Compte rendu Evolusat MK2

Équipe Cansat-MK2
Arthur Fourcart
Corentin Vigourt
Dany Jorges
Vladislav Kupin
Lucas Gilbert
Célian Aldehuelo-Mateos

Année 2025

Table des matières

Présentation

1.1 Association

Evolutek est l'association de robotique du groupe IONIS, répartie sur les trois écoles : EPITA, EPITECH et IPSA, et ouverte aux étudiants de tous établissements.

Elle a pour objectif d'initier les étudiants à la robotique et à la programmation embarquée de manière pratique, tout en leur permettant de mettre en œuvre les compétences acquises lors de leur cursus. Pour cela, différents projets en robotique, électronique et programmation embarquée sont conçus et réalisés par les membres de l'association.

Cette année, Evolutek a choisi de poursuivre ses projets dans le secteur de l'aérospatial avec sa seconde participation au concours CanSat. Les objectifs de cette année étaient de réaliser et de fiabiliser une première version de CanSat d'un volume de 33 cl, de réduire, par conséquent, la taille de notre contrôleur de vol tout en préservant sa multifonctionnalité, et de réaliser le portage du *runtime* pour la famille de cartes STM32G4.

1.2 Equipe

Pour cette seconde participation, notre équipe est composée de six membres d'Evolutek, ainsi que de Cosmo-Banane, mascotte de notre projet :

Arthur Fourcart


Corentin Vigourt


Dany Jorges


Vladislav Kupin


Célian Aldehuelo-Mateos


Lucas Gilbert


Cosmo-Banane

1.3 Sponsors

La réalisation de ce projet n'aurait pas été possible sans l'aide des nombreuses entreprises et écoles, qui nous ont apporté leur soutien financier et matériel. Voici leurs noms :

AdaCore

RS

Imprimante 3D France

STMicroElectronics

Würth Elektronik

Lucas Gilbert

EPITA

EPITECH

IPSA

Missions

2.1 Mission principale

2.1.1 Séparation

Cette année, le premier objectif de la mission principale était de réaliser une séparation à une hauteur comprise entre 40 et 60 mètres.

Pour atteindre cet objectif, nous avons conçu un système de bague libératrice. Le principe est très simple : une bague effectue un mouvement rotatif autour l'axe central du CanSat à l'aide d'un servomoteur et, lorsque les fixations s'alignent avec les trous de la bague, la partie inférieure est libérée.

2.1.2 Transmission

Le second objectif était d'assurer la transmission des données d'altitude, ainsi qu'une mesure atmosphérique quelconque vers une station au sol.

Pour atteindre cet objectif, nous avons choisi de transmettre les données de nos capteurs via le protocole LoRa jusqu'à notre station au sol

2.2 Missions secondaires

2.2.1 Rétressisement

Le but de cette mission était de réduire le volume maximal du CanSat au format de 33 cl. Cette mission était très intéressante et présentait un véritable défi à la fois mécanique et électronique.

Afin de respecter le volume imposé, nous nous sommes basées sur les standards de taille des canettes vendues sur le marché. Il a donc fallu miniaturiser tout l'intérieur du CanSat, car, malgré une construction simple, le mécanisme de la séparation allait prendre beaucoup de place et il fallait également pouvoir intégrer un contrôleur de vol dans chacune des parties du CanSat.

2.2.2 Instrumentation cassée

Cette mission supposait qu'un instrument à bord, en particulier le baromètre, était indisponible, suite à une défaillance potentielle. Il fallait donc pouvoir déterminer l'altitude du

CanSat à partir d'un autre capteur.

Dans ce but, nous avons envisagé l'utilisation d'un IMU afin d'estimer l'altitude du CanSat (sachant la hauteur initiale environ égale à 120 mètres et en intégrant deux fois l'accélération verticale) pour lancer la séparation au bon moment. Néanmoins, nous avions prévu d'utiliser tout de même un baromètre, ainsi qu'un GPS dans le seul but de pouvoir comparer nos résultats après le vol. Finalement, cette mission a été abandonnée, afin de privilégier un meilleur contrôle et d'obtenir de meilleures données durant le vol.

2.2.3 Distance

Cette mission supposait l'intégration d'un système permettant de mesurer les distances entre les deux parties de CanSat après leur séparation.

Nous avons commencé par prototyper plusieurs solutions en nous interdisant l'utilisation du GPS ou du baromètre, afin de valider la mission "Instrumentation cassée". L'idée qui se concrétise était d'exploiter les IMU présents dans sur les deux cartes électroniques des deux parties. Sachant la position au moment de la séparation, nous pouvions estimer la position de la partie inférieure, puis estimer une distance entre ces deux parties dans l'air. Après de l'abandon de la mission précédente, nous avons décidé d'utiliser le GPS et les baromètres respectifs de nos deux parties du CanSat.

2.2.4 Mission libre

Cette année, notre idée de la mission libre était de reprendre la mission de l'année précédente "hisser les couleurs".

Pour cette mission, nous élevons un petit drapeau une fois que le CanSat réalise sa séparation. En plus de compléter cette mission, cela nous a permis de valider visuellement le comportement de notre CanSat.

CanSat

3.1 Architecture mécanique

Comme nous l'avons prévu l'année précédente, nous sommes partis de la base réalisée l'année dernière et nous l'avons adapté aux consignes de cette année.

Cette architecture mécanique est composée de six éléments : la structure centrale, la coiffe du haut, le capot avant, le capot arrière, la coiffe du bas et le parachute.

3.1.1 Structure centrale

La structure centrale (fig. 3.1) représente le squelette de notre CanSat. C'est cette structure qui va porter la batterie, le contrôleur de vol, le module LoRa, le servomoteur, réalisant la séparation du CanSat et ouvrant le drapeau, l'antenne LoRa nous permettant de communiquer avec la base au sol et un second baromètre.

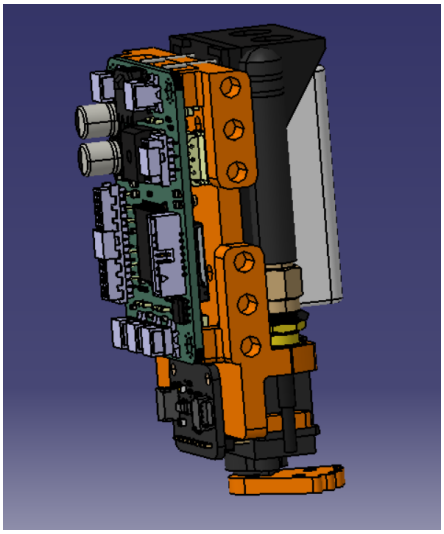

FIGURE 3.1 – Structure centrale

3.1.2 Coiffe du haut

La coiffe du haut (fig. 3.2) est la partie qui va venir faire lien entre notre parachute et notre CanSat. C'est sur celle-ci que sont placés le GPS, notre drapeau, notre système de tirette magnétique et les suspentes de notre parachute.

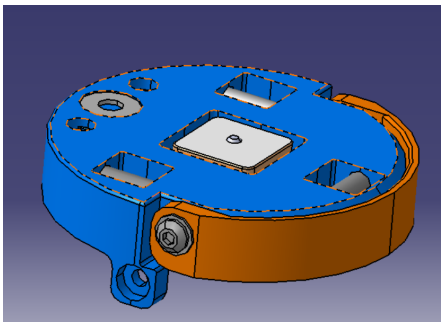

FIGURE 3.2 – Coiffe du haut

3.1.3 Capot avant

Le capot avant (fig. 3.3) vient protéger la batterie et l'antenne du CanSat et permet d'avoir une surface suffisante pour afficher tous nos sponsors.

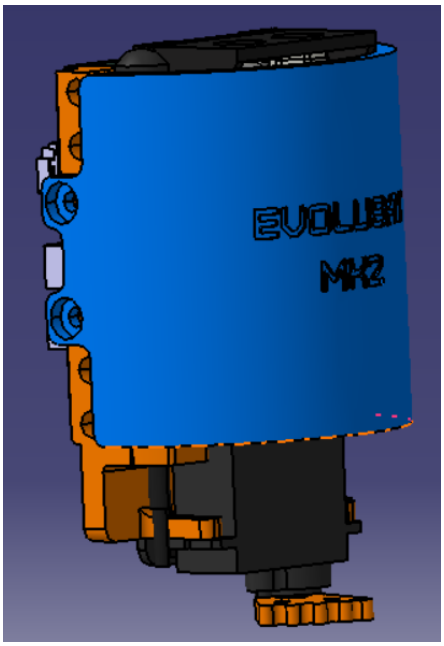

FIGURE 3.3 – Capot avant

3.1.4 Capot arrière

Le capot arrière (fig. 3.4) possède la carte d'interface, mais également de protéger le contrôleur de vol lors de la chute.

3.1.5 Coiffe du bas

Comme l'année précédente, la coiffe du bas (fig. 3.5) est la partie réservée à l'actionneur du CanSat. Pour cette année, cette coiffe était destinée à la séparation. La coiffe est constituée de quatre parties : le maintien du système, la bague tournante, la plaque de plexiglass de

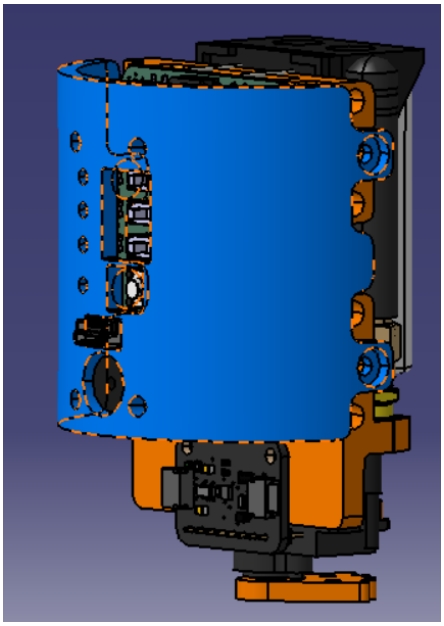

FIGURE 3.4 – Capot arrière

séparation et la partie larguée. Le système est simple : un servomoteur fait pivoter la bague jusqu'à que les supports de la partie larguée se trouvent dans les trous de la bague.

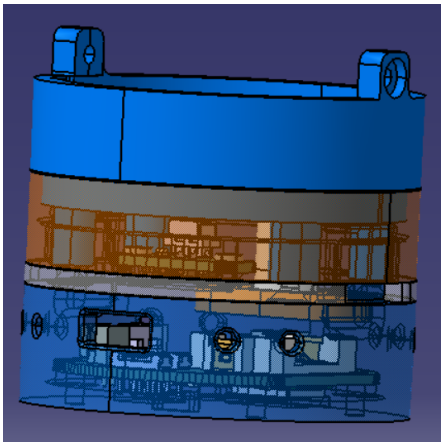

FIGURE 3.5 – Coiffe du bas

3.1.6 Parachute

Le parachute constitue la pièce la plus importante de notre CanSat, car il permet de réduire significativement la vitesse de chute. Notre parachute est confectionné à la main en polyester et compte douze suspentes regroupées en trois groupes de quatre, facilitant leur fixation à la coiffe supérieure.

3.2 Architecture électronique

Nous avons conçu au total plusieurs cartes électroniques :

— un contrôleur de vol générique, potentiellement réutilisable dans d'autres projets
— un second contrôleur de vol, destiné à la partie détachée lors de la séparation
— une carte d'entrée/sorties (IO).

Les deux contrôleurs de vol intègrent un ensemble de capteurs similaires. Nous allons vous présenter ces deux cartes.

3.2.1 Advanced Flight System (AFS)

Cette carte constitue le cerveau de l'ensemble des projets spacieux. C'est pourquoi elle est dotée d'un microcontrôleur STM32F407, offrant une puissance de calcul suffisante, ainsi qu'un large champ de possibilités de connectivité.

La carte va intégrer un baromètre (BMP390), une centrale inertielle (BNO085) un lecteur de carte SD, les trois câblés en SPI (Serial Peripheral Interface), et un ensemble de connecteurs pour des périphériques externes. Sur un de ces connecteurs vient se brancher la carte d'extension de contrôle des entrées/sorties, intégrant notamment un buzzer, plusieurs témoins d'activité (LEDs) et plusieurs interrupteurs permettant de désactiver, d'allumer le système entier et de désactiver la télémétrie, par exemple. Sur d'autres, nous allons brancher un module LoRa, pour la télémétrie, un module GPS externe (PA1616d), câblé en UART, un servomoteur en PWM pour le contrôle de la bague libératrice. Vous pouvez trouver une schématique complète ci-dessous. Ce contrôleur était alimenté par une batterie de 9V. Vous pouvez consulter son schéma sur la figure 3.6.

3.2.2 Electronical Separation Payload (ESP)

Cette deuxième carte constitue le seul composant contenu dans le module largué. Elle n'intègre que trois capteurs : un baromètre (BMP390), une centrale inertielle (BNO085) connectés en SPI, ainsi qu'un module LoRa câblé en UART. Ce système étant relativement simple, ne nécessite pas d'un microcontrôleur aussi puissant et aussi riche en connectivité que celui de l'AFS.

C'est d'ailleurs pourquoi nous avons choisi une autre famille des puces STM32, la série G4, et plus précisément la STM32G431. Ce choix a permis, par ailleurs, de réduire la consommation énergétique, ce qui était important puisque cette carte est alimentée par une toute petite batterie de 3,3 V.

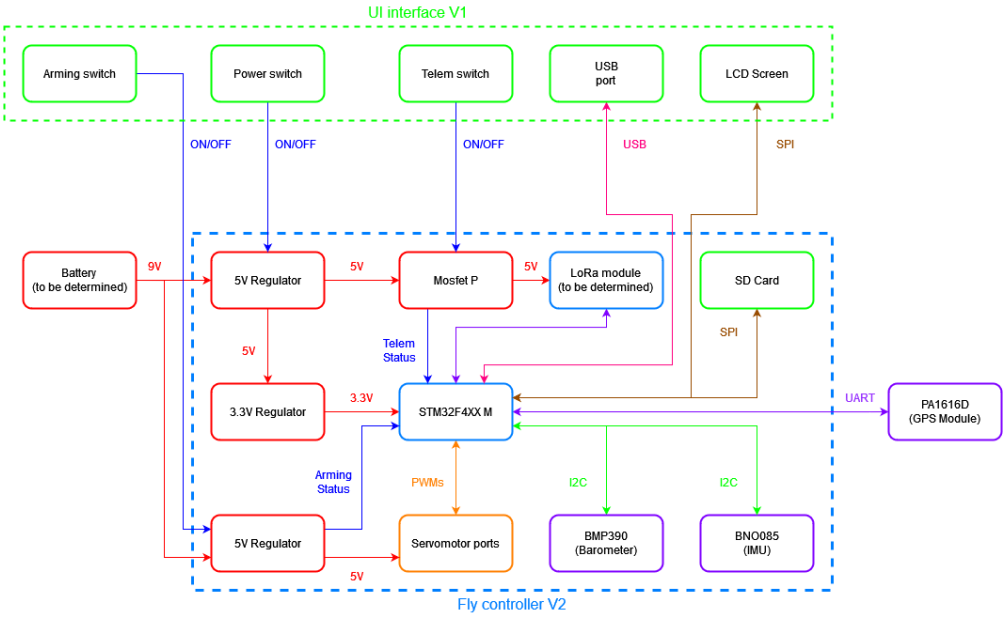

FIGURE 3.6 – Schéma de la carte d'extension de contrôle des entrées/sorties

3.3 Software

3.3.1 Langages de programmation utilisés

Étant sponsorisés par AdaCore, nous avons utilisé le langage de programmation ADA pour programmer les STM32 ce qui correspond à la majorité de notre software, mais nous avons également utilisé Python pour faire un script de réception des données en LoRa.

3.3.2 Les drivers en ADA

Cette année, nous avons pu récupérer le code des drivers écrit l'année dernière, car nous avons utilisé la même puce STM32 pour notre contrôleur de vol principal et les mêmes composants, ce qui nous a permis d'alléger une partie de travail.

Nous avons donc nos propres drivers écrits « *from scratch* » en ADA, parce qu'il n'existait pas de bibliothèque existante :

- servomoteur via PWM
- module LoRa via UART
- BNO085 via SPI
- BMP390 via SPI
- PA1616D via SPI

Ces drivers sont très spécifiques à nos projets et manquent encore de généralité.

Toutefois, cette année, nous avons essayé d'organiser mieux notre projet, c'est ainsi que nous avons créés notre propre index alire (alire étant un outil de gestion de projet en Ada), contenant tous les drivers et des runtime qui nous pourraient être utiles lors du développement.

3.3.3 Portage du runtime Ada pour la famille G4.

Le portage du runtime vers la famille STM32G4 en Ada repose principalement sur trois éléments clés :

- le SVD (*System View Description*), qui permet un accès aux registres via une interface générée en Ada,
- la HAL STM32 existante (notamment pour la famille F4),
- les *runtimes* AdaCore.

Le fichier SVD est un format standard pour les microcontrôleurs ARM, disponible sur les sites de STMicroelectronics ou ARM. Pour notre cas, le fichier `STM32G431.svd` décrit l'architecture des registres du microcontrôleur STM32G431.

Afin de générer une interface Ada à partir de ce fichier, on utilise l'outil `svd2ada`. La commande suivante permet de produire les interfaces :

```
\textbf{
svd2ada STM32G431.svd -o stm32g431 -p STM32_SVD --gen-uint-always --boolean
```

Les options `-gen-uint-always` et `-boolean` assurent une meilleure compatibilité avec les types natifs du langage Ada.

Les fichiers générés par `svd2ada`, ainsi que la HAL de la STM32F4, servent de base pour adapter la HAL à la famille STM32G4. Ce travail consiste à ajuster les fichiers et à les adapter aux spécificités du STM32G431, en s'appuyant sur le manuel de référence du microcontrôleur.

Le fichier `alire.toml` est ensuite modifié afin d'y ajouter la cible `stm32g431`. De plus, le projet `stm32_hal.gpr` est mis à jour pour inclure le nouveau cas suivant :

```
when "stm32g431" => Runtime_Device := "stm32g4";
```

Une fois cette configuration en place, chaque driver doit être adapté individuellement en tenant compte des définitions de registres spécifiques à la famille G4. Les pilotes concernés incluent, entre autres, l'ADC, le DAC, le GPIO, l'I^2C, l'USART, le SPI, les timers (TIM), le RCC, etc. L'objectif est de conserver une interface cohérente afin de permettre la réutilisation des drivers développés pour la STM32F4.

La majorité des pilotes ne nécessitent que des ajustements mineurs, les définitions des registres étant globalement similaires entre les familles F4 et G4. Néanmoins, le driver I^2C a dû être entièrement réimplémenté, en raison de différences significatives dans la structure des registres I^2C entre les deux familles.

3.3.4 La télémétrie

La télémétrie a été encodée avec un format binaire pour occuper le moins d'octets possible par transmission :

- 45 octets ont été utilisés par paquet.
- Un paquet est envoyé 2 fois par seconde.

— Toutes les mesures effectuées dans un intervalle de 0.5 seconde sont réunies en une seule à 'aide d'une moyenne.

Un paquet contient :

— Le capteur maison (vitesse de rotation)
— Le GPS (longitude, latitude)
— La pressure
— La température
— Le gyroscope (x, y, z)
— L'accéléromètre (x, y, z)

Pour recevoir la télémétrie on a utilisé une carte de développement LoRa que l'on a connecté en USB à l'ordinateur et un script Python a été utilisé pour communiquer avec la carte LoRa et recevoir puis traiter la donnée.

Toutes les données reçues sont automatiquement enregistrées dans un fichier CSV pour pouvoir les analyser ultérieurement.

Résultats

4.1 Nos vols

Durant le C'Space, nous avons pu faire deux vols de différentes hauteurs. Nous avons pu larguer notre CanSat deux fois à partir de 120 mètres.

FIGURE 4.1 – Premier vol

Le premier vol a pu démontrer le bon fonctionnement de notre canette volante. Nous avons pu observer un bon déploiement des parachutes, une séparation a bien eu lieu entre les 40 et 69 mètres (fig. 4.1) et nous avons pu obtenir suffisamment de données du vol via LoRa.

Lors du deuxième vol (bonus), nous avons largué notre CanSat, la télémétrie éteinte, comme c'est ce qui était demandé par les organisateurs. C'est ici qu'on a pu observer un des défauts de notre système de séparation, car les deux parties, malgré le bon fonctionnement du mécanisme de la bague, ne se sont séparés qu'à terre.

4.2 Problèmes identifiés

Nous avons dont identifié un des problèmes majeurs. Le mécanisme de séparation n'est pas fiable et nécessite un pliage de parachute particulier, pour que rien ne gène la libération de la coiffe inférieure.

Un autre point important que nous avons identifié avant le lancement, c'est que la pile, alimentant la partie larguée, se déchargeait trop vite, contre toutes attentes, le module LoRa consommant trop d'énergie même en mode veille. C'est pourquoi par le moment du largage du CanSat par le drone, la batterie avait déjà le temps de se décharger, c'est pourquoi nous n'avons pas pu obtenir des résultats convenables pour estimer la distance entre les deux parties du CanSat.

Le troisième défaut majeur, était le système de déclenchement, qui était trop sensible et pouvait se détache avec un mouvement léger. Nous avons décidé d'utiliser une tirette alimentée, or la puissance de l'aimant n'était pas suffisante, nous avons pu y remédier sur le moment grâce à un scotch double-face. Cette solution n'est pas du tout satisfaisante et nécessite une amélioration, une utilisation d'une prise Jack par exemple.

4.3 Les données collectées

Grâce aux données reçues par via LoRa, nous avons pu tracer quelques courbes, notamment celle de l'altitude (fig. 4.2), de pression (fig. 4.3) et de température (fig. 4.4) :

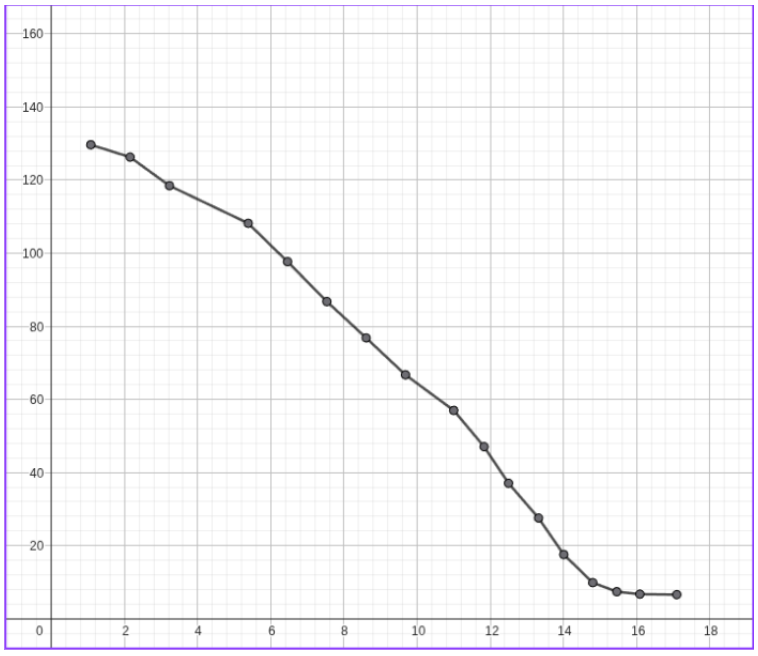

FIGURE 4.2 – Courbe de l'altitude

Grâce aux graphes de l'altitude et la pression, nous pouvons voir, que notre chute était plus ou moins à une vitesse constante de 7,1 m/s pour une durée de vol de 17 secondes. Ces

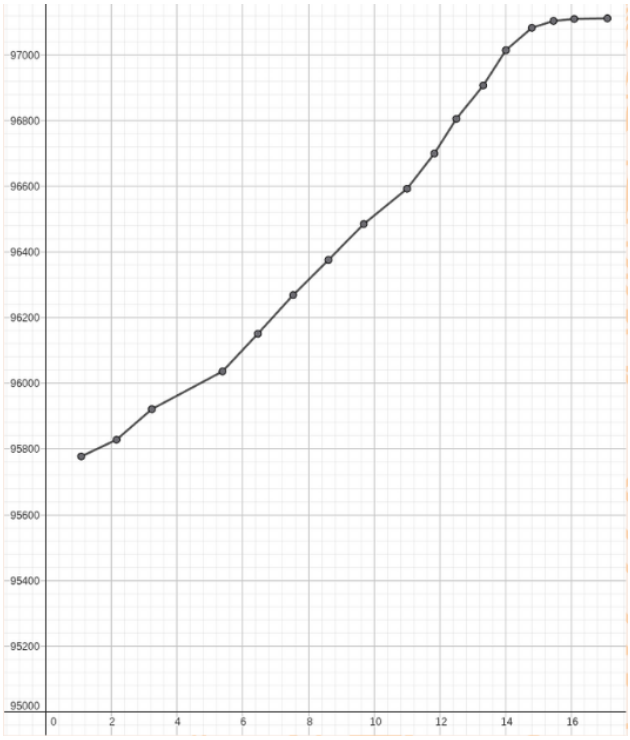

FIGURE 4.3 – Courbe de la pression

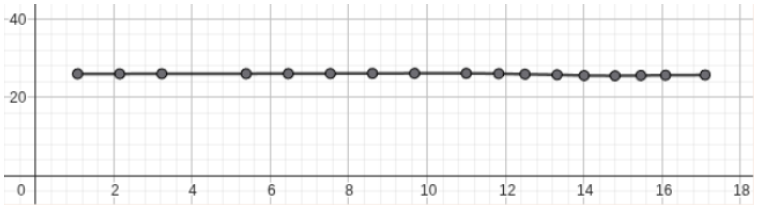

FIGURE 4.4 – Courbe de la température

données diffèrent radicalement de notre vitesse estimée de 4 m/s et une durée de vol de 30 secondes. Nous avons en déduit que la surface de notre parachute était insuffisante pour garantir une descente plus souple.

Par ailleurs, en étudiant les données de la centrale inertielle, nous avons pu constater que les données étaient trop bruitées, il faudrait donc utiliser des techniques des filtrages, comme le filtre de Kalman.

Conclusion

Pour conclure sur notre projet, nous sommes satisfaits de cette reprise du projet, malgré des nombreuses péripéties pouvant mettre fin à tout notre projet. Nous avons à présent un meilleur aperçu des contraintes que requiert la construction d'un CanSat dans le volume de 33 cl. Nous avons pu concevoir un contrôleur de vol, qui avec quelques perfections pourrait être réutilisable dans les projets des années suivantes, ainsi qu'une grande partie de la mécanique.

Nous devrions nous pencher plus sur la sûreté de fonctionnement de notre système en essayant de limiter le nombre de défaillances conséquent que nous avons eu cette année.

Le prix remporté nous motive de persévérer dans cette direction et de repartir sur une nouvelle année riche en explorations durant une nouvelle itération du concours CanSat !