogeeAtteinte && hauteur < (hauteurMax - 5)) {
                   apogeeAtteinte = true;
                 dataFile.print("Apogee");
          myServo.write(180); // Activer le servomoteur
         tone(BUZZER_PIN, 1000, 100); // Bip à l'apogée
     delay(4000); // Attendre 4 secondes avant l'activation
       myServo.write(0); // Réinitialiser le servomoteur
                                }

               // Vérification de la fin du vol
            if (hauteur < 1 && apogeeAtteinte) {
                        volTermine = true;
 tone(BUZZER_PIN, 1000, 3000); // Bip continu à l'atterrissage
                      while (true) {
     tone(BUZZER_PIN, 1000, 1000); // Bip toutes les 3 secondes
                          delay(3000);
                             }
                            }

          // Enregistrement de l'activation du servomoteur
                      if (apogeeAtteinte) {
              dataFile.println("Servo activated normally");
                          } else {
                dataFile.println("Servo not activated");
                            }

                // Enregistrement des données
                          dataFile.flush();
     delay(1000); // Enregistrement toutes les secondes
                          }

             // Fin de l'enregistrement des données
                          dataFile.close();
delay(1000); // Attendre 1 seconde après la détection de la fin du vol
                          }
```

```
                                void setup() {
                                    Serial.begin(115200);
                                pinMode(LED_ROUGE_PIN, OUTPUT);
                                pinMode(LED_VERTE_PIN, OUTPUT);
                                  pinMode(BUZZER_PIN, OUTPUT);
                                pinMode(USB_DETECT_PIN, INPUT);

                                 // Initialisation du MPU6050
                                    if (!mpu.begin()) {
        Serial.println("Impossible de trouver le MPU6050, checkez les connexions !");
                                digitalWrite(LED_ROUGE_PIN, HIGH);
                                            while (1);
                                              }

                                 // Initialisation du BMP280
                                    if (!bmp.begin()) {
         Serial.println("Impossible de trouver le BMP280, checkez les connexions !");
                                digitalWrite(LED_ROUGE_PIN, HIGH);
                                            while (1);
                                              }

                               // Initialisation du servomoteur
                                  myServo.attach(SERVO_PIN);

                               // Initialisation de la carte SD
                                  if (!SD.begin(SD_CS_PIN)) {
                    Serial.println("Impossible d'initialiser la carte SD !");
                                digitalWrite(LED_ROUGE_PIN, HIGH);
                                            while (1);
                                              }

                                digitalWrite(LED_VERTE_PIN, HIGH);
                            tone(BUZZER_PIN, 1000, 100); // Bip au démarrage

                                // Initialisation du fichier de données
                              dataFile = SD.open("datalog.txt", FILE_WRITE);
                                           if (!dataFile) {
                    Serial.println("Erreur d'ouverture du fichier de données !");
                                digitalWrite(LED_ROUGE_PIN, HIGH);
                                            while (1);
                                              }

                            // Écrire les en-têtes de données dans le fichier
        dataFile.println("Temps(ms),Hauteur(m),Position(deg),Temp(C),Activation");
                                              }

                                           void loop() {
                                          detecterProblemes();

                        // Vérifier la déconnexion USB pour détecter le décollage
                                if (digitalRead(USB_DETECT_PIN) == LOW) {
                             Serial.println("USB déconnecté, début du vol détecté.");
                                // Commencer à enregistrer les données de vol
                                        enregistrerDonneesDeVol();
                                              }
                                             }
```

Figure 12 : Code C++ de l'ordinateur de bord ARC.