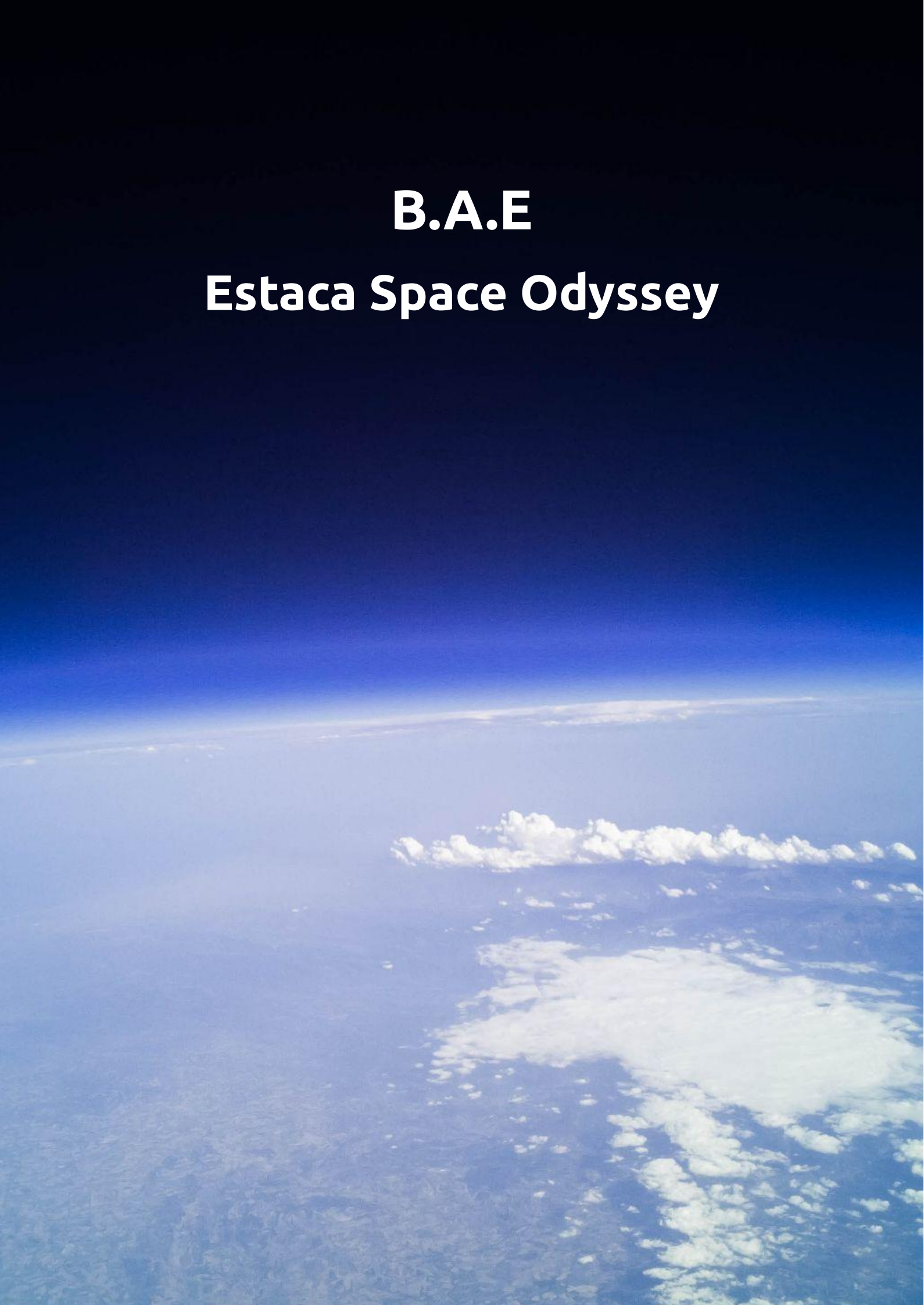
B.A.E

Estaca Space Odyssey



Sommaire

La nacelle juste avant le lâcher

1. Introduction

Les disciplines de la télécommunication et de la télémesure ont toujours été indispensables dans le domaine spatial. Elles permettent aux stations terrestres d'avoir un retour en temps réel de l'état d'un système dans l'atmosphère ou dans l'espace. L'importance de ces disciplines ne peut que croître dans les décennies à venir. En effet, les enjeux se portent de plus en plus vers de l'exploration lointaine.

Étant étudiants en ingénierie aérospatiale et curieux de tester de nouvelles technologies de télécommunication/télémesure, nous souhaitions réaliser un projet qui présente des défis dans cette discipline.

D'autre part, notre association prépare un projet ambitieux de fusée sonde dont l'apogée dépasserait les 30 km. La possibilité de tester des composants et circuits électroniques dans des conditions réelles à cette altitude nous intéressait.

Nous avons ainsi décidé d'entreprendre un projet de ballon sonde dans le cadre du C'SPACE et selon le cahier des charges de Planète Science et du CNES. Cette catégorie de projet est celle qui monte le plus en altitude (pouvant atteindre plusieurs dizaines de kilomètres et plusieurs centaines de kilomètres de portée). Nous avons jugé que ce projet répondait le plus à nos exigences. Ce projet est nommé B.A.E pour Ballon Avionique ESO.

Dans ce rapport, vous trouverez les explications détaillées des aspects mécanique, électronique et informatique de chaque expériences constituant notre projet. Vous trouverez également un descriptif du vol du ballon (qui a eu lieu durant le C'SPACE 2023) ainsi qu'une présentation des résultats de ce vol où l'on analysera les données acquises. Finalement, une partie sera consacrée aux voies d'améliorations possibles pour de futurs projets similaires.

2. Architecture mécanique

a. Première version de la nacelle

Après avoir commencé par une modélisation 3D de la nacelle pour estimer la place nécessaire afin de pouvoir insérer tous les éléments, nous avons réalisé une première version. La nacelle était composée de 7 morceaux : 4 murs, un sol, un plafond et une séparation interne. En effet, l'intérieur était séparé en deux : la partie supérieure (complètement isolée de l'extérieur) où se trouvait les composants électroniques du ballon, et la partie inférieure (exposée à la température extérieur par des trous dans les murs) où se trouvait les composants électroniques de l'expérience. La partie avionique était accessible en retirant le sol de la nacelle et la partie support était accessible en retirant le plafond.

Pour réaliser les différents morceaux de la nacelle, nous avons utilisé du polystyrène expansé que nous avons découpé grâce à un fil chaud. Puis nous avons voulu les coller ensemble mais nous avons rencontré quelques problèmes. Après avoir commencé avec de la colle néoprène, qui venait dissoudre le polystyrène, nous avons fini avec de la colle "sans clou ni vis" qui est une colle polymère hybride. Puis, nous avons voulu utiliser un pistolet à colle chaude pour terminer l'isolation de la partie supérieure, cependant la chaleur faisait fondre le polystyrène. Nous avons donc finalement opté pour un mastic de fixation.

Néanmoins cette version posait plusieurs problèmes : on ne pouvait pas intégrer l'avionique et le circuit de support en même temps, et il était difficile de fermer le sol et le plancher (qui aurait dû être collé au moment du lâcher).

b. Deuxième version de la nacelle

Pour cette deuxième version, nous avons décidé d'essayer un système d'emboîtement où chaque morceau vient s'insérer dans les autres. Cependant, il n'est pas possible de le réaliser en coupant le polystyrène avec un fil chaud. Nous avons donc utilisé une machine CNC (Computer Numerical Control) qui permet d'enlever de la matière. Cette technique nous a aussi permis de créer une porte qui pouvait coulisser, ce qui réglait un autre problème de la 1ère version.

Cependant, lors de la conception 3D de la nacelle, nous avions peur que, même si elle était isolée, la partie supérieure soit trop basse en température notamment pour la batterie. Nous l'avions donc dimensionné avec le volume le plus petit possible, d'après les certifications du CNES, afin que l'air à l'intérieur reste le plus chaud possible. Cependant nous n'avions pas pris en compte que, plus tard lors de l'intégration, de nouveaux composants électroniques se rajouteraient ce qui laisse moins d'espace disponible pour l'air et plus de sources de chaleur. Ainsi, alors que nous avions peur d'une température trop basse, nous nous sommes retrouvés avec une température trop haute. Pour contrer cela, nous avons percé trois petits trous à l'arrière qui donnaient à l'intérieur sur un ventilateur qui envoyait cet air frais de l'extérieur sur le micro-ordinateur.

c. Utilisation d’une couverture de survie

Comme expliqué dans le paragraphe précédent, notre première peur a été de se retrouver avec une température trop basse dans la partie supérieure. Nous avons donc ajouté une couverture de survie en mettant le côté doré vers l'extérieur, afin de conserver la chaleur.

3. Architecture électronique

a. Circuit de support

Pour vérifier que l’avionique fonctionne correctement dans les différentes phases du vol, il nous fallait un circuit de support connecté à l’avionique. Ce circuit est composé d’un micro-contrôleur RP2040 qui agrège les données de la majorité des capteurs ainsi que les données de l’avionique, les enregistre dans une carte sd et le retransmet par UART sur une Raspberry Pi 4. Le rôle du RP2040 est d’exécuter un programme simple et d’enregistrer toutes les données brutes qu’il reçoit. La Raspberry Pi 4 reçoit toutes les données reçues par le RP2040 et les enregistrent de son côté. Mais elle s’occupe aussi de tâches plus complexes.

Elle est équipée d’un module 4G LTE Cat-4 SIM7600G-H qui lui permet d’accéder à internet via des antennes relais au sol. Le choix de cette technologie est détaillé plus tard. Il est donc possible de s’y connecter à distance et de transférer un volume important de données lorsque le ballon est relativement proche du sol.

Elle enregistre également une vidéo prise par une caméra CSI-2 dans deux résolutions et qualités différentes: 1640x1232 10 Mbit/s et 640x480 700 kbit/s. Le flux haute résolution étant uniquement enregistré sur carte SD tandis que le flux basse résolution est envoyé vers un serveur par 4G pour un visionnage en direct. Toutes les minutes la caméra prend une photo en résolution maximale (8Mp) ce qui engendre un arrêt des flux vidéos pendant 0.8s. La compression des photos en jpeg et DNG se fait en arrière plan après que les flux vidéos sont rétablis. On gagne 1.2s de vidéo par prise de vue grâce à ce traitement en arrière plan.

Après avoir récupéré les données de tous les capteurs sous forme de fichier texte, elle compresse toutes les données grâce à l’algorithme zstandard qui permet de diviser jusqu’à 10 fois la taille de ces données.
Par exemple, l'entièreté des trames NMEA du GPS enregistrées pendant le vol pèsent 11Mo et elles pèsent, une fois compressées, 1.4Mo. Cette compression alliée à l’abondante bande passante de la 4G, rend l’impact d’un filtrage des données et d’une conversion de texte à binaire négligeable. Nous pouvons donc transmettre les données brutes et faire le traitement sur le serveur sans perdre de données qui pourraient s’avérer utiles plus tard.

Voici la liste des capteurs et à quel système ils sont connectés:

- **DS18B20** n°1: Mesure la température situé au dessus du CPU de la Raspberry Pi 4 et connecté au **RP2040 par 1-Wire**
- **DS18B20** n°2: Situé près de la batterie, connecté au **RP2040 par 1-Wire**

- **DS18B20** n°3: Situé en dessous de la nacelle, à l'extérieur, connecté à la **Raspberry Pi 4 par 1-Wire**
- **DS18B20** n°4: Situé dans le compartiment avionique, connecté à la **Raspberry Pi 4 par 1-Wire**
- **BMP280**: Capteur de pression et de température, connecté au **RP2040 par I²C**
- **Avionique**: Connecté au **RP2040 par UART**
- **INA219** Ballon: Mesure la puissance électrique en sortie de la batterie, en amont de l'abaisseur de tension. Connecté au **RP2040 par I²C**
- **INA219** Avionique: Mesure la puissance électrique consommée par l'avionique, en aval de l'abaisseur de tension. Connecté au **RP2040 par I²C**
- **J305** Compteur geiger: connecté au **RP2040 par GPIO en mode IRQ**
- **MAX-M8Q**: Module GPS haute altitude, connecté à la **Raspberry Pi 4 par UART** à travers le chip USB vers UART CP2102 de la carte 4G
- **Raspberry Pi Camera V2:** Pointée vers l'extérieur sur le côté de la nacelle, connectée à la **Raspberry Pi 4**

b. Gestion de la puissance électrique

Les principaux consommateurs de courants sont la Raspberry Pi 4, la carte 4G, la Raspberry Pi pico et l'avionique. Elles sont toutes alimentées en 5V directement depuis un régulateur UBEC de 3A continue et 5A en pic. Lui même connecté à une batterie Li-Ion 3S2P de 58Wh composée de 6 cellules 18650 et avec un BMS intégré. Un capteur INA219 est connecté en série avec la batterie côté positif, pour mesurer la puissance consommée et la tension de la batterie. Le régulateur UBEC est basé sur un abaisseur de tension de type hacheur série mais intègre en plus des protections en court-circuit, en surintensité, un blindage métallique et une réduction du bruit dans la tension de sortie.

Le GPS et le compteur Geiger étaient également alimenté en 5V. Les autres capteurs étaient alimentés par le régulateur 3.3V du Raspberry Pi 4 ou du RP2040 car ils ne consommait que très peu de courant.

Notre consommation maximale avec l'avionique, en transmission à 8 Mbit/s et en enregistrant une vidéo est de 10W, ce qui nous laisse 5h30 d'autonomie. Sachant que pendant la majeure partie du vol nous n'émettons pas en 4G la puissance consommée est alors de 7W, ce qui nous laisse largement assez d'autonomie pour le vol et récupérer toutes les données.

4. Architecture informatique

a. Raspberry Pi 4

La Raspberry pi 4 était sous Raspbian bullseye 64bit. Pour accélérer le développement et sachant que nous n'avions pas besoin de précision temporelle ni d'une grande fréquence d'échantillonnage nous avons décidé d'utiliser du Python. Les scripts étaient divisés le plus possible pour isoler les potentielles erreurs. Chaque script était lancé dans un "screen" linux pour nous permettre d'interagir avec eux sans pour autant que la shell utilisateur soit attaché au script. Chaque screen était lancé et surveillé par systemd. Si un script plante, il est automatiquement relancé 5 secondes après. Chaque

script était tolérant à un reboot de la Raspberry Pi 4 : grâce à systemd les scripts redémarrent en cas de reboot.

b. RP2040

Pour les mêmes raisons que la Raspberry Pi 4, nous avons choisi de programmer le RP2040 en MicroPython.

Nous avons adopté une architecture orientée objet avec une classe parent appelée Sensor. Elle implémente une méthode “log_data” commune à tous les capteurs qui permet d’écrire une chaîne de caractères contenant les données mesurées et la date correspondant à ces mesures sur la carte sd et sur le port série relié à la Raspberry Pi 4.

Les classes filles (une pour chaque type de capteur : DS, BMP…) implémentent une méthode “measure” qui récupère les données des capteurs et appellent la méthode parent “log_data” avec comme argument la chaîne de caractère à enregistrer. La lecture des données est différente pour chaque type de capteurs (qui n’utilisent d’ailleurs pas tous le même protocole). Pour les capteurs de type DS (température) nous avons utilisé les bibliothèques “onewire” et “ds18x20” pour pouvoir créer une liste qui récupère les données mesurées par les capteurs. Pour les capteurs de type BMP (pression) nous avons utilisé les bibliothèques “i2c” et “bmp280”, en utilisant leurs classes et leurs méthodes, nous avons pu accéder aux données du capteur. De la même façon, dans la classe fille “INA(Sensor)” de notre programme (capteurs de courant et de tension), nous avons utilisé les bibliothèques “i2c” et “ina219”.

L’architecture informatique de la RP2040 a été décidée en raison des similitudes de fonctionnement des différents capteurs. Elle permet d'enregistrer et d'envoyer les données de tous les capteurs de la même manière (d’ailleurs, l’expérience principale, détaillées dans la prochaine partie, est même considérée aux yeux de la RP2040, comme un capteur qui envoie ses données par protocole UART). Cette architecture et ce découpage en classes mère et filles simplifie considérablement le programme.

5. Expérience principale

a. Cadre

L'expérience principale à bord du Ballon englobait de manière exhaustive l'avionique de l'ESTACA SPACE LAUNCHER -1, le fer de lance spatial élaboré par l'ESO dans l'optique de pulvériser le record d’altitude étudiant lors de l'EUROC. ESL-1, itération inaugurale du projet, vise à être lancé lors du prochain événement C’Space en 2024. Le dispositif avionique déployé visait l'acquisition et l'archivage massif de données émanant de la multitude de capteurs embarqués, une fraction de ces données étant destinée à une diffusion par le biais de télémesure. L'intégralité des composants présents à bord a été soumise aux contraintes environnementales par les ouvertures spécifiquement agencées dans la partie inférieure de la nacelle.

b. Composants

Les composants sélectionnés revêtent des rôles spécifiques, chacun contribuant de manière substantielle à l'architecture globale. Les directives du cahier des charges de l'EUROC demeurent explicites : le déploiement du parachute principal est impératif avant d'atteindre l'altitude de 420 mètres. Dans cette optique, l'évaluation précise de l'altitude et de l'apogée s'avère cruciale. Dans notre contexte, cette tâche est confiée non pas à une minuterie, mais plutôt à un traitement haute vitesse des données provenant des centrales inertielles, barométriques et GPS. Ces informations seront ensuite acheminées vers un filtre de Kalman, qui réduira significativement les potentielles erreurs liées à l'échantillonnage des capteurs. L'intégration des données inertielles en collaboration avec les données GPS permettra de déterminer avec précision la position spatiale.

Ici les composants utilisés :

- **Teensy 4.1**: Micro-contrôleur régissant la collecte, le traitement, la sauvegarde et le transfert des données
- **Arduino Uno:** Micro-contrôleur relié au module de réception télémétrique au sol
- **BMP280**: Capteur de pression et de température, connecté au **Teensy 4.1par I²C**
- **BNO055:** Centrale inertielle sur 9 degrés de liberté (gyroscope x,y,z;accéléromètre x,y,z; magnétomètre x,y,z), connecté au **Teensy 4.1par I²C sur le premier Bus**
- **BNO055:** Centrale inertielle sur 9 degrés de liberté (gyroscope x,y,z; accéléromètre x,y,z; magnétomètre x,y,z), connecté au **Teensy 4.1par I²C sur le second Bus**
- **NEO-M9N:** Gps , connecté au **Teensy 4.1 via une connection en UART**
- **LoRa SX1276MBMAS:** Module de transmission télémétrique connecté au **Teensy 4.1 via une connection en SPI**
- **LoRa SX1276MBMAS:** Module de réception télémétrique connecté au **ARDUINO UNO via une connection en SPI**
- **MICRO-SD:** connecté au **Teensy 4.1 via une connection en SPI**
- **Ballon:** connecté au **Teensy 4.1 via une connection en UART**
- **BUZZER:** connecté au **Teensy 4.1 via une connection en PWM**

Le choix de ces éléments découle de leur capacité à endurer les contraintes inhérentes au vol, conformément aux conditions familières de manipulation par les membres de l'équipe.

c. Architecture électrique:

En ce qui concerne la configuration architecturale du circuit imprimé, il s'agit d'un disque d'environ 18 cm de diamètre, possédant une géométrie spécifiquement conçue pour optimiser l'utilisation de l'espace au sein du rack d'ESL-1. L'agencement du circuit imprimé était justifié par les caractéristiques des composants qu'elle devait accueillir. En effet, jusqu'à la phase de contrôle du C'Space, la gestion de la télémétrie était orchestrée par un microcontrôleur distinct du principal Teensy 4.1. Il s'agissait précisément d'un STM32 Nucleo-64 L073RZ, qui nécessitait une place considérable. Ce choix découle d'une démarche cohérente, étant donné qu'un projet associatif il y a quelques années avait déjà été élaboré autour de cet équipement L073RZ, associé au module SX1276MBMAS.

Malheureusement, à cause de la complexité de la communication inter-cartes, nous avons finalement pris la décision de ne recourir qu'au Teensy 4.1 exclusivement.

Nous avons opté pour la conception d'un circuit imprimé à double couche afin de rationaliser les coûts de fabrication de notre PCB. Sur ces deux couches, la partie restante du cuivre, à l'exception des pistes tracées, était dédiée à la masse et connectée à l'aide de vias.

Les tracés présents sur la couche 1 étaient principalement disposés horizontalement, tandis que ceux sur la couche 2 adoptaient majoritairement une orientation verticale. En adoptant cette approche de routage, plusieurs avantages significatifs étaient réalisés, tels qu'une réduction des interférences entre les signaux et une diminution des risques de couplage indésirable. De plus, une telle disposition simplifie la tâche de débogage et facilite les modifications ultérieures du circuit, tout en contribuant à une meilleure efficacité du plan de masse et à une amélioration générale de la qualité du signal.

Le second capteur Bno055 était interconnecté avec le capteur BMP280 sur le même Bus I²C vers le teensy 4.1, leurs adresse étant différente cela n'a pas posé de problèmes, en revanche le second capteur IMU Bno055 manquait deux connections SCL et SDA vers le MCU (Micro-Controller Unit) et ont dû être reliés par des câbles soudées.

L'architecture actuelle du PCB est plus que suffisante en vue d'un vol à bord de l'ESL-1. Parallèlement, l'implémentation d'une minuterie dans le code du microcontrôleur est sérieusement envisagée. Cette approche offrirait une forme de redondance cruciale en prévision du vol lors du C'Space en 2024. Cependant, dans le contexte de la version 2 de l'unité avionique, une orientation vers un format plus compact sera favorisée. De plus, l'objectif sera de consolider le système en adoptant un unique microcontrôleur en visant une transition vers des composants montés en surface (CMS) pour les capteurs, ainsi que pour le microcontrôleur. Ces améliorations seront une cible atteignable pour le pôle avionique du projet ESL.

d. Télémétrie LoRA

En ce qui concerne la télémétrie, nous avons pris la décision de réutiliser le travail d'un projet antérieur mené en 2020/2021, qui consistait à élaborer la télémétrie pour le projet ESL. Dans cette optique, nous avons repris l'intégralité du projet, englobant à la fois le matériel et le logiciel. Le projet se composait d'un module émetteur/récepteur LoRa SX1276MBAS, pouvant être ajusté sur une plage de fréquences allant de 434 MHz à 868 MHz.

Les spécifications principales du transceiver SX1276 étaient les suivantes :

- Puissance de sortie : jusqu'à +20 dBm / 100 mW
- Sensibilité : jusqu'à -148 dBm
- Budget de liaison maximal : 168 dBm

Où dBm correspond au "décibel milliwatt", une unité de mesure de la puissance absolue en décibels par rapport à 1 mW : dBm = 10log(Pout/1mW) avec Pout mesuré en mW.

Ce module ne disposait pas de microcontrôleur intégré et avait été apparié avec un Nucleo-L073RZ, la carte préconisée par Semtech, afin de nous permettre de programmer la télémesure en C++.

La modulation LoRa offre plusieurs modes de transmission, sélectionnables en ajustant deux paramètres : le facteur d'étalement et la largeur de bande. Un facteur d'étalement plus élevé favorise une portée accrue mais diminue le débit, tandis qu'une largeur de bande plus importante engendre un débit supérieur mais une portée moindre. Ces paramètres peuvent donc être adaptés facilement en fonction des besoins spécifiques.
Il est important de noter que le débit demeure indépendant de la fréquence utilisée, du fait de la manière dont la modulation LoRa transmet les données.
Nos paramètres étaient les suivants :

```
// LoRa settings
#define BAND 433.5E6    // Frequency band
#define PW 11           // Power (0-20)
#define SF 9           // Spreading factor (6-12)
#define BW 250E3       // Bandwidth (10.4, 15.6, 20.8, 31.25, 41.7, 62.5, 125, 250, 500)
#define CR 8           // Coding rate (5 -> 4/5 (1.25 overhead), 6 -> 4/6 (1.5), 7 -> 4/7 (1.75), 9 -> 4/8 (2))

bool printserial=false;
```

L'antenne Yagi en notre possession pour la réception était calibrée pour 434 MHz, nous obligeant à opérer ces modules sur cette fréquence en utilisant une antenne émettrice avec un gain de 2 dBi. Le gain modeste de l'antenne émettrice pouvait être aisément compensé grâce à l'antenne réceptrice.

Malheureusement, nous avons rencontré des difficultés à retrouver la portée atteinte lors du vol en 2020/2021. Après investigation, il s'est avéré que la broche RxTx du module devait être alimentée à 3,3 V, car les modules étaient initialement en mode réception. Cette correction nous a permis d'étendre la portée, passant de 11 cm à 800 mètres. Néanmoins, cette portée demeurait encore bien en dessous de nos attentes, ce qui nous a incités à explorer d'autres pistes, en vain.
Il convient de souligner que la réglementation nous restreint également à une puissance d'émission de 10 mW à 434,5 MHz.

Frequency Bands	Maximum radiated power, e.r.p.	Channel spacing	Duty cycle
433,050 MHz to 434,790 MHz	10 mW	≤25 kHz	10% @ <10 mW
433,050 MHz to 434,790 MHz	1 mW	No restriction	100% @ ≤1 mW
434,040 MHz to 434,790 MHz	10 mW	≤25 kHz	100%

En vue des développements futurs, nous envisageons d'adopter une cadence de transmission réduite, même si nous restions en dessous débit théorique de 50 kbits/s.

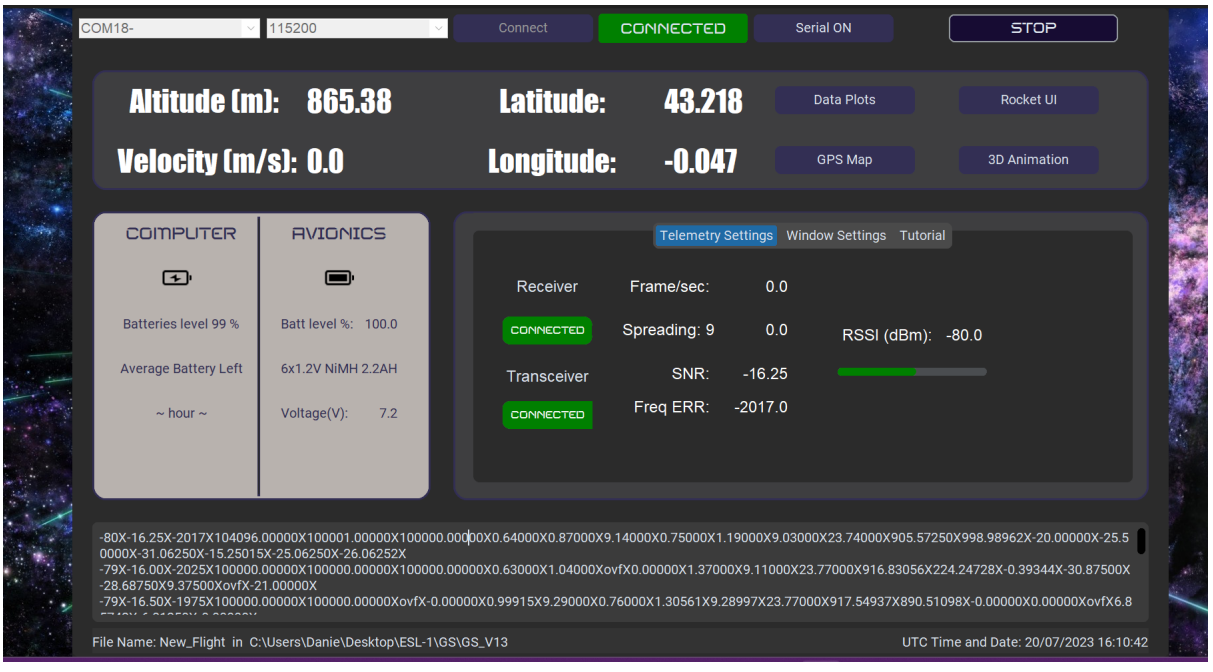
e. Station Sol

La station sol recevait les données par le biais d'un câble USB relié à un Arduino Uno, qui était en liaison avec le module LoRa de réception, connecté à l'antenne Yagi.

L'interface, développée en Python, s'appuyait principalement sur la bibliothèque personnalisée "custom tkinter" pour l'affichage de l'interface graphique, ainsi que sur la bibliothèque "serial" pour l'ouverture et la lecture du port série. Les données étaient transmises sous forme de chaînes de caractères, chaque valeur étant séparée par un "x", par exemple : "12345X33.6X222.7". Après réception, les valeurs étaient ensuite parsées. Via la bibliothèque "maptkinter", nous pouvions afficher jusqu'à trois cartes Google Maps dans lesquelles nous pouvions positionner des marqueurs, tracer les parcours et afficher les coordonnées GPS en format LLH (latitude, longitude, altitude). L'interface avait la capacité d'enregistrer toutes les données reçues dans un fichier .csv.

Par ailleurs, elle pouvait afficher une représentation en 3D du véhicule, facilitant ainsi la vérification rapide de l'exactitude des données issues des IMU. Un module en cours de développement permettrait d'afficher la trajectoire spatiale de la fusée pour visualiser les données générées par le filtre de Kalman. La fonctionnalité de bascule entre le mode jour et le mode nuit s'est avérée utile, car la visibilité était différente sur le pas de tir selon le moment de la journée. La station au sol mettait en avant les données GPS, l'altitude et la vitesse du véhicule en priorité. En arrière-plan, elle fournissait des informations sur les batteries, notamment la tension et la télémétrie. Enfin, elle permettait de choisir d'afficher ou non les données brutes, en fonction des besoins.

Dernière Trame reçue:



f. Conclusion

En résumé, malgré des défis tels que la perte de membres en cours de route et la nouveauté du domaine pour chaque membre de l'équipe, notre expérience a été un succès global. Nous avons conçu un circuit imprimé sophistiqué en utilisant habilement les microcontrôleurs et les modules LoRa pour la télémétrie, bien que la connexion au-delà de 600 mètres ait été rompu . Nous avons montré notre capacité à innover et à atteindre nos

objectifs. Malgré quelques redémarrages, l'avionique est restée fonctionnelle pendant toute la durée du vol.
Cependant, malgré nos réalisations, nous reconnaissons qu'il existe plusieurs axes de progression et avons hâte de présenter un encore meilleur travail au prochain C'Space.

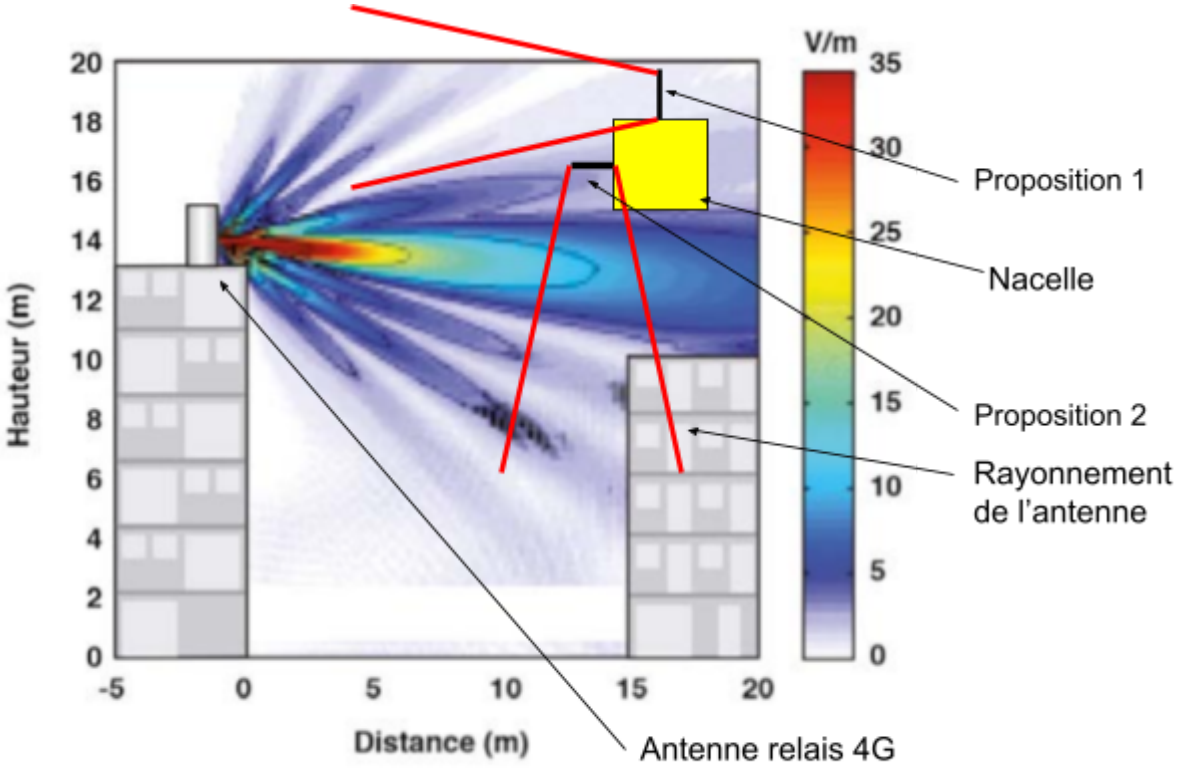
6. Expérience secondaire

a. Télémesure 4G

Nous devions choisir une télémesure propre au circuit de support qui ne causerait pas d'interférences avec la télémesure LoRa que l'avionique devait utiliser. Sachant que l'avionique n'étaient pas encore certains de quelle fréquence LoRa (433 ou 868MHz) ils allaient utiliser, nous ne pouvions pas simplement décider d'utiliser une télémesure LoRa utilisant l'autre fréquence. L'idée nous est alors venue d'utiliser la 4G LTE Cat-4. Cette technologie nous permet de nous affranchir d'une station sol et donc de ne pas avoir à pointer le ballon sonde avec une antenne directionnelle. La portée est également théoriquement infinie. Même si le ballon atterrit à des centaines de kilomètres de nous, si la zone est couverte par une antenne relais, nous aurons un signal. Un autre avantage est l'importante bande passante qu'elle nous apporte : Une télémesure LoRa est capable d'envoyer des données avec une bande passante entre une cinquantaine de bit/s et 62 kbit/s en fonction des réglages. Tandis que la 4G LTE Cat-4 peut fournir entre 1 et 10 Mbit/s.
La télémesure 4G a quand même des désavantage, le premier étant la consommation qui peut aller jusqu'à 4W quand on transfert à 10Mbit/s mais le principal est l'absence de signal en haute altitude.

b. 4G LTE en altitude

En France, les antennes relais 4G sont dirigées vers le sol car toute la puissance est émise dans le ciel à cause du manque de clients en haute altitude. Même si les antennes relais étaient omni-directionnelles, la portée des antennes 4G est d'environ 10 km ce qui n'est qu'un tiers de l'apogée de notre ballon-sonde. Nous avons donc décidé d'abandonner l'idée d'avoir de la 4G en haute altitude et de nous concentrer sur la qualité du signal proche du sol. Cela nous permettra de transférer l'entièreté des données de vol peu avant l'atterrissage grâce à l'importante bande passante. Nous avons donc orienté l'antenne verticalement de sorte à capter les antennes relais horizontalement plutôt que par le dessus. Sur le graphique ci-dessous, on peut voir que la proposition 1 nous offre un meilleur signal proche du sol. Tandis que la proposition 2 ne nous donne quasiment aucun signal au sol et très peu en haute altitude à cause de l'orientation de l'antenne relais 4G.



Source du diagramme: INRS

Nous avons trouvé très peu d'informations sur la qualité du signal 4G en altitude, seulement certains forum d'aviateurs particuliers qui disent avoir du signal GSM à environ 2 km d'altitude mais avec un smartphone et à l'intérieur d'une carlingue. Notre antenne a un meilleur gain que les smartphones et est à l'air libre.

c. Protocoles réseau

Les trois flux de données sortant de la Raspberry Pi sont: la vidéo en direct, une trame de "survie" et les fichiers de vol. Le signal 4G pouvant être instable à cause de la distance, l'altitude et la vitesse, nous devions donc utiliser un protocole réseau résilient.

Pour la vidéo nous avons opté pour le protocole SRT (Secure Reliable Transport) qui se base sur un protocole UDP. Il a l'avantage par rapport aux autres de laisser l'émetteur envoyer tous les paquets qu'il veut à la vitesse où il le peut et le récepteur demande seulement à intervalle régulier de retransmettre certains paquets qui se seraient perdus afin de les rassembler dans la pile. Comme ça chaque paquet émis n'a pas à attendre de savoir s'il a bien été reçu pour envoyer le suivant. Et si la retransmission du paquet n'arrive pas dans un temps imparti, le récepteur peut passer une image sans bloquer.
L'émetteur n'a également pas besoin de vérifier si le récepteur est présent pour envoyer les données, il n'y a donc pas de boucle de re-connexion à effectuer.

Pour les fichiers nous voulions utiliser le même protocole mais il fallait relancer le serveur à chaque fois que nous voulions envoyer un autre fichier. De plus, les contraintes sur l'envoi d'un fichier sont bien différentes de celles pour l'envoi d'un flux vidéo en direct. Il est important lorsque nous transmettons un fichier de savoir s'il s'est envoyé complètement et sans corruption. Alors qu'avec un flux de vidéo en direct, perdre des images n'est pas un problème.
Nous avons donc opté pour la solution la plus simple et la plus fiable: scp (secure copy).
Ce programme utilise le protocole ssh lui-même basé sur le protocole TCP.
Le premier avantage est que scp vérifie qu'un fichier s'est transféré correctement avant de passer au suivant. Un autre avantage est que l'émetteur peut indiquer le chemin de destination au lieu de laisser le récepteur choisir. Le côté réception est assuré par un serveur ssh qui est toujours allumé. Donc aucun risque que l'émetteur se retrouve devant un port fermé, il faut seulement essayer continuellement d'envoyer les fichiers jusqu'au succès.

Pour la "trame de survie" nous avons choisi d'utiliser le protocole UDP car, contrairement au TCP il n'y a pas de vérification de la bonne transmission des paquets, il n'y a donc pas de blocage de paquets en attente de la retransmission d'un paquet échoué. Le désavantage est que nous risquons de perdre des paquets ou d'avoir des corruptions mais dans notre cas ce n'est pas un problème, si une trame n'est pas bien transmise, on peut attendre la prochaine. Il faut néanmoins savoir si une trame est corrompue, le premier signe est la longueur non-conforme de celle-ci, ce qui est facile à vérifier. Pour détecter un "bit-flip" (inversion d'un bit en chemin) nous avons ajouté une checksum CRC32 à la fin de chaque trame. En comparant la checksum calculée à la réception avec celle transmise on peut savoir si il y a eu un bit-flip. Comme son nom l'indique, l'overhead du CRC32 est de seulement 4 octets ce qui est négligeable par rapport à la bande passante que nous disposons et la taille de notre trame. L'autre avantage de l'UDP est que nous n'avons pas besoin d'établir de connexion avec le récepteur, donc l'émetteur peut envoyer des trames dans le vide jusqu'au moment où la carte 4G peut transmettre ces 54 octets. C'est vital car cette trame contient les coordonnées GPS et l'état général du ballon, c'est donc le dernier flux de données qui doit s'interrompre et il doit pouvoir se "faufiler" parmi les autres flux.

d. Station de réception

Contrairement à une télémesure radio classique, notre station sol n'est pas un ordinateur portable relié à une antenne directionnelle qu'il faut pointer en direction du ballon sonde. Mais bien un serveur au chaud dans un datacenter à 600 km du site de lâcher. C'est un des avantages de cette technologie. Sachant que nous utilisons principalement le protocole UDP qui est unidirectionnel, la latence n'est pas limitante. Le serveur pourrait être à l'autre bout du monde sans pour autant impacter les performances. En plus des serveurs SRT, UDP et SSH, il était équipé d'un programme qui envoie un message discord à toute l'équipe lorsque la trame de survie est perdue et quand elle est retrouvée. Ce programme nous envoie également toutes les 2 minutes un récapitulatif des principales mesures accompagné d'un lien google maps. Il s'occupe également de détecter quand il reçoit des fichiers par scp. Dès qu'un nouveau fichier lui est transmis, il génère plusieurs courbes et une carte de la trajectoire complète. Il nous notifie également des nouveaux fichiers et nouvelles photos reçues en nous envoyant des messages discord.

Message reçu à la récupération du signal 4G

7. Gestion thermique

Étant inexpérimenté dans le domaine des ballons sonde nous avions très peur des basses températures qui pourraient endommager nos composants. Nous avons donc utilisé 40mm d'épaisseur de polystyrène extrudé comme isolation. Après avoir fabriqué la nacelle et avoir fait un test dans un congélateur à -18°C, nous nous sommes rendu compte que simplement avec la Raspberry Pi 4 allumée, nous perdions seulement 2°C par heure à l'intérieur du compartiment ballon sonde. Sachant que le congélateur était à la pression atmosphérique standard de 10^5 Pa, et donc que le coefficient de convection thermique était au maximum. Nous avons également remarqué que le compartiment avionique, qui est aéré par deux trous de 4 cm de diamètre, a quand même un delta de température stable de 10°C dans un frigo à 0°C avec en plus une certaine inertie thermique.

Nous avions conçu la nacelle de sorte qu'elle soit la plus compacte possible pour diminuer la surface de convection. Le volume interne était donc le plus petit possible alors que nous avons dû ajouter des composants qui n'étaient pas prévus. Tout ceci lié à la consommation électrique assez importante (7W en émission 4G et avec enregistrement de vidéo), nous nous sommes heurtés à un problème inattendu: au bout de 1h30 les composants du ballon atteignent une température critique de 80°C et ce même dans un congélateur.

Nous avons divisé par 2 l'épaisseur de la plaque inter-étage, ce qui a eu pour effet d'ajouter un 2 cm d'air en plus et de multiplier par 2 le flux thermique entre les 2 étages. Nous avons également ajouté un ventilateur et 2 trous de 2 mm de diamètre à travers la

parois. Cela a permis de diminuer la vitesse à laquelle la température du compartiment support augmente et d'atteindre la température critique au bout de 2h30 dans un frigo à 0°C. Nous avons également intégré au programme d'enregistrement de la vidéo une protection qui désactive l'enregistrement si la température CPU monte au-dessus de 73°C. Tous les autres programmes y compris la prise de photos sont maintenus car ils ne demandent pas autant de puissance de calcul.

8. Déroulé du vol

a. Chaîne de vol

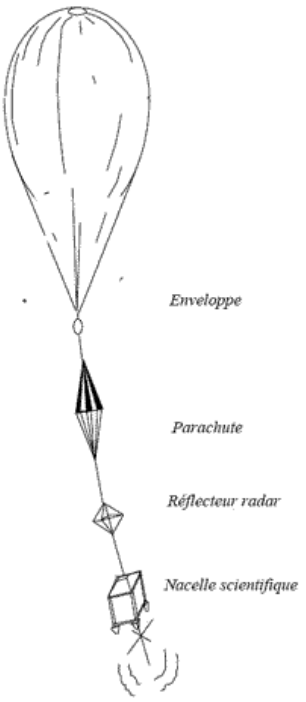
Le ballon-sonde se compose de quatre composantes essentielles qui, ensemble, forment la chaîne de vol intégrée :

> L'enveloppe : fabriquée à partir de caoutchouc (ou de latex), elle est gonflée avec de l'hélium.

> Le parachute : qui ralentit la descente de la nacelle une fois que l'enveloppe a éclaté.

> Le réflecteur-radar : conçu pour permettre aux avions de repérer le ballon et d'éviter toute collision.

> La nacelle : qui abrite les divers systèmes embarqués du ballon, tels que les équipements expérimentaux, les instruments de télémétrie, la caméra, …



Source Planète Sciences

b. Prévision de la trajectoire de vol

En accord avec les prévisions météorologiques et les données sur la direction des vents, le ballon a été gonflé afin d'atteindre une altitude de 29 km avant d'éclater. Sa trajectoire devait le porter environ 115 km vers le nord-est à partir de son point de départ. Après une période totale de vol de 2 heures et 20 minutes, le ballon devait toucher le sol juste à l'est de Toulouse.

c. Étapes du vol

> T-00:39 (15h00 / 0.43 km)

Installation du matériel dans la zone de lancement, montage de la chaîne de vol, préparation des bonbonnes d'hélium et de l'enveloppe. L'altitude du point de départ est de 430 m.

> T-00:14 (15h25 / 0.43 km)

Début du gonflage de l'enveloppe qui est maintenue sous une bâche pour éviter qu'elle ne s'envole trop tôt, démarrage de la nacelle.

> T-00:00 (15h39 / 0.43 km)

Lâcher du ballon

> T+00:07 (15h46 / 1.9 km)

Passage au-dessus de la couche nuageuse

> T+00:20 (15h59 / 4.97 km)

Perte du signal 4G et donc de la télémesure 4G et du direct de la caméra.

> T+01:37 (17h16 / 20.51 km)

Le ballon, en dépassant les 20 km d'altitude, se retrouve porté par les vents stratosphériques qui poussent vers l'ouest.

> T+02:25 (18h04 / 33.21 km)

Éclatement de l'enveloppe après avoir atteint 33,21 km d'altitude en 2h25 de vol, soit une vitesse de montée moyenne de 3,77 m/s. Le ballon, sans l'enveloppe, commence donc sa descente.

> T+02:31 (18h10 / 20.41 km)

Le ballon repasse sous les 20 km d'altitude et est de nouveau poussé par les vents de basse altitude vers le nord-est. Il est tombé de 12,8 km en 6 min soit une vitesse moyenne de 35,55 m/s.

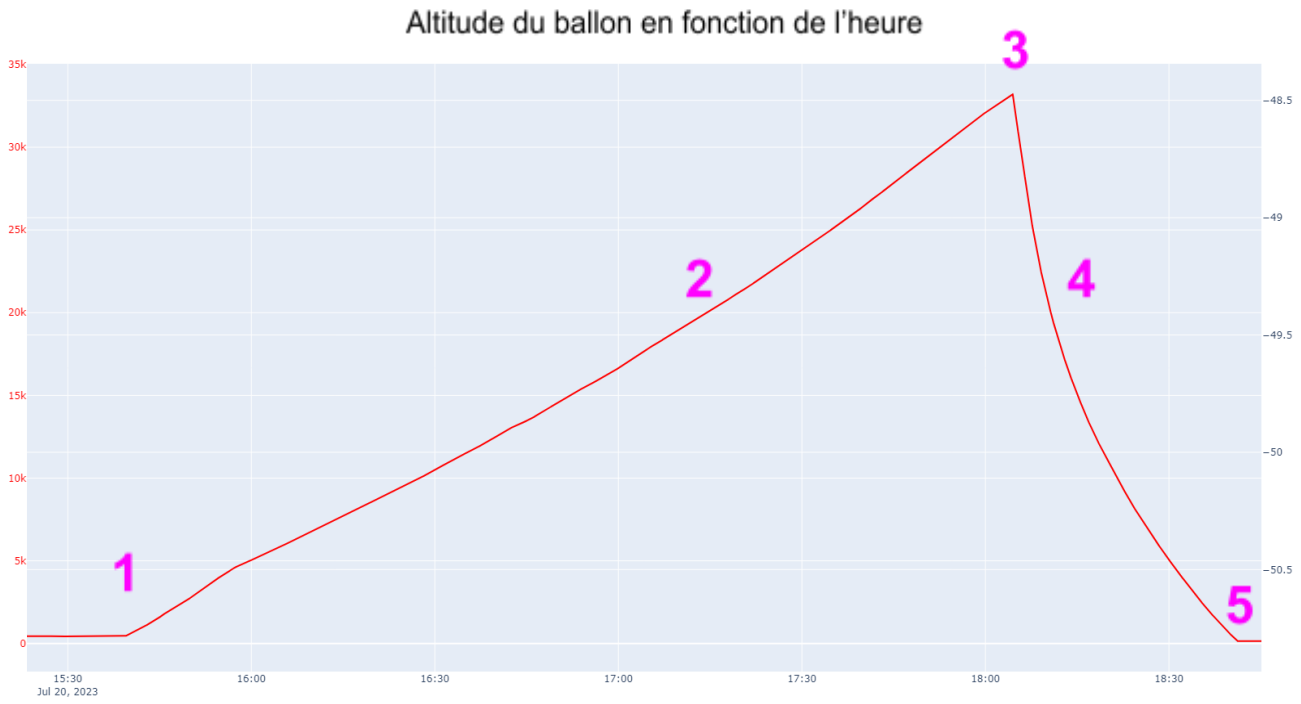
> T+03:01 (18h40 / 0.3 km)

Récupération du signal GSM 4G et de la position GPS du ballon.

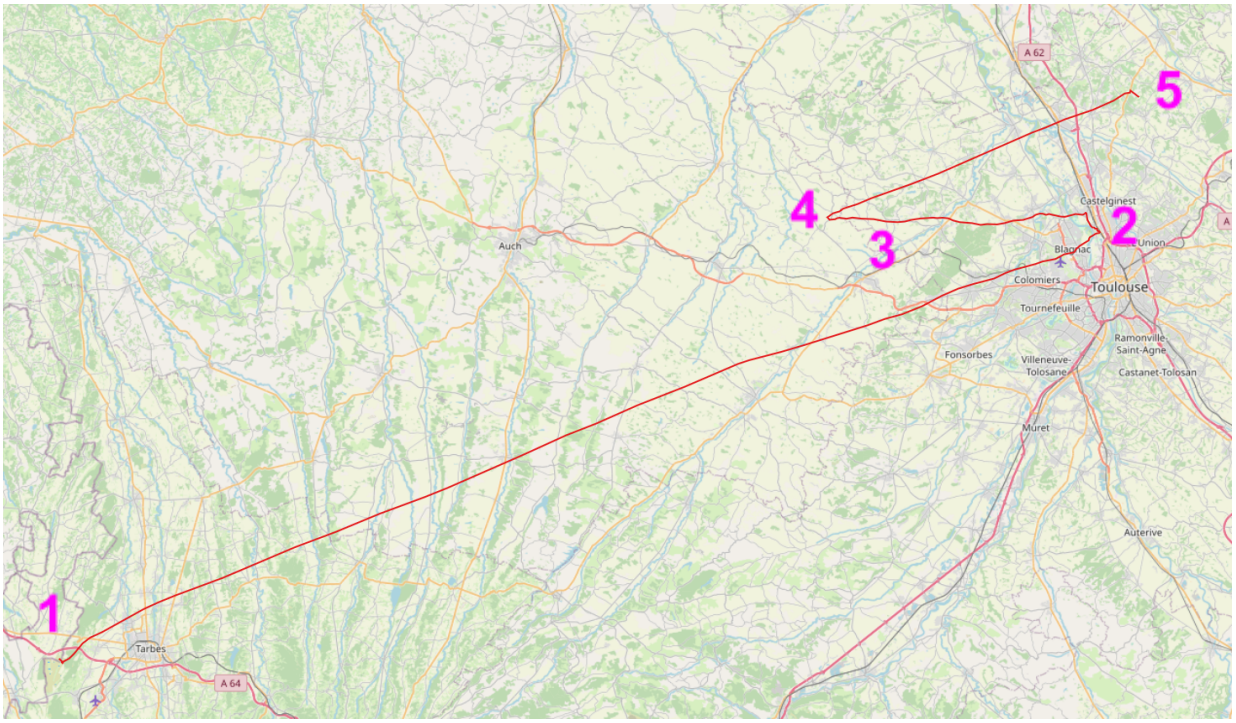
> T+03:02 (18h41 / 0.15 km)

Le ballon touche le sol à une altitude de 150m après une descente de 37 min. Dans la seconde partie de la descente (après être passé sous les 20 km d'altitude), la vitesse moyenne était de 10,89 m/s.

Le ballon a atterri près de la commune Vacquiers, au nord de Toulouse, à environ 140 km du point de départ.



Position GPS du ballon au cours du vol



> 1 : Lâcher du ballon
> 2 : Passage au dessus des 20 km d'altitude
> 3 : Eclatement de l'enveloppe
> 4 : Passage en dessous des 20 km d'altitude
> 5 : Atterrissage du ballon

d. Différence avec les prévisions

On peut noter une différence significative entre les prévisions et le réel déroulement du vol. Déjà par l'altitude d'éclatement (4 km de différence) et la distance parcourue (26 km de différence), mais surtout par la durée du vol. En effet, le vol a duré au total 3h02, soit 42 minutes de plus que prévu surtout à cause de la montée qui devait durer environ 1h35 et qui en réalité a duré 2h25 (50 minutes de plus).
La descente, quant à elle, a une différence négligeable avec une durée de 37 min au lieu des 45 min prévues. Toutes les autres différences peuvent s'expliquer par le fait que l'enveloppe a sûrement été légèrement sous-gonflée, le ballon est donc monté moins vite, a éclaté plus haut et a touché le sol plus loin que prévu.

e. Problème de parachute et de balise GSM

Pendant la montée, nous avons perdu le signal 4G à partir de 5 km d'altitude, ce qui était deux fois plus que nos prévisions. Cependant, alors que nous nous attendions à récupérer le signal à peu près à même altitude à la descente, nous avons reçu la notification seulement quand le ballon était à 150m du sol quelques dizaines de secondes avant de le toucher. Cela a pu être provoqué par plusieurs raisons, notamment une difficulté de l'antenne à se connecter à cause de la vitesse.

L’autre problème a été le parachute, en effet en arrivant sur le lieu d'arrivée du ballon, nous n’avons pas retrouvé tous les composants de la chaîne de vol. La nacelle et le réflecteur étaient intactes, cependant il manquait tout ce qui se trouvait au-dessus (dont le parachute). Nous avons aussi retrouvé des morceaux de l’enveloppe déchirée sur le haut du réflecteur. Au vu de la vitesse de descente très rapide au début et des dégâts observés, nous pensons que la rotation rapide de la nacelle après l’éclatement de l’enveloppe a provoqué le détachement du parachute. Notre nacelle étant assez légère (1,2 kg), nous supposons que le réflecteur a agit comme un petit parachute et a ralenti suffisamment la nacelle pour la faire revenir intacte.

9. Résultats

L'avionique a fonctionné pendant tout le vol malgré 2 redémarrages du MCU. Le signal de la télémesure LoRa a été perdu au bout de 500m de distance mais toutes les données ont été à la fois enregistrées sur carte sd mais aussi communiquées par UART au circuit de support qui a pu les re-transmettre par 4G.

La totalité du ballon sonde a consommé 24 Wh pendant les 3h de vol soit la moitié de sa capacité. Mais le temps que l'équipe arrive sur place et récupère le ballon, il a consommé 8 Wh de plus pour envoyer par 4G les photos et vidéos qui n'ont pas pu l'être pendant le vol.

La télémesure 4G du circuit de support a pu envoyer le live vidéo, la trame de survie et les fichiers de vol (photos et mesures compressées) jusqu'à 5km de haut. Bien que nous n'ayons pas utilisé la totalité de la bande passante disponible, nous avons pu émettre en moyenne 2 Mbit/s jusqu'à 5 km d'altitude où la 4G s'est brusquement coupée par manque de signal. À la redescente, le signal s'est rétabli assez bas, à 150m au-dessus du sol mais nous ne savons pas si c'est par manque de signal ou s'il fallait du temps pour accrocher les antennes relais après avoir parcouru 140 km pendant 2h sans signal. Néanmoins pendant les 10 secondes de descente restantes on a pu transmettre l'ensemble des données de l'avionique et la totalité des mesures du circuit de support. De plus, nous avions un très bon signal une fois au sol ce qui nous a permis de récupérer 300 Mo de photos et de vidéos à 8 Mbit/s avant que nous arrivions sur place.

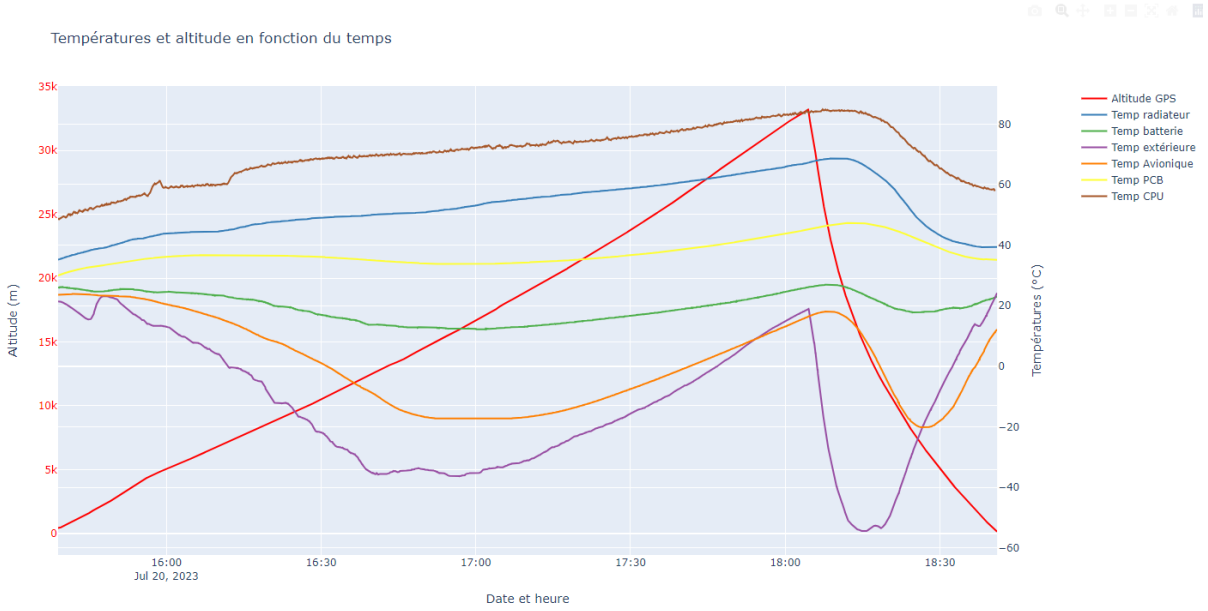
a. Photos

Photo de la courbure de la terre et de la couche d'ozone prise à 32 km d'altitude

Dépassement des nuages à 2,2 km d'altitude

[Lien vers un diaporama de toutes les photos]

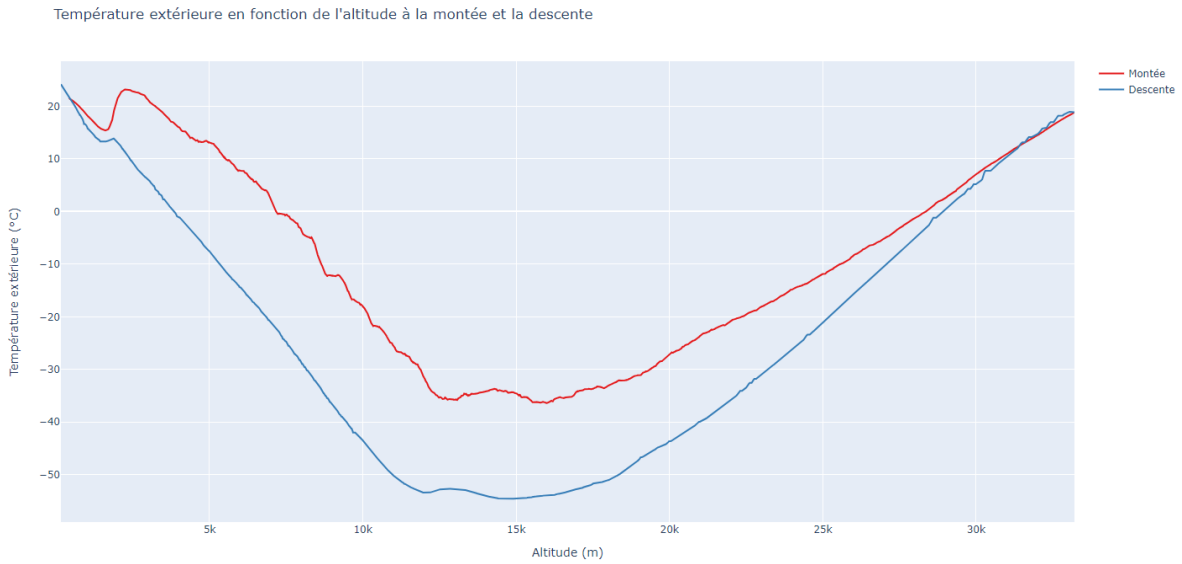
b. Températures et altitude



[Lien du graphique en haute résolution]

Ce graphique permet de voir l'évolution des 5 capteurs de température installés à différents endroits de la nacelle. Nous pouvons remarquer que même le compartiment avionique (courbe orange) possède une grande inertie thermique. Alors que la température extérieure était à son minimum de -54°C, le compartiment avionique, aéré par 2 trous de 4 cm était 64°C plus chaud à 10°C et a continué à perdre de la chaleur 10 minutes après que la température extérieure ait commencé à augmenter.

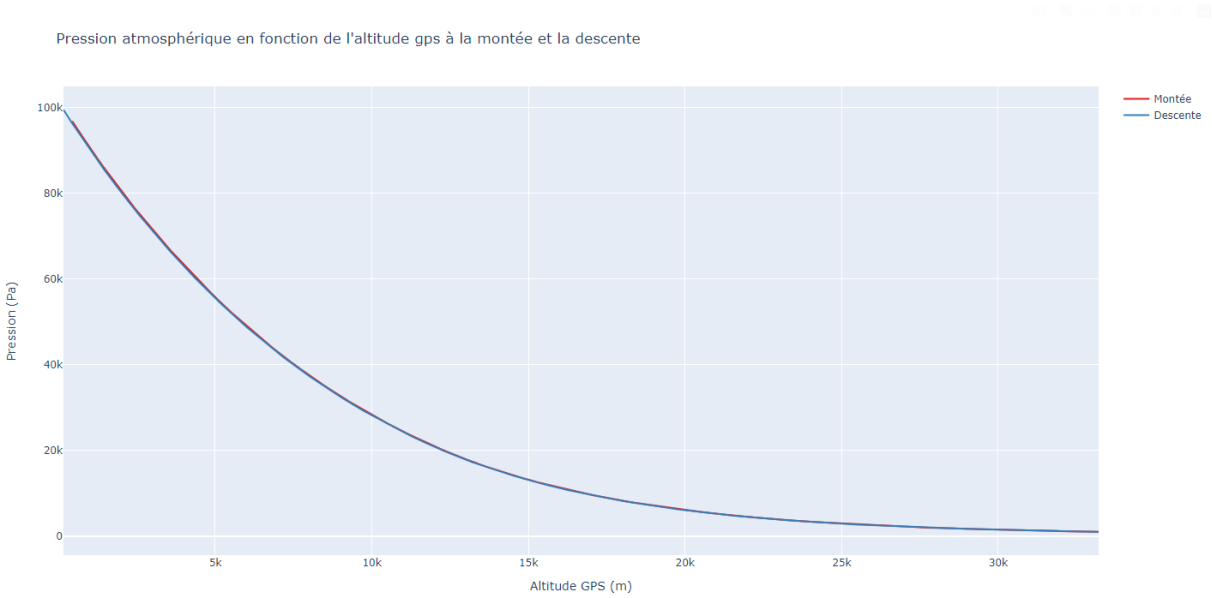
On peut voir que la température du compartiment de support est restée plutôt chaude avec un minimum en vol de 12°C sur la batterie, 34°C sur le pcb et 62°C à l'intérieur du cpu de la Raspberry Pi 4. Tandis que la température extérieure est descendue brusquement quand l'enveloppe a éclaté jusqu'à atteindre la température minimale de -54°C.



[Lien du graphique en haute résolution]

On peut également constater que la température descend jusqu'à 15 km d'altitude avant de remonter jusqu'à l'éclatement. On peut voir qu'à la descente la température est plus faible de 13°C en moyenne, sans doute à cause de la vitesse de descente plus élevée et du vent relatif associé.

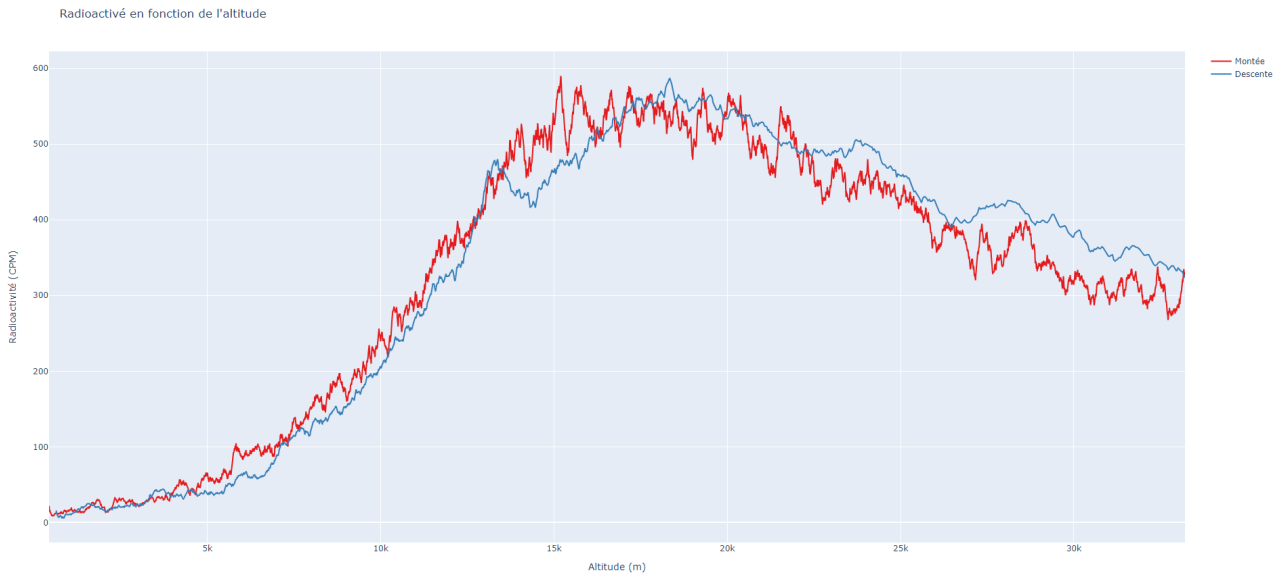
c. Pression



[Lien du graphique en haute résolution]

Les courbes de pression en fonction de l'altitude GPS en montée et en descente sont quasiment confondues ce qui est intéressant sachant que la descente a eu lieu 140 km plus loin que la montée.

d. Radioactivité

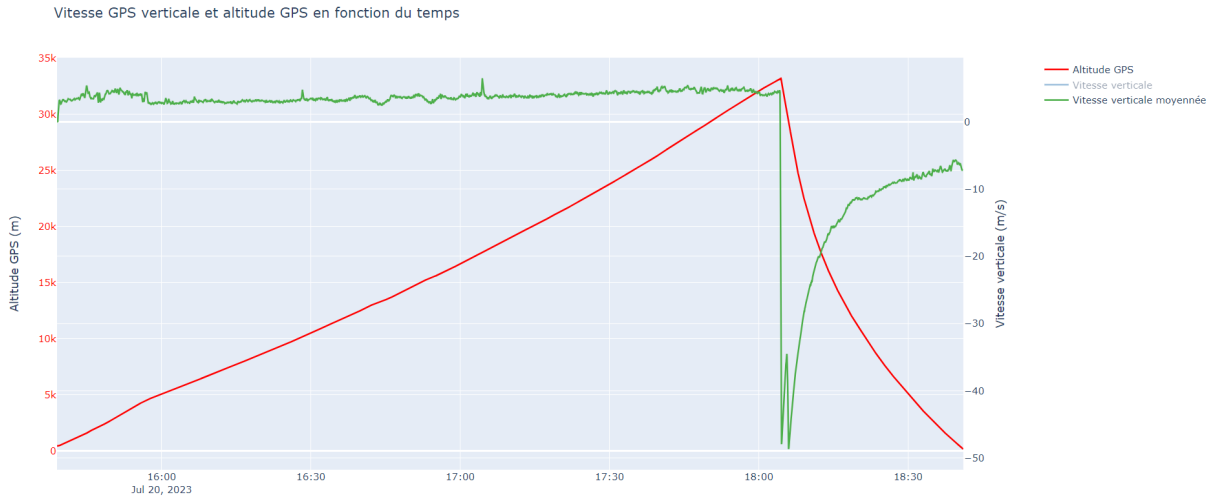


[Lien du graphique en haute résolution]

Les mesures prises à la montée et à la descente sont très similaires, ce qui montre qu'elles sont principalement influencées par l'altitude. La radioactivité monte rapidement jusqu'à descendre doucement aux alentours de 16 km d'altitude. Le phénomène mesuré ici est le maximum de Regener-Pfotzer. Nous avons également pu remarquer qu'un des deux

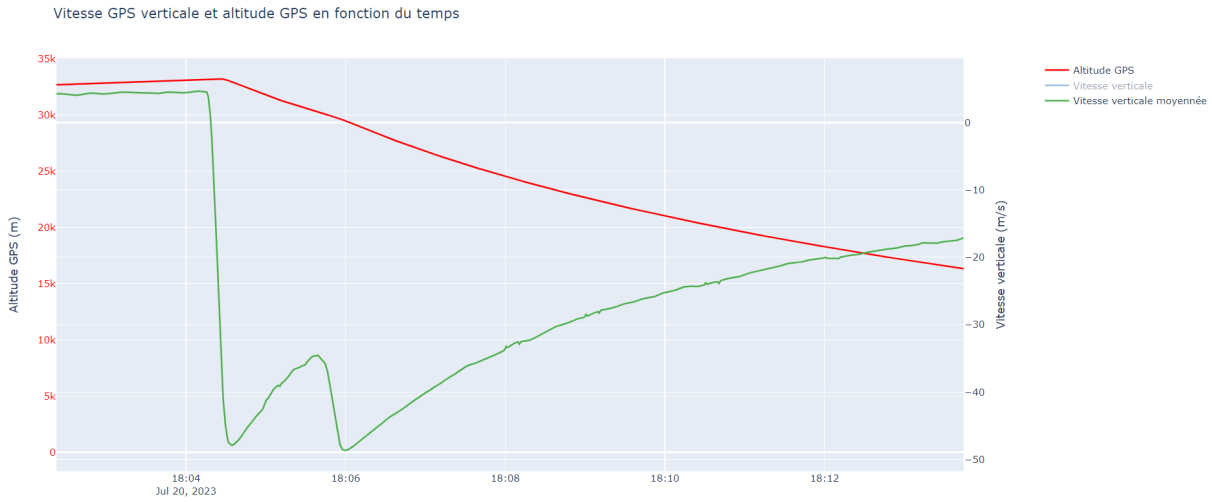
redémarrages de l'avionique est arrivé dans une zone à haute radioactivité. Mais il est difficile d'être certain que ce soit la cause sachant que le circuit de support n'a eu aucune perturbation et qu'un des redémarrages de l'avionique s'est produit proche du sol.

e. Vitesse verticale



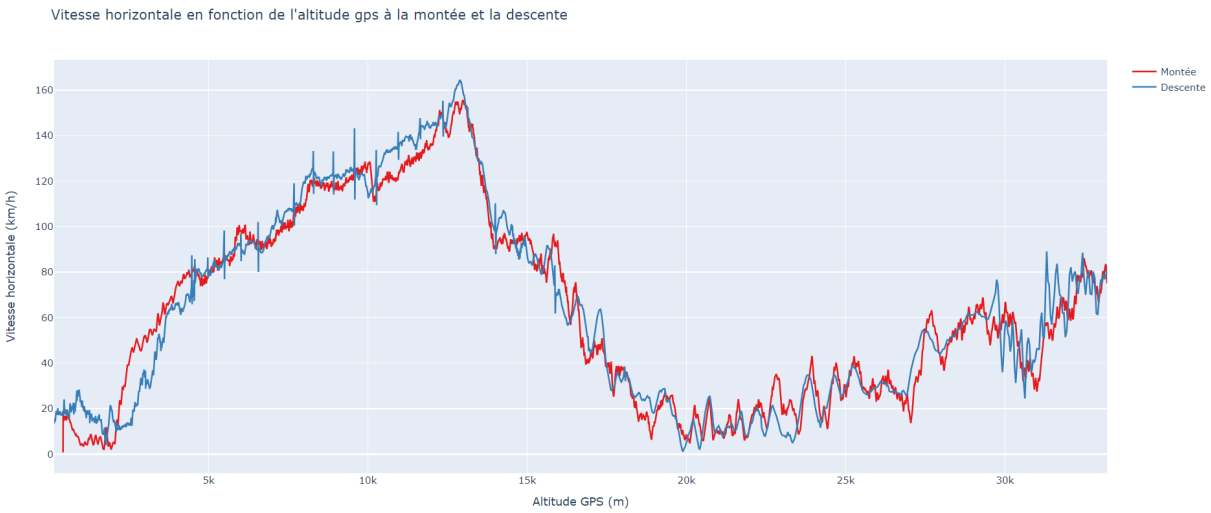
[Lien du graphique en haute résolution]

La vitesse verticale en montée a été relativement stable à environ 4 m/s.
La descente est très intéressante : on voit bien la vitesse de descente qui diminue au fur et à mesure que l'atmosphère se densifie et donc augmente les forces aérodynamiques sur la chaîne de vol. La nacelle chute à 40 m/s juste après l'éclatement du ballon et ralentit petit à petit jusqu'à 6 m/s à l'atterrissage.



2 minutes après l'éclatement du ballon on peut voir que la vitesse de descente augmente brusquement. Nous pensons que c'est le parachute qui s'est séparé de la chaîne de vol à ce moment-là. En effet, nous n'avons pas retrouvé le parachute sur le site d'atterrissage. Le réflecteur radar a suffi pour réduire suffisamment la vitesse de descente.

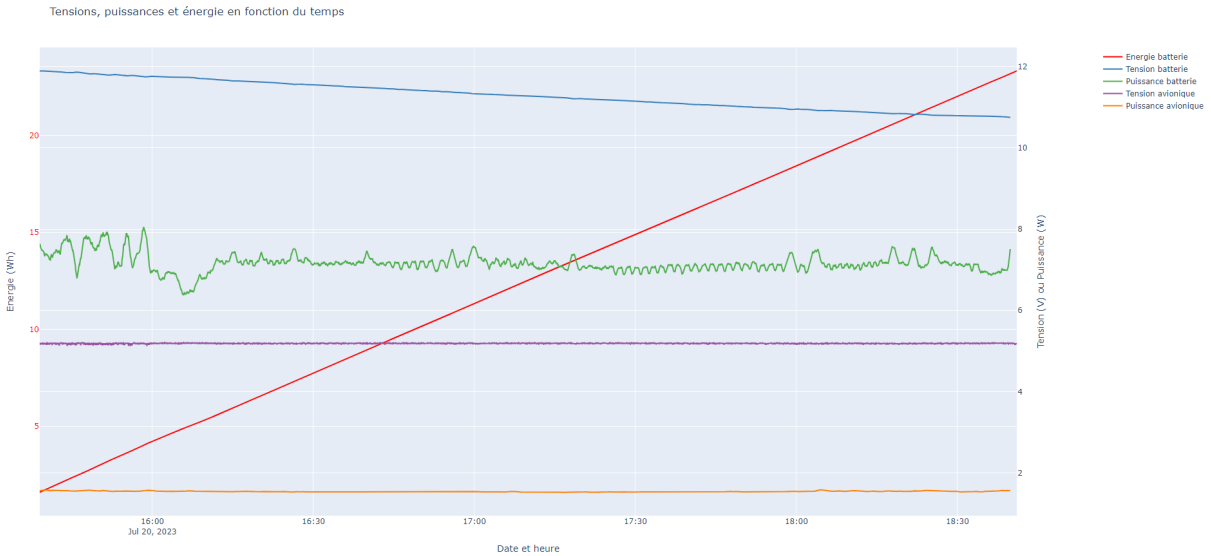
f. Vitesse horizontale



[Lien du graphique en haute résolution]

La vitesse horizontale a été en moyenne de 70 km/h avec un maximum de 165 km/h à 13 km d'altitude lors de la descente.
Les vitesses à la montée et à la descente sont très similaires, ce qui indique que le vent n'a pas changé alors que la descente s'est faite à 140 km de la montée.

g. Consommation électrique

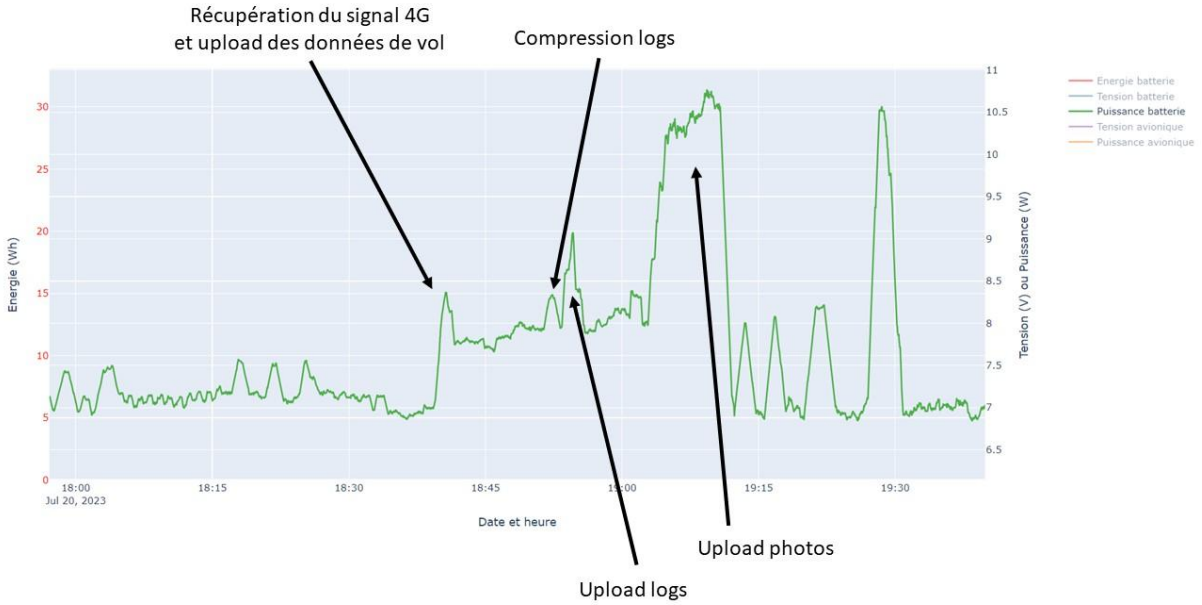


[Lien du graphique en haute résolution]

Le circuit de support comprend 2 capteurs de courant et de tension (INA219) : l'un d'entre eux connecté directement à la sortie de la batterie et l'autre entre le régulateur 5 V et l'avionique. Le premier nous permet de suivre l'état de la batterie grâce au capteur de tension et la puissance totale consommée par le ballon sonde. Le deuxième nous permet de vérifier la stabilité de la tension en sortie du régulateur et la puissance consommée par l'avionique.

On voit la courbe bleu de la tension batterie qui diminue progressivement de 11.9 V à 10.7 V. En même temps, la courbe rouge de l'énergie consommée augmente jusqu'à atteindre 23Wh à l'atterrissage, soit environ 40% de l'énergie stockée dans la batterie. Mais nous avons consommé 8 Wh de plus après l'atterrissage pour transférer le reste des données.

Au niveau de l'avionique la tension est restée stable à 5 V et sa consommation constante à 1.5 W.



Cet agrandissement de la courbe de puissance totale consommée nous permet de constater que la consommation totale du ballon dépend beaucoup de l'activité de la télémesure 4G. Après l'atterrissage nous avons pu nous connecter par SSH au circuit de support pour lui demander de nous compresser puis transférer l'intégralité des logs et des photos. Chacun de ces transferts consomme entre 1 et 3 W en plus de la consommation de 1 W du module quand il est connecté au réseau 4G.

h. Vidéo

Voici la vidéo de la montée et de la descente:
[C'Space 2023] Vidéo du lâcher du ballon stratosphérique de l'ESO
Il y a des sauts d'images toutes les minutes le temps que la caméra prenne une photo. Également, la protection anti-surchauffe s'est déclenchée entre 19km à la montée et 8km à la descente ce qui fait qu'il manque 1h10 de vidéo sur les 3h de vol.

10. Pistes d'amélioration

a. Surchauffe

Pour palier au problème de surchauffe du processeur nous aurions pu mettre un radiateur plus gros sur la Raspberry Pi 4 afin d'augmenter à la fois la dissipation par convection mais aussi l'inertie thermique. Nous avons également remarqué que l'extérieur de la nacelle ne

restait pas très longtemps en dessous de 0°C. De plus, très peu d'isolation suffit à créer un delta de température. Par exemple, la baie avionique avec 2 trous de 4 cm de diamètre était au minimum à -20°C alors que l'extérieur était à -58°C. De plus, les composants électroniques utilisés peuvent supporter des températures jusqu'à -20°C sans problème. Sachant qu'ils chauffent d'eux-mêmes il n'est pas possible qu'ils descendent en dessous de leur température critique. Nous aurions donc dû minimiser l'isolation de la nacelle et faire un compartiment plus isolé pour la batterie, car c'est la seule à être réellement sensible au froid.

b. 4G LTE Cat-1

Nos besoins en débit n'étant pas excessifs, nous aurions pu nous suffire de la technologie LTE Cat-1 qui consomme beaucoup moins donc chauffe moins et nous permet quand même d'avoir un débit montant de 5 Mbit/s (contre 50 Mbit/s avec le Cat-4) ce qui est largement suffisant.

c. Interrupteur externe

N'ayant pas eu l'expérience d'un lâcher de ballon, nous n'avions pas réalisé que fermer la nacelle au dernier moment perturbe la mise en œuvre de la chaîne de vol. Nous aurions donc dû prévoir un interrupteur général accessible depuis l'extérieur.

11. Conclusions

En conclusion, ce projet a testé avec succès des capteurs qui seront embarqués sur des fusées de l'ESO. Nous avons pu montrer que la télémesure 4G était un moyen viable pour transmettre des données de vol et un live vidéo dans un ballon sonde. Il nous a permis également de collecter beaucoup de données sur l'atmosphère terrestre.

Ce projet a été le premier ballon sonde de tous les membres, il nous a permis de gagner en expérience sur les possibilités et contraintes qui y sont associées. Nos membres sont plus que motivés pour en réaliser un pour les prochains C'Space voir même passer la formation Aéro-technicien.

12. Remerciements

Ce projet n'aurait pu être réalisé sans l'implication de Planète Sciences et du CNES, qui ont non seulement fourni l'hélium et la chaîne de vol, mais ont aussi joué un rôle crucial en nous guidant à travers sa conception et sa mise en œuvre.

Nos remerciements s'adressent également au fablab de l'ESTACA et à son responsable, M. Mangin, qui nous ont permis de travailler efficacement directement sur le campus.

Nous exprimons notre profonde gratitude à Jacques Bonfils, qui a conduit pendant 6 heures pour récupérer le ballon après son atterrissage.

Un remerciement spécial à Arthur Gerst de l'association SiERA, qui a réalisé une soudure critique et délicate sur la prise de l'antenne 4G.

Enfin, nos remerciements vont tout particulièrement à Damien Poix et Paul Miailhe de Planète Sciences, qui ont apporté un soutien continu à ce projet tout au long de l'année.