PROJET FX-05 : SCALAR V

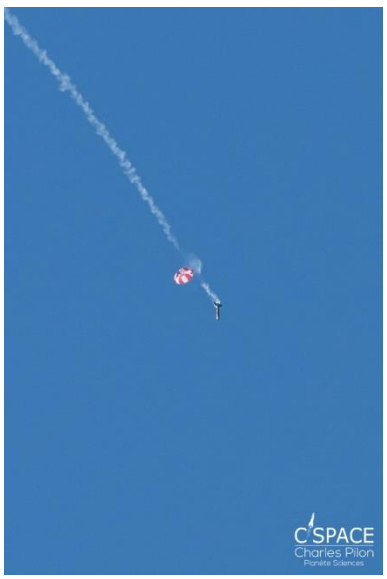
Figure 1 : Equipe SCALAR décembre 2023 & membres au CSpace

Remerciements

Nous remercions très chaleureusement notre école, l'ISAE-SUPAERO, qui nous fournit un local pour la réalisation de nos projets, ainsi que l'association des élèves de l'école qui a financé le projet.

Nous remercions également Planète Sciences Occitanie, pour le soutien qu'ils nous ont apporté tout au long du projet, ainsi que les départements DEOS (Département Electronique, Optronique et Signal) et DMSM (Département Mécanique des Structures et Matériaux) de notre école, qui nous ont aidés à réaliser cette fusée expérimentale.

Enfin nous remercions l'ensemble des bénévoles au sein de l'organisation du C'Space.



Rapport rédigé par Florian TOPEZA, Edgar VANCE, Thomas BINET, Théo PICHAVANT et Ludovic BASTIEN

Table des matières

Table des illustrations

A. Introduction

La fusée expérimentale SCALAR V a été réalisée au sein de la Supaéro Space Section de l' ISAE-SUPAERO sur les années scolaires 2022-2023 et 2023-2024. Comme son nom l'indique, Supaero CanSat LAuncheR vise à déployer des CanSats en vol. Elle a été conçue, produite, testée et lancée par l'équipe suivante :

- Ludovic BASTIEN : Chef de projet 2024
- Antoine CHAUVIN : Chef de projet 2023
- Bertrand MACE : Responsable électronique
- Florian TOPEZA : Responsable charge utile et informatique
- Edgar VANCE : Responsable mécanique et trajectoire 2023
- Simon MAUDUIT-GROUSSARD : Responsable trajectoire et récupération 2024
- Thomas BINET : Responsable mécanique 2024
- Tristan BALDIT, Caoimhe MORESMAU, Mehdi FARES : Membres du pôle CanSats
- Théo PICHAVANT, Damien BESSEDE, Martin GUYON, Pierre COORNAERT : Membres du pôle mécanique
- Valentin GRIFFON : Membre du pôle électronique
- Titouan PAYER : Membre du pôle informatique

Sans oublier la participation de Maxime LEMERLE et Victor LOIR qui ont relancé le projet SCALAR après une pause due à la crise du COVID-19.

L'objectif initial de mission du projet SCALAR V était de délivrer deux CanSats indépendants à l'aide d'une fusée expérimentale, cependant nous avons décidé en fin de projet de n'éjecter qu'un seul CanSat, tandis que l'autre expérience resterait embarquée à bord. La Fusex et le CanSat disposaient tous deux d'un parachute, déployé sur la phase finale du vol.

D'un point de vue calendaire, la phase de conception préliminaire s'est achevée en 2023. L'ensemble des pièces ont été fabriquées en 2024 et le lancement a eu lieu lors du C'Space 2024. A noter qu'un planning prévisionnel avait été réalisé (figure 2) mais qu'à partir de février 2024, il n'a que peu été suivi en raison de plusieurs retards, en particulier à cause d'un incendie dans l'atelier de l'école où nous fabriquions la majeure partie des pièces.

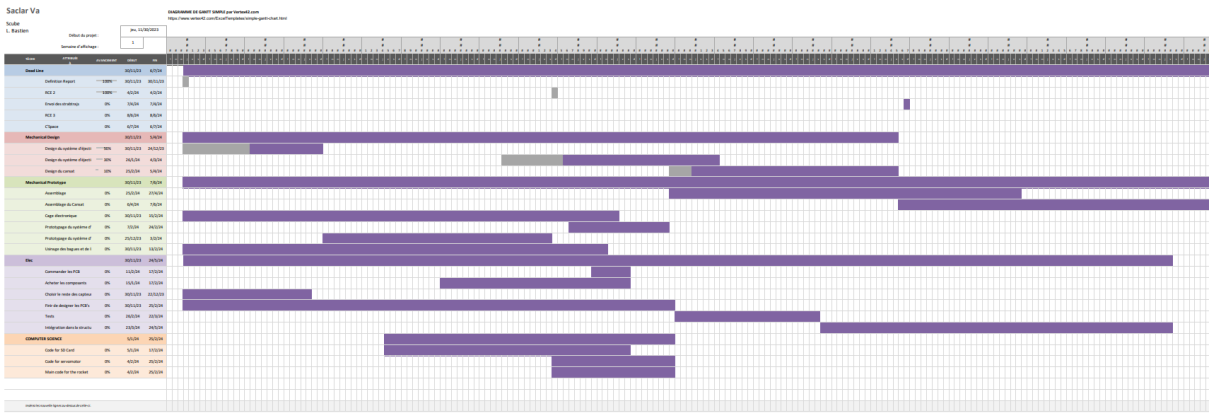


Figure 2 : planning prévisionnel

B. Description mécanique

1. Description générale

SCALAR V est une Fusex mono-étagée conçue pour éjecter des CanSats en vol. Le corps de SCALAR V est constitué de deux sections porteuses en fibre de carbone, fabriquées à partir de 3 couches de tissu de fibre de carbone préimprégné, d'un diamètre interne de 160mm et d'épaisseur inférieure à 2.5mm, jointes par une bague usinée en aluminium. Vient s'ajouter par-dessus une coiffe en fibre de verre issue de SCALAR IV, qui se sépare du corps dans le cadre d'un déploiement parachute par le haut et qui vient porter la hauteur de l'ensemble à 1922mm. Pour ce qui est des 160mm de diamètre qui donnent cet aspect trapu à la fusée, ils sont principalement hérités du passé du club : les premières fusées réalisées ayant eu ce diamètre et le matériel d'alors étant réutilisé au fil des projets. Entre la trainée et la taille importante des ailerons en conséquence, ce diamètre induit un impact significatif sur les performances aérodynamiques de SCALAR mais cela n'a pas été bloquant dans le cadre du profil de mission souhaité pour SCALAR V.

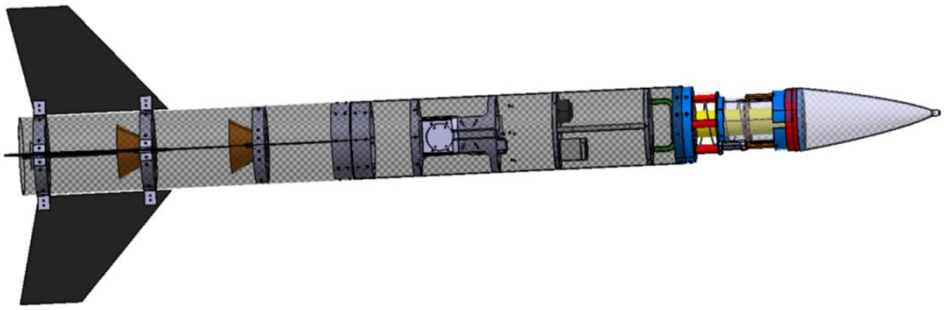


Figure 3 : Vue 3D assemblage CAO

2. Section basse

SCALAR V dispose de quatre ailerons trapézoïdaux non masqués, découpés dans des plaques de fibre de carbone de 2 mm d'épaisseur et fixés à l'aide d'équerres en aluminium. Leur taille importante, comme évoquée précédemment, est une conséquence du diamètre important du corps de la fusée. Nous avons constaté qu'entre la marge statique et le coefficient de portance des ailerons, c'est ce dernier qui dimensionne principalement leur taille.

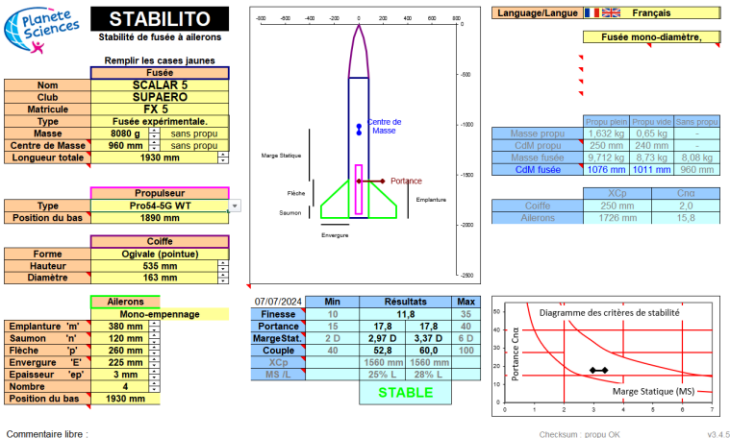


Figure 4 : Stabtraj

L'autre fonction de cette section est d'accueillir le moteur Pro 54 qui propulse la fusée. Trois bagues identiques usinées en aluminium ont été réalisées dans ce but, la plus basse servant à visser le moteur et les deux autres présentant une jupe imprimée en PLA pour faciliter l'insertion du moteur.

Figure 5 : vue bague moteur

3. Section haute

La partie supérieure du corps contient, de bas en haut, le compartiment dédié au CanSats et la cage électronique.

Le compartiment CanSats a été imprimé en PLA en un seul bloc auquel viennent s'ajouter quelques pièces, elles aussi en impression 3D, pour le mécanisme d'éjection. Sa géométrie a été réfléchie de façon à trouver un juste équilibre entre masse réduite et robustesse suffisante pour rigidifier la structure (les ouvertures pour l'éjection des CanSats induisant des efforts plus importants, qui sont partiellement repris par le bloc). Le bloc principal est vissé à la peau à l'aide d'inserts filetés. Il comprend deux cases l'une au-dessus de l'autre, où viennent se loger les charges utiles à l'horizontale. Afin d'assurer une meilleure répartition des efforts, les trappes sont disposées de façon diamétralement opposée. Les portes de chaque côté sont maintenues fermées à l'aide de loquets, qui se déverrouillent grâce à la bielle manivelle actionnée par un seul servomoteur. Des ressorts disposés au fond des cases éjectent les portes (attachées au corps à l'aide de ficelles) et les charges utiles une

fois les loquets désengagés. Ce système ne permet certes pas une intégration des plus faciles des CanSats, mais permet de n'utiliser qu'un seul servomoteur pour en éjecter deux.

Figure 6 : vue système éjection CanSats

Le compartiment contenant l'électronique embarquée occupe un tronçon dans la partie haute de la fusée, séparé du reste par 2 bagues imprimées en PLA et fixées par des vis et inserts filetés à la peau. Ces bagues sont séparées par 2 piliers, également en PLA, faisant aussi office de rails dans lesquels vient se placer verticalement l'unique carte électronique. La case contenant les batteries est placée sur la bague du bas, très proche du PCB pour pouvoir brancher les batteries par leurs câbles très courts. La case batteries est isolée par 3 plaques en plastique qui se clipsent et se déclipsent de la case elle-même, pour répondre aux exigences de facilité d'accès aux batteries. Le buzzer est fixé à la bague du haut. L'accélérocontact est lié à la partie haute du PCB et dépasse de la bague du haut. Un trou dans le côté de la bague, aligné à un trou identique dans la peau, permet le passage d'une goupille, repérée à l'extérieur par une flamme, pour empêcher le déclenchement involontaire de la séquence de vol avant le lancement.

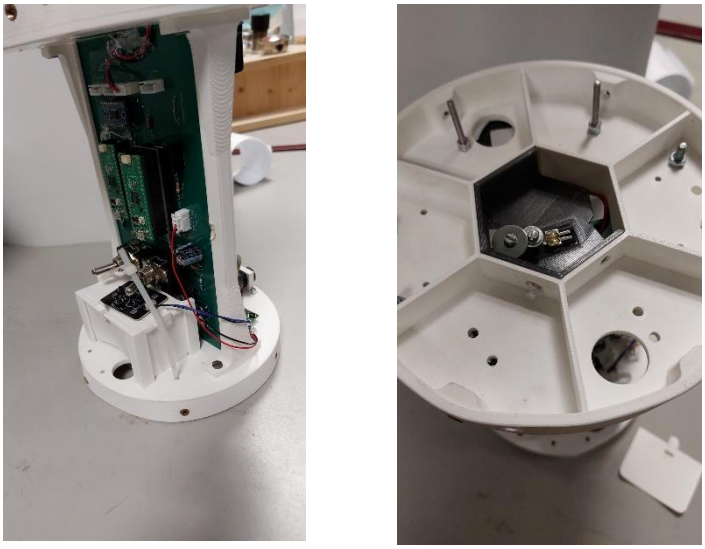


Figure 7 : Vue structure PCB

4. Coiffe et parachute

La coiffe, ainsi que le système d'éjection du parachute qu'elle contient, a été reprise de SCALAR IV. En effet, ayant déjà prouvé son bon fonctionnement et ne nécessitant qu'une légère remise en état, ce système pouvait être opérationnel bien plus rapidement qu'un nouveau design. Avec le recul cette deuxième option aurait probablement pu être possible, toutefois, le choix qui a été fait était malgré tout le bon compte tenu des nombreuses incertitudes dans les phases initiales du projet. La coiffe en fibre de verre est séparée en deux, une partie inférieure solidaire du reste du corps et une partie supérieure libérée pour déployer le parachute. Le cœur du système est un anneau à doigt fabriqué en aluminium, piloté par un servomoteur (un engrenage permet d'adapter les rotations). En position verrouillée, les doigts viennent se loger dans des encoches de la coiffe supérieure et la maintienne en position. Lors de l'ouverture, le servo désengage les doigts des encoches et ces derniers servent alors à reprendre la force d'un ressort jusque-là comprimé, qui éjecte la partie supérieure de la coiffe.

Pour assurer l'éjection du parachute nous avons joué sur les longueurs de cordes, de façon a ce que lorsque la coiffe commence à chuter, les suspentes du parachutes (qui doivent être courtes pour éviter le risque que le parachute vienne s'emmêler dans la fusée et ses ailerons qui pourraient sectionner les suspentes) extraient celui-ci en se tendant nettement avant la corde qui empêche la coiffe de tomber en chute libre.

Figure 8 : vue parachute, système éjection et coiffe

C. Description électronique et informatique

L'électronique de bord est composée d'un PCB comportant deux circuits distincts sans aucun lien électrique entre les deux. Il devait il y avoir un transfert d'information entre le coté expérience et le séquenceur pour activer l'éjection du parachute à l'apogée, mais à la suite d'une erreur de branchement, l'optocoupleur était inutilisable. Les microcontrôleurs sont des Raspberry Pi Pico, la carte étant inspirée par le projet de formation de notre club Sparrow, lui-même inspiré du projet BerryRocket.

La partie alimentation est décrite dans le schéma suivant, elle est composée de batterie 2S, de BMS et d'un régulateur afin d'avoir une sortie en 8V et une sortie en 5V.

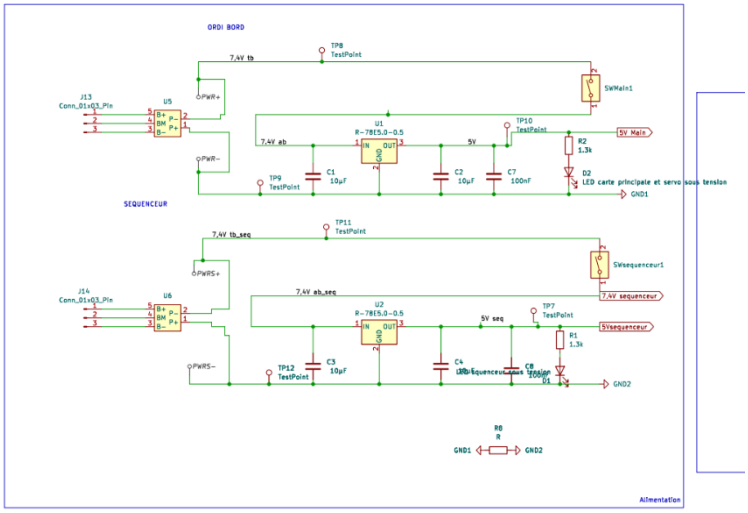
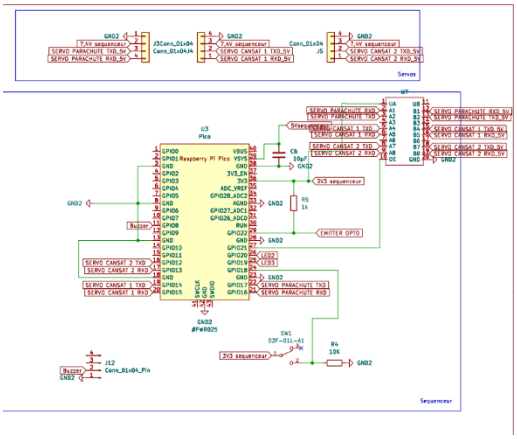


Figure 9 : alimentation pcb

Le séquenceur contrôle les servomoteurs d'éjection des CanSats et du parachute. Les servomoteurs sont des Herkulex DRS0101, ils ont besoin d'un signal de commande en 5V, d'où l'ajout d'un module qui convertit le signal logique UART sur 3,3V de la Pico en UART sur 5V.

La carte expérience est composée d'un accéléromètre MPU9250, d'un capteur de pression et de température lps22hbtr sur un shield allant sur la Pico.



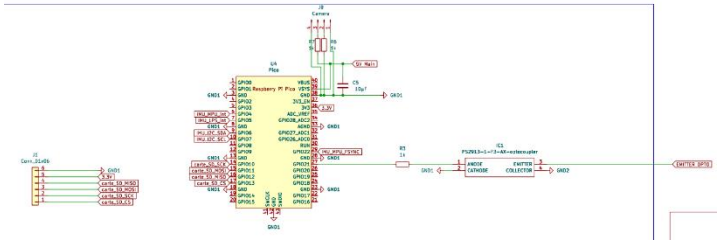


Figure 10 : schéma carte expérience

Figure 11 : Schéma séquenceur

Une fois que nous avons établi ces schématiques, nous avons réalisé le PCB suivant composé de 4 layer.

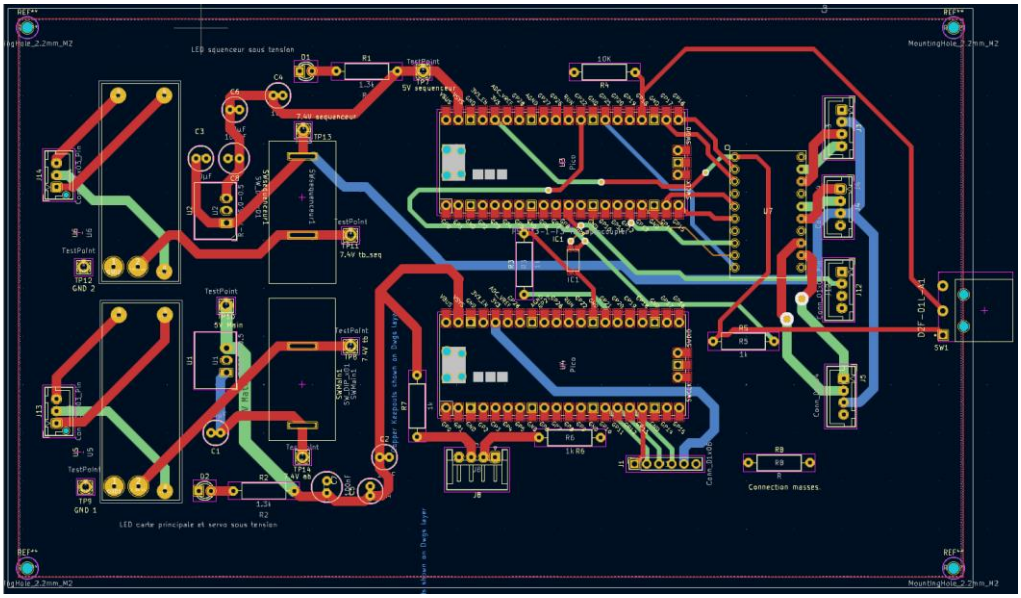


Figure 12 : PCB

Une fois le PCB reçu nous avons soudé les composants et réalisé qu'il y avait quelques erreurs de conception, notamment l'optocoupleur (IC1) qui a ses entrées inversées d'un côté et une alimentation manquante sur le convertisseur logique. Nous n'avons pas essayé de résoudre le problème de l'optocoupleur car optionnel pour le vol et nous avons rajouté un câble collé sur le PCB pour le problème d'alimentation.

Un autre problème majeur sur le projet, qui nous a donné beaucoup de fil à retordre, est la réalisation des câbles pour les servomoteurs. En effet il y avait régulièrement des faux contacts au niveau des connecteurs sur nos câble faits main, ce qui empêchait le fonctionnement fiable des servomoteurs et donc le fonctionnement du système de récupération. Ce problème fut résolu à la suite de nombreux essais, en torsadant les câbles pour limiter les interférences et assurer une transmission correcte des données en UART.

Figure 13 : vue PCB après vol

SCALAR V comporte deux microcontrôleurs Raspberry Pi Pico à programmer, l'une pour le séquenceur, l'autre pour la carte expérience. Le séquenceur gère le buzzer et l'activation des servomoteurs permettant le déploiement des CanSats et l'ouverture du parachute de la fusée, tandis que la carte expérience enregistre les données de vol d'un IMU (pression, température, accélération). Les deux cartes ont été programmées en MicroPython.

Sur la carte séquenceur, une librairie contenant toutes les fonctions pour faire fonctionner les servomoteurs Herkulex DRS0101 utilisés dans la fusée a été téléversée, ainsi qu'un code principal qui initialise la communication avec ces servomoteurs et la détection du décollage. Pour la librairie des servomoteurs, nous nous sommes appuyées sur une librairie Arduino en C++, notamment en transcrivant les fonctions nécessaires à la commande des servomoteurs en MycroPython.

Du coté de la carte expériences se trouvent deux librairies, issues du projet Sparrow (autre projet du club, visant à la formation des nouveaux arrivants, en collaboration avec Planète Science), servant à faire fonctionner l'IMU connecté à cette carte, et le code principal. Le code principal détecte le décollage et l'atterrissage d'après les variations d'altitudes et d'accélération détectées et enregistre les données de vol sur la mémoire interne de la carte expérience.

D. Expériences

Notre fusée était équipée de plusieurs expériences, d'une part le système d'éjection des CanSats, les CanSats dont le sujet est approfondi ultérieurement dans le rapport, et une carte expérience. Cette carte expérience nous a permis d'obtenir la pression, la température, et l'accélération subies par la fusée. A noter que cette carte dite expérience n'est pas le cœur de l'expérience de notre Fusex mais seulement là pour avoir des informations sur le vol. L'expérience principale est l'éjection du CanSat et l'expérience des CanSats.

E. Campagne de lancement et déroulement du vol

Nous tenons à souligner la difficulté de la campagne de lancement, constitué de nombreuses heures de travail et de validation avant le lancement. Nous pensons qu'avoir été seulement deux au C'Space pour notre fusée expérimentale était insuffisant. Nous remercions ainsi l'ensemble des bénévoles du C'Space qui nous ont donné un coup de main lorsqu'on avait besoin de plus de mains, en particulier Thomas Noguerra.

Après avoir rempli l'ensemble des critères du cahier des charges et trois vols simulés nous avons finalement obtenuel'autorisation de vol. Le premier vol simulé n'a pas été validé car notre chronologie n'était pas suffisamment détaillée, le deuxième a donné lieu à une grosse frayeur car le système d'éjection du parachute n'a pas fonctionné. Après avoir trouvé le problème (erreur de course sur le servomoteur) nous avons pu valider la fusée sur le troisième vol simulé.

Le vol a eu lieu le 10 juillet 2024, dans la matinée, avec une météo très favorable, beau temps et peu de vent. SCALAR V a décollé à 11h00 et a eu une trajectoire nominale. Nous n'avons pas relevé d'oscillations lors du vol, ce qui a été confirmé par la vidéo du suivi par la DGA. Le CanSat a été éjecté et le parachute c'est correctement déployé bien qu'il ait mis un peu de temps à sortir. Cependant le système de télétransmission n'a pas fonctionné le jour du vol, et les coordonnée GPS du CanSat n'ont donc pas pu être récupérées. Nous pouvons voir le CanSat éjecté (AéroSat) et la coiffe séparée sur la figure suivante.

Figure 14 : apogée & éjection CanSat

SCALAR V a été récupérée le soir même après les lancements, grâce au drone qui l'a repérée dans les hautes herbes et au militaire qui nous a accompagnés pour confirmer que la fusée n'avait pas atterri en zone rouge. En effet le poste de sécurité considérait qu'elle avait atterri dans cette zone et ne voulait pas qu'on récupère le lanceur. Heureusement, SCALAR V a atterri à une vingtaine de mètres en dehors de la zone rouge, nous avons donc pu la récupérer. Cependant nous n'avons pas retrouvé le CanSat, caché dans les hautes herbes et sans données GPS reçues qui auraient pu permettre de le localiser.

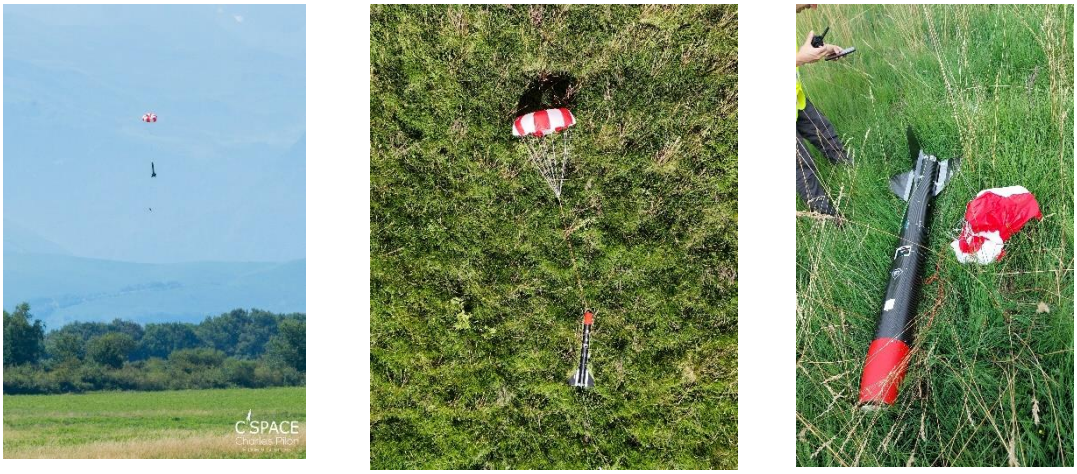


Figure 15 : récupération de la fusée

F. Résultats

Dans cette partie nous allons analyser les données de l'accéléromètre (fig. 16) et du capteur de pression qui permet le calcul de l'altitude (fig. 17). Selon nos capteurs, la fusée a ainsi eu une accélération maximale au décollage de 150m.s^{-2}, avant une ouverture du parachute à T+16s, ce qui a engendré une décélération de 10g puis une descente sous parachute et sous pesanteur terrestre classique. En ce qui concerne l'altitude, nous avons atteint 1000m. Si nous comparons avec la prévision du stabtraj (fig. 18) la performance est légèrement meilleure que celle prévue.

Nous avons une bonne confiance sur les résultats des mesures d'accélération. En effet l'erreur est minimale (moins de 1ms^{-2} vérifiable en descente sous 1g). Cependant la valeur de l'altitude à probablement une erreur non-négligeable, le capteur de pression n'étant pas forcément très précis ni étalonné dans les règles de l'art avec une station météo à proximité du lieu du lancement.

Enfin notre expérience principale, l'éjection du CanSat, est validée par vidéo et par la photo de Charles Pilon.

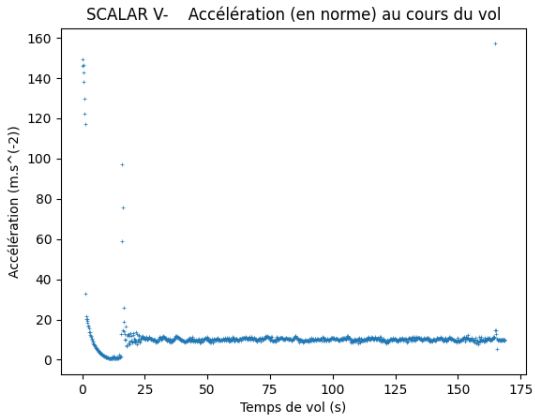


Figure 16 : Accélération en vol

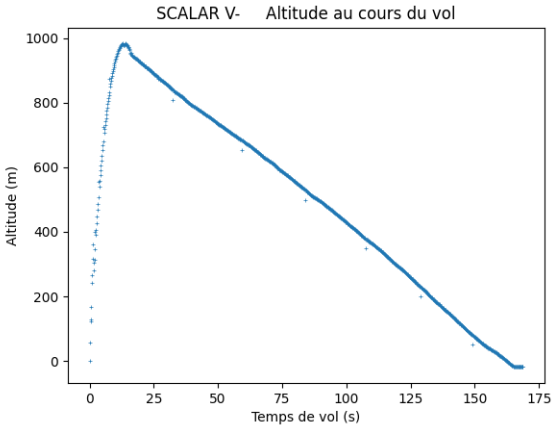
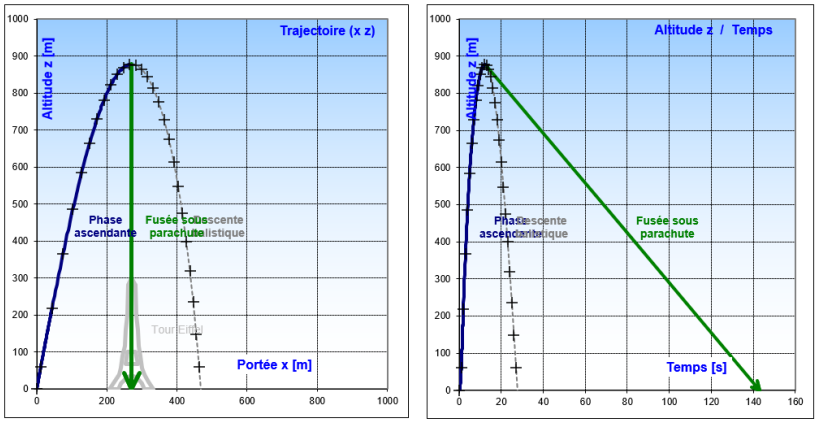


Figure 17 : Altitude en vol



07/07/2024	Temps	Altitude z	Portée x	Vitesse	Accélération	Efforts
Sortie de Rampe				30,9 m/s		
Vit max & Acc max				180 m/s	129 m/s²	
Culmination, Apogée	12,3 s	878 m	270 m	17 m/s		

Figure 18 : Prévisions trajectoire Stabtraj

G. Charge Utile

SCALAR V embarquait deux charges utiles, AéroSat et MuonSat. La première est un CanSat, éjecté de la fusée au cours du vol et redescendant sous parachute, qui comporte un détecteur d'aérosols PMS5003. L'objectif de cette expérience était de mesurer les concentrations en pollen, CO2 et particules fines dans l'atmosphère au cours de la descente. La seconde charge utile, MuonSat, n'était finalement pas un CanSat mais une expérience embarquée à bord de la fusée au cours du vol, et comprenait un détecteur de muons Cosmic Watch pour détecter les muons atmosphériques et leur énergie au cours du vol.

1. Electronique

Les deux charges utiles ont une conception électronique similaire, la seule différence étant que les connexions du PMS5003 avec la batterie et le microcontrôleur sont remplacées par une connexion à la batterie pour alimenter l'Arduino Nano du détecteur de muons. Chacune des deux charges utiles comportait un microcontrôleur (Raspberry Pi Pico), un récepteur GPS (Locosys 20031), un buzzer, un connecteur de carte micro SD (Catalex) et une batterie de 600mAh (même modèle que le projet Sparrow). Aérosat était de plus équipé d'un module de télémétrie (module Lorawan Waveshare SX1262) pour transmettre ses données GPS et du PMS5003 au cours du vol.

La batterie débitant environ 8V, un régulateur de tension continue à découpage R-78E-0.5 permet de l'abaisser à 5V. Une capacité de 10µF placée avant le régulateur et reliée à la masse permet de lisser le courant débité par la batterie avant d'entrer dans le régulateur. Sont alimentées en 5V la Raspberry Pi Pico, le connecteur de carte micro SD et le PMS5003. Le GPS est alimenté en 3,3V par la Raspberry Pi Pico et le détecteur de muons reçoit directement 8V en sortie de la batterie sur le pin VIN de l'Arduino Nano. Les deux charges utiles sont équipées d'un interrupteur ON/OFF accessible depuis l'extérieur de la charge utile.

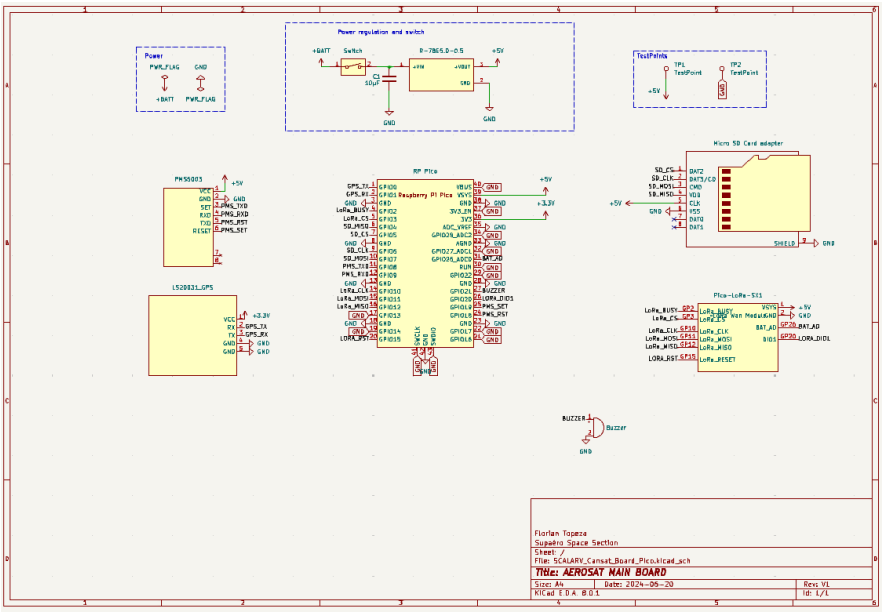


Figure 19 : *Schématique d'AéroSat*

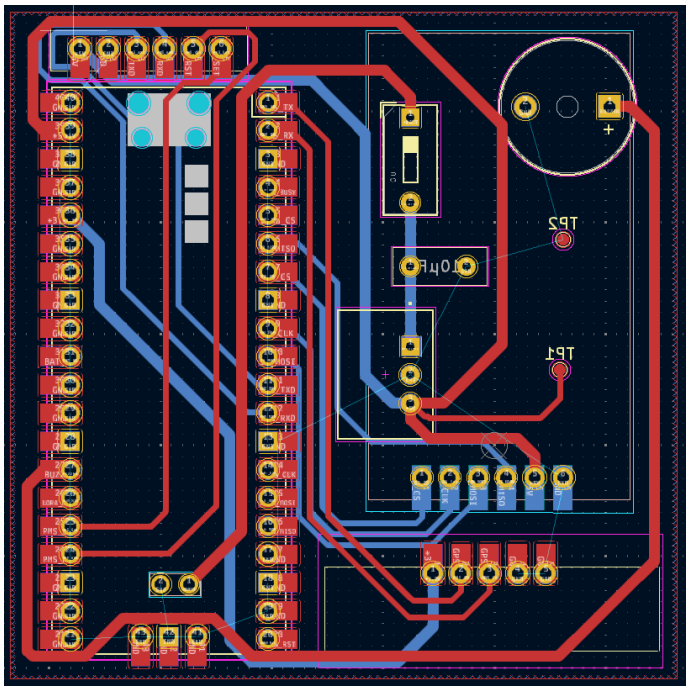


Figure 20 : *PCB d'AéroSat*

2. Informatique

La Raspberry Pi Pico a été utilisée avec des codes en MicroPython. Pour faire fonctionner la carte SD, le GPS, le buzzer, le PMS5003 et la télémétrie (ces deux derniers composants n'étant présents que sur AéroSat), le microcontrôleur utilise des librairies disponibles sur Internet, téléversées sur la carte et appelées dans le programme principal de la carte. Le fonctionnement global du programme principal (le main sur la carte) se déroule comme suit. Les librairies nécessaires sont importées, les connexions avec les autres composants (carte micro SD, GPS…) sont initialisées, puis la boucle principale (while) débute. Pour AéroSat, cette boucle est répétée jusqu'à l'atterrissage, tandis que MuonSat continue d'exécuter son programme jusqu'à l'arrêt de l'alimentation électrique. A chaque itération de cette boucle, les données GPS (ainsi que celle du PMS5003 pour AéroSat) sont écrites sur la carte micro SD. Elles sont également envoyées à la station sol dans le cas d'AéroSat.

3. Réception des données de télémétrie

La station sol se compose d'une Raspberry Pi Pico, d'un module de télémétrie identique à celui embarqué à bord d'AéroSat, et d'une carte micro SD connectée à la Raspberry Pi Pico, également avec un connecteur Catalex. La Raspberry Pi Pico est connectée à un ordinateur avec le logiciel Thonny. Les librairies nécessaires pour utiliser la carte SD et le module de télémétrie sont téléversée sur la carte et un programme est lancé depuis Thonny pour détecter la réception des données et les enregistrer sur la carte micro SD.

4. Structure

La structure des deux charges utiles ont été imprimées en 3D. Dans le cas d'AéroSat il s'agit d'une structure de CanSat complète, tandis que pour MuonSat il s'agit essentiellement de maintenir l'électronique de la charge utile en place dans son compartiment.

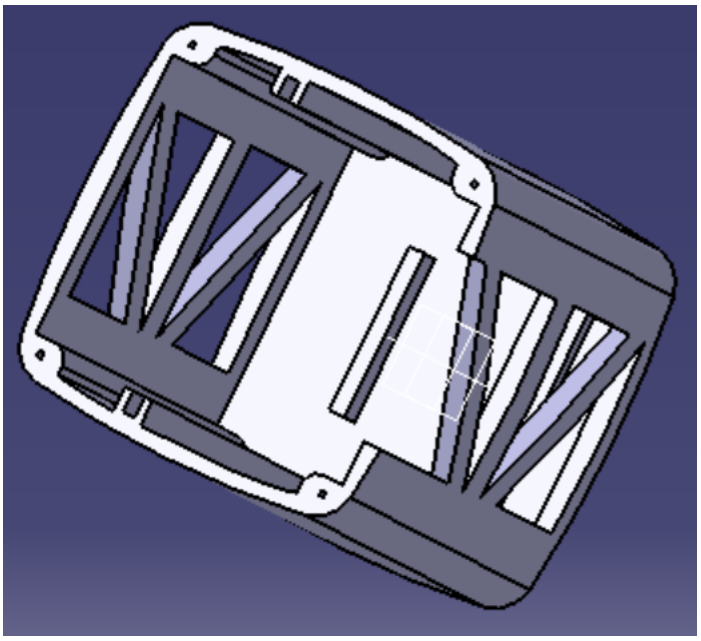


Figure 21 : *CAD d'AéroSat*

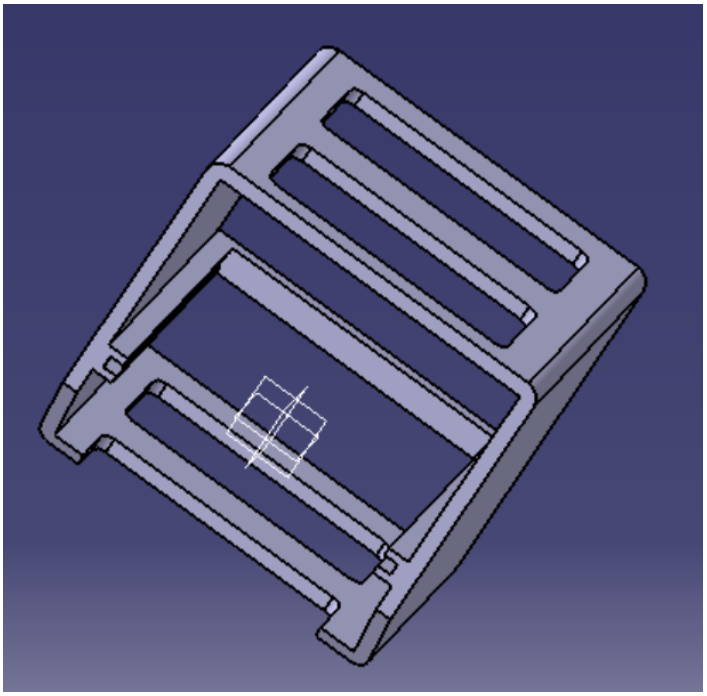


Figure 22 : *CAD de MuonSat*

5. Résultats du vol

Les deux charges utiles ont été qualifiées avec succès et ont été intégrées à la fusée comme prévu. AéroSat a été éjecté de la fusée comme attendu et son parachute s'est bien déployé. Cependant, la télémétrie n'a pas fonctionné (bien qu'elle ait été testée avec succès plusieurs fois avant le lancement et au C'Space), aucune donnée d'AéroSat n'a été reçue par la station sol. AéroSat n'ayant pas été retrouvé au sol après le vol, aucune donnée d'AéroSat n'a donc pu être récupérée. MuonSat, quant à lui, a parfaitement fonctionné et a enregistré des données pendant un peu plus de 2 heures. Cela est nettement plus court que les 3 heures d'autonomie attendues pour la batterie. Une telle différence peut s'expliquer par les hautes températures dans la fusée une fois celle-ci au sol. En effet, la fusée était exposée en plein soleil après son vol en attendant d'être récupéré, le capteur de température monté avec le détecteur de muons a enregistré une hausse constante et rapide de la température après le vol, jusqu'à 56°C dans la fusée lors de l'extinction de MuonSat.

Figure 23 : *AéroSat visible peu après son éjection*

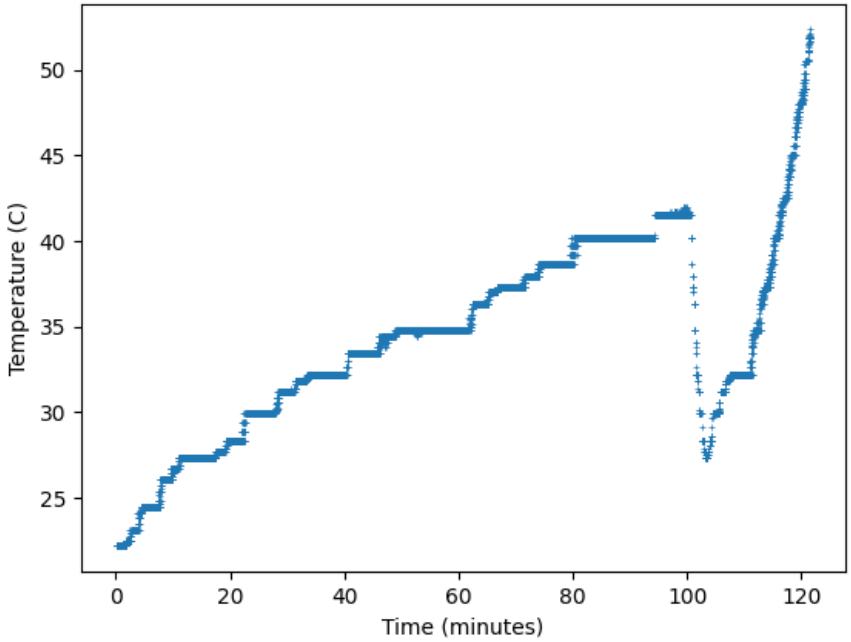


Figure 24 : Relevé de la température au niveau de MuonSat.

Le vol correspond à la baisse brutale de la température. Une fois la fusée au sol en plein soleil, la température remonte en flèche.

Les fortes températures peuvent également expliquer les déformations observées sur la structure en PLA de MuonSat après l'avoir récupéré, en plus des contraintes mécaniques auxquelles cette structure a été exposée lors du vol (notamment plus de 10g au décollage).

Figure 25 : *Structure de MuonSat après le vol*

Les données du détecteur de muons révèlent un flux de muons constant au sol, avec une diminution lors du vol. Environ 80% des muons détectés étaient d'une énergie faible, résultant en un signal

amplifié par le détecteur de moins de 100mV. A l'opposé, de rares particules très énergétiques résultaient en un signal de plus de 2V. Il n'est pas impossible que ces particules ne soient pas de muons, mais d'autres particules qui puissent être aussi détectées par le détecteur. Une étude plus approfondie permettrait peut-être d'en savoir plus sur ce sujet. Une analyse détaillée des résultats du vol, afin de notamment les corréler avec les modèles physiques existants, sera menée ultérieurement.

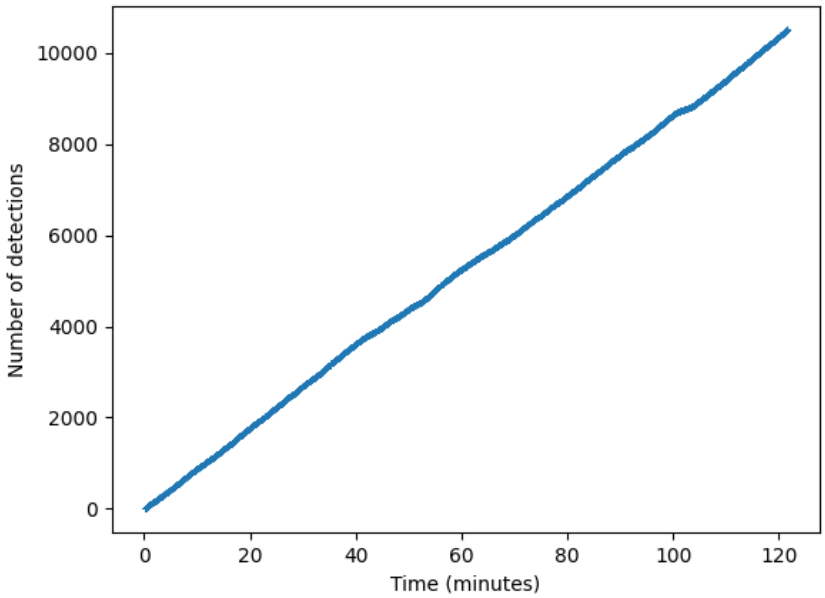
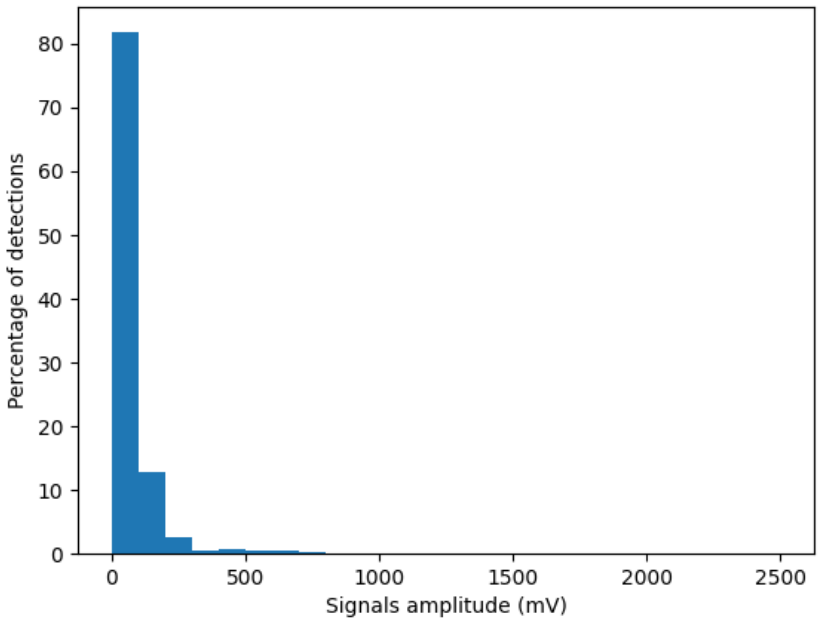


Figure 26 : Nombre total de muons détectés au cours du temps

Le vol correspond au plat vers 100 minutes.



Distribution de l'amplitude des signaux détectés

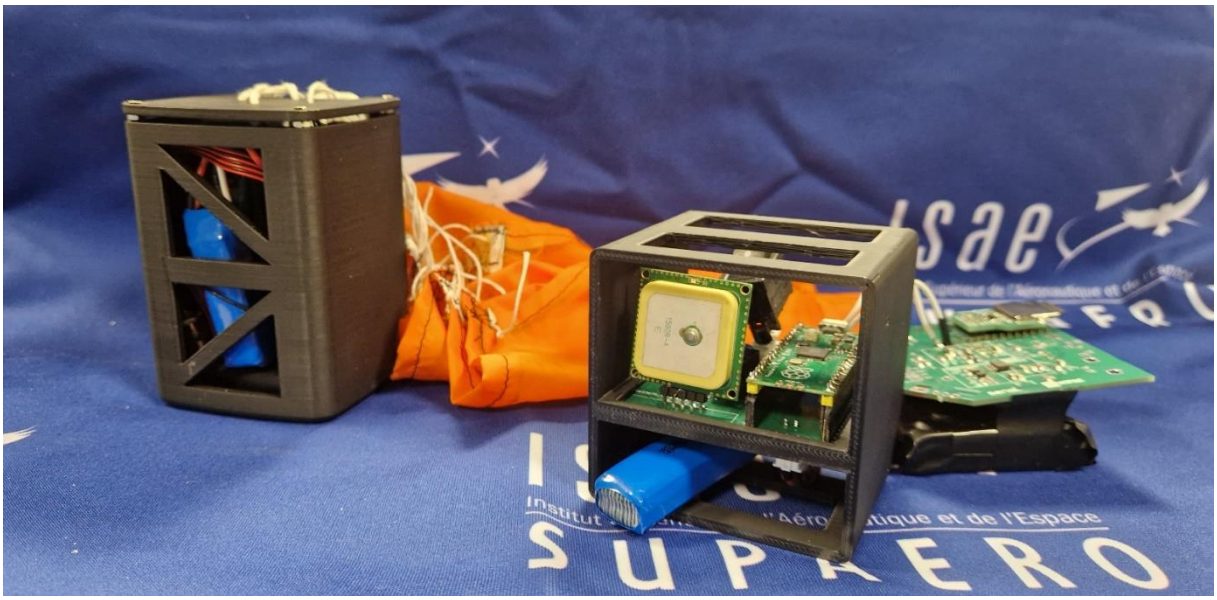


Figure 27 : *Les deux charges utiles*

H. Conclusion

SCALAR V est un projet concluant, qui s'est terminé au mieux avec un vol nominal. L'équipe est très fière d'avoir pu faire voler cette fusée expérimentale, ce qui marque le retour de la Supaéro Space Section au C'Space après cinq ans d'absence, dus à la crise du Covid-19 et des difficultés techniques sur les projets menés. L'équipe a gagné en compétences que ce soit sur la mécanique ou l'électronique, et nous sommes très motivé pour la suite. SCALAR V c'est aussi beaucoup de nuits blanches, de belles frayeurs et un défi animé par une passion qui a permis de voir cette fusée voler. En effet le projet a connu de multiples rebondissements, d'un incendie au département de mécanique de l'école dans lequel se trouve le local de notre club, au CanSat qui ne rentre plus dans le volume imposée et le choix de le garder au sein de la fusée ou encore le manque de fiabilité des servomoteurs choisis par nos prédécesseurs.

Sur les points qui ont bien fonctionné et que nous gardons pour les années suivantes, il y a la structure de la fusée, notamment la peau carbone portante et le système d'éjection de CanSats. En revanche, nous pensons abandonner l'éjection du parachute par la coiffe, le système étant trop complexe.

Les axes d'amélioration que nous voyons comportent principalement l'électronique, où nous aimerions obtenir des mesures plus précises et avoir une meilleure télémétrie.

Enfin, le projet SCALAR n'a pas prévu de s'arrêter et devrait revenir avec une fusée bi-étagée pour SCALAR6, une télémétrie et encore des expériences dans des CanSats.

Nous remercions très chaleureusement notre école, l'ISAE-SUPAERO, qui nous fournit un local pour la réalisation de nos projets et l'association des élèves de l'école qui a financé le projet.

Nous remercions également Planète Sciences Occitanie, pour le soutien qu'ils nous ont apporté tout au long du projet, ainsi que les départements DEOS (Département Electronique, Optronique et Signal) et DMSM (Département Mécanique des Structures et Matériaux) de notre école qui nous ont aidé à réaliser cette fusée expérimentale.

Enfin nous remercions l'ensemble des bénévoles au sein de l'organisation du C'Space.

Institut Supérieur de l'Aéronautique et de l'Espace ISAE-SUPAERO
Supaéro Space Section

SCALAR V Charge utile

RAPPORT FINAL

Florian Topeza
Responsable du département charge utile SCALAR V
18 octobre 2024

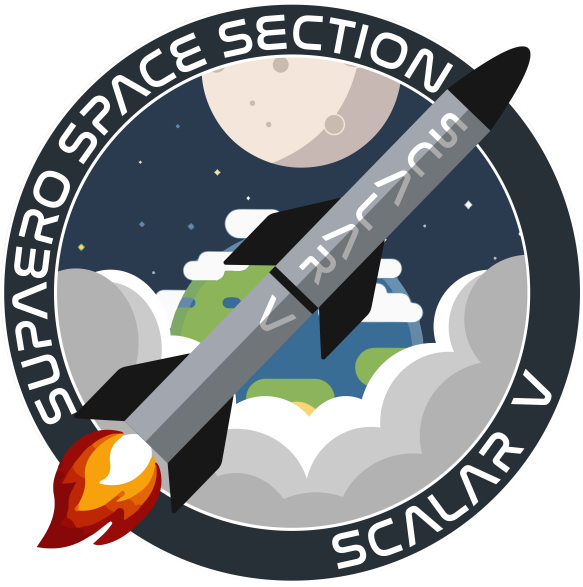


Table des matières

Table des figures

Liste des tableaux

Remerciements

Je tiens à remercier chaleureusement l'équipe charge utile ; concevoir deux charges utiles ambitieuses en 5 mois n'a pas été une tâche facile compte tenu de l'expérience technique limitée que nous avions au début du projet. Je remercie également M. Falguières, M. Beaugendre et surtout M. Berranger du DEOS de l'ISAE-SUPAERO pour leurs conseils sur la mise en place du détecteur de muons et le temps qu'ils nous ont accordé pour répondre à nos questions.

Introduction

SCALAR V est la cinquième édition du projet de fusée expérimentale SCALAR (pour Supaero CAnsats LAuncheR) de la Section Espace de Supaéro. Pour cette édition, nous avons conçu deux charges utiles à embarquer sur la fusée. AéroSat est un Cansat dont l'expérience est un capteur de matière particulaire, et MuonSat inclut un détecteur de muons à bord de la fusée. Les deux charges utiles sont équipées d'un récepteur GPS et peuvent sauvegarder leurs données sur une carte micro SD. AéroSat dispose également d'un module de télémétrie pour transmettre ses données au sol pendant le vol. Ce rapport présente en détail la conception et le fonctionnement de ces deux charges utiles.

Des ressources complémentaires telles que les fichiers CAO peuvent être trouvées dans mon dépôt GitHub.

Première partie

Gestion du projet

I.1 Organizational Breakdown Structure

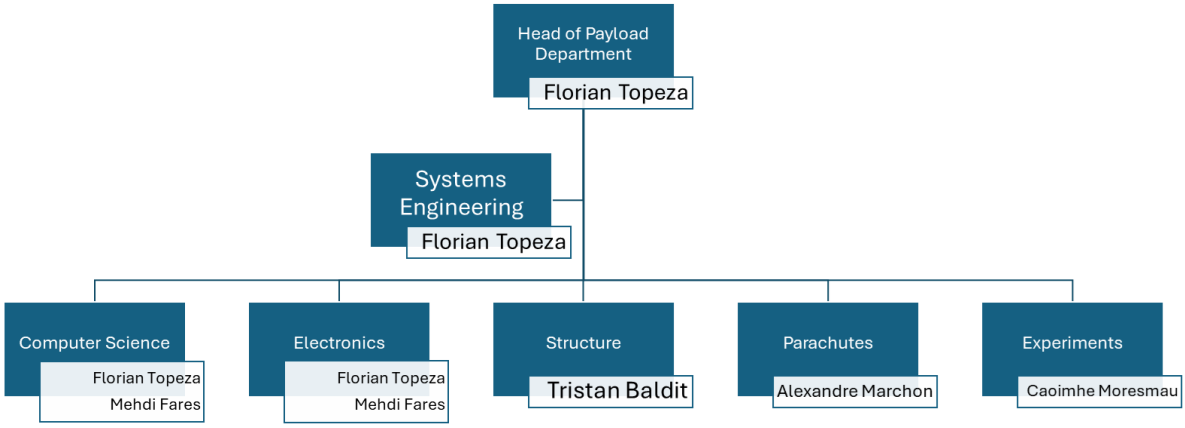


FIGURE 1 – Organizational Breakdown Structure

I.2 Product Breakdown Structure

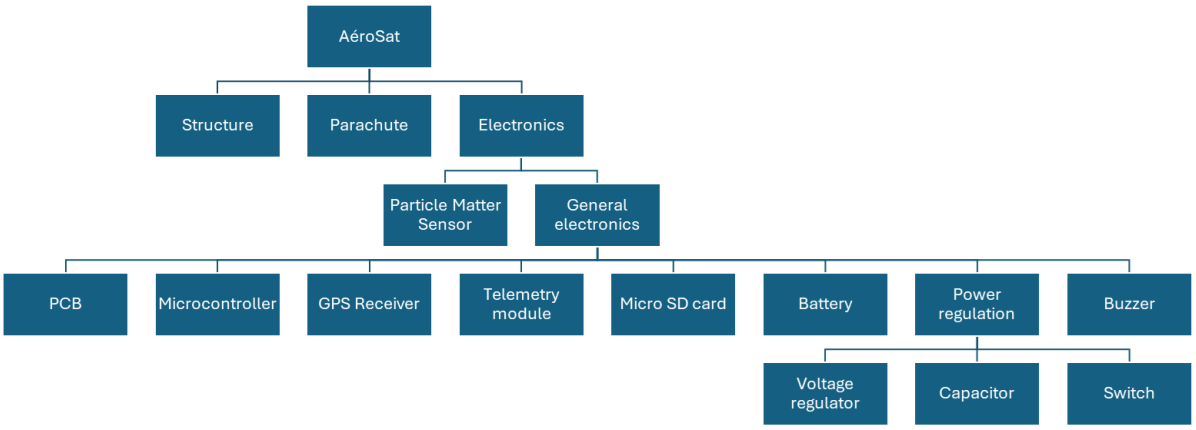


FIGURE 2 – AéroSat Product Breakdown Structure

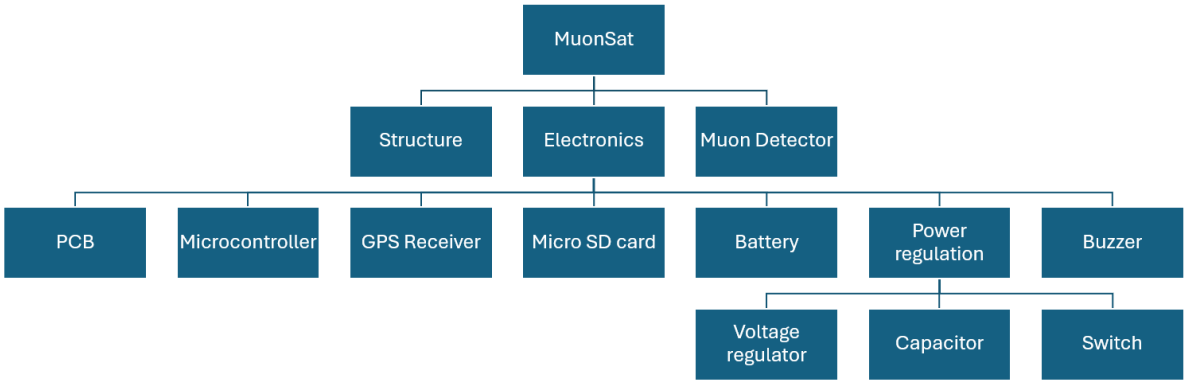


FIGURE 3 – MuonSat Product Breakdown Structure

I.3 Work Breakdown Structure

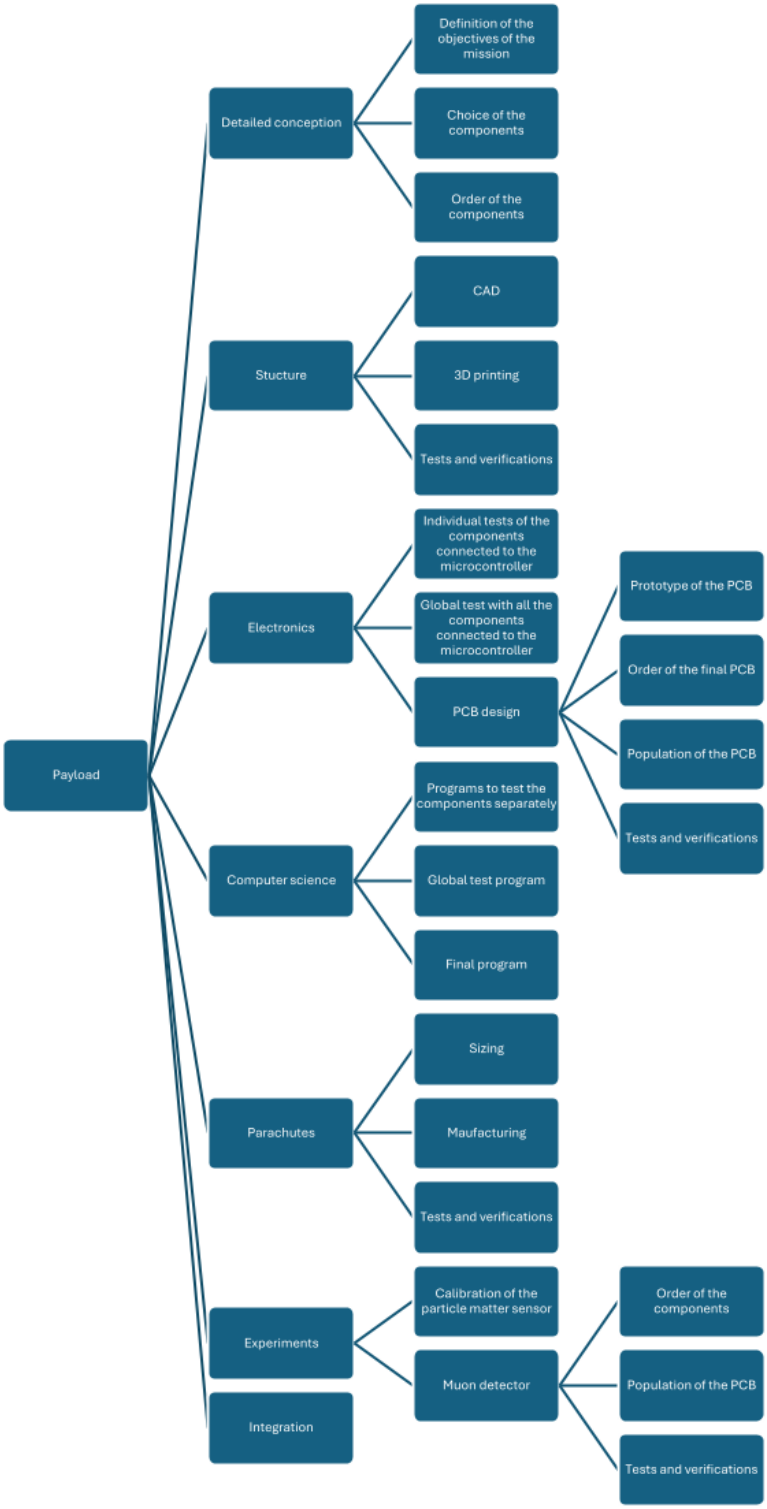


FIGURE 4 – Work Breakdown Structure

I.4 Exigences

Les exigences pour les charges utiles proviennent des exigences rédigées par le CNES et Planète Sciences pour les fusées expérimentales à un étage et à deux étages, auxquelles nous avons ajouté des exigences spécifiques pour notre projet. Les exigences du CNES et de Planète Sciences sont disponibles sur le site de Planète Sciences.

I.4.1 Exigences liées aux objectifs de l'expérience

Exigence	Description	Conformité
OBJ1	Le club doit définir les objectifs de l'expérience.	AéroSat mesure l'altitude et la concentration des PM1.0, PM2.5, PM10, PM1.0, PM2.5, PM10, >0.3um dans 0.1L d'air, >0.5um dans 0.1L d'air, >1.0um dans 0.1L d'air, >2.5um dans 0.1L d'air, >5.0um dans 0.1L d'air, >10um dans 0.1L d'air pendant le vol. MuonSat mesure l'altitude et enregistre toute détection de muons. À chaque détection, le détecteur indique l'heure de la détection, la tension amplifiée SiPM et le niveau correspondant sur l'ADC du circuit.
OBJ2	Le club doit déterminer les paramètres à analyser.	Pour AéroSat, nous traçons les concentrations des particules en fonction du temps de vol et de l'altitude. Pour MuonSat, nous traçons le nombre total de détections dans le temps et la répartition des amplitudes des signaux SiPM et ADC.
OBJ3	Le club doit définir les plages de mesure et les précisions requises.	Les plages et précisions du capteur de particules sont indiquées dans sa fiche technique. Le détecteur de muons est capable de détecter les muons qui émettent au moins une dizaine de photons lorsqu'ils traversent le scintillateur plastique.

OBJ-METH1	Le club doit indiquer les paramètres effectivement mesurés.	Pour le capteur de particules, la concentration des différents types de particules est mesurée, corrigée selon la calibration. Pour le détecteur de muons, l'heure de détection, la tension amplifiée SiPM et le niveau ADC sont mesurés.
OBJ-METH2	Le club doit déterminer les fonctions de conversion entre les paramètres à mesurer et les paramètres effectivement mesurés.	Pas de conversion pour le détecteur de muons ? Nous utilisons une régression linéaire pour le capteur de particules.
OBJ-METH3	Le club doit déterminer les incertitudes de mesure.	Pour le capteur de particules, l'erreur de cohérence maximale est indiquée dans la fiche technique. Aucune information n'est fournie sur les incertitudes du détecteur de muons et nous n'avons pas eu le temps de les calculer, ce qui nécessiterait d'examiner les circuits du détecteur, de trouver les fiches techniques de ses composants et de calculer le gain du circuit d'amplification en tenant compte des incertitudes de chaque composant pour obtenir l'incertitude sur la tension SiPM. Nous aurions également besoin de connaître l'incertitude de l'ADC.
OBJ-METH4	Le club doit, au début du projet, déterminer comment utiliser les résultats de l'expérience. Il doit notamment définir comment décoder la télémétrie et comment l'utiliser, ainsi que prédire les résultats attendus.	Seule AéroSat utilisera la télémétrie. Elle est enregistrée dans un fichier CSV, à partir duquel nous extrayons les données pertinentes pour tracer les courbes. Toutes les données des deux charges utiles sont enregistrées dans des documents CSV et TXT sur une carte micro SD, qui peuvent être facilement traitées en Python.

TABLE 1 – Exigences liées aux objectifs de l'expérience

I.4.2 Exigences liées à la définition de la chaîne de mesure

Exigence	**Description**	**Conformité**
DEF1	Le club doit choisir les liens dans la chaîne de mesure (capteurs, conditionneurs, codeurs, etc.) pour atteindre les performances requises.	Performance d'AéroSat basée sur les performances des capteurs choisis, ainsi que pour MuonSat.
DEF2	Le club doit planifier les méthodes de calibration pour les différents canaux de mesure.	Pas besoin de calibrer le détecteur de muons, il est calibré dans son programme Arduino. Pour calibrer le capteur de particules, nous comparons ses mesures aux données fournies par une station de mesure officielle à Toulouse et calibrons le capteur avec une régression linéaire.
MES1	Tous les canaux de mesure doivent être calibrés.	Détecteur de muons auto-calibré, méthode de calibration du capteur de particules décrite ci-dessus.
MES2	La chaîne de mesure globale doit avoir une autonomie d'au moins 1 heure avec l'électronique allumée pendant la montée et au moins 3 heures pour le reste.	Les tests effectués ont montré une autonomie d'au moins 3 heures pour les deux charges utiles.
TEL1	Les points de test et les cavaliers doivent être présents entre chaque élément de la chaîne de télémétrie.	Pas nécessaire.
TEL2	Les systèmes de transmission de données doivent être capables de résister à une interruption de transmission.	Les interruptions de transmission ne doivent pas se produire. Si cela se produit, le système peut redémarrer sans problème.
TEL3	Le club doit démontrer qu'il est capable de décoder les données reçues.	Tests effectués.
TEL4	Si les données transmises sont en dehors de la bande 20 Hz - 20 kHz, le club doit fournir un moyen de stocker les données reçues.	Carte micro SD si nous recevons les données avec un Raspberry Pi Pico, ou ordinateur si nous les recevons directement sur un ordinateur.

TEL5	L'émetteur doit être capable de transmettre les données de l'expérience dans de bonnes conditions, conformément aux règlements internationaux en matière de télécommunications. Cette condition est remplie si un émetteur fourni par PLANETE SCIENCES est utilisé correctement. Pour le Kiwi, cela inclut : - L'alimentation de l'émetteur doit être comprise entre 7,5V et 14V. - La tension de modulation doit être comprise entre 0,1V et 5V crête à crête. Si le club utilise deux émetteurs KIWI simultanément, PLANETE SCIENCES et le CNES doivent être informés au moins 2 mois avant C'Space. Si le club utilise une fréquence GSM, elle doit se faire via le réseau GSM pour lequel l'émetteur a une carte SIM. Il est interdit d'utiliser directement les fréquences GSM. Le club doit également se conformer à la règle SECU4. Si le club utilise un autre émetteur, la fréquence et la puissance de transmission doivent être indiquées dans le dossier de conception.	Bande de fréquence LoraWan EU433.

TEL6	Les fréquences utilisables et les puissances HF transmises doivent être les suivantes : - Plus de 150 mW lors de l'utilisation d'un émetteur KIWI (137,05 et 137,5 MHz) ; - Moins de 10 mW pour la bande de fréquence 433,05 MHz à 434,79 MHz ; - Moins de 25 mW pour la bande de fréquence 868 MHz à 869,2 MHz ; - Moins de 500 mW pour la bande de fréquence 869,4 MHz à 869,65 MHz. L'EIRP doit être : - Moins de 100 mW pour la bande de fréquence 2400 MHz à 2483,5 MHz pour les systèmes à large bande (bande WiFi) ; - Moins de 500 mW pour la bande de fréquence 5470 MHz à 5725 MHz pour les systèmes à large bande (bande WiFi) ; La bande 144-146 MHz peut être utilisée à condition qu'un radioamateur licencié soit présent pendant les transmissions.	Dans le programme, nous limitons la puissance du Lorawan à 10 dBm pour respecter cette exigence.
TEL7	L'utilisation de bandes de fréquence non mentionnées dans la règle TEL6 ou non incluses dans les bandes de fréquence GSM est interdite.	Lorawan respecte la règle TEL6.
TEL8	L'émetteur doit avoir sa propre alimentation, avec un interrupteur indépendant des autres interrupteurs. L'autonomie de l'émetteur doit être d'au moins 1 heure.	Non nécessaire, donc non implémenté.
TEL9	Le VSWR (Voltage Standing Wave Ratio) doit être inférieur à 2 (à la fréquence de transmission).	Lorawan satisfait cette exigence.

TEL10	Tout lien montante doit être limité à l'implémentation. Le lien montante doit être désactivé à la fin de l'implémentation par le club avant de procéder aux opérations pyrotechniques.	Pas de lien montante.
TEL11	Les émetteurs doivent être éteints pendant les opérations pyrotechniques. Cette inhibition doit être levée par un dispositif pouvant être activé à distance par les pyrotechniciens (par exemple, Jack).	La conception des charges utiles satisfait cette exigence.

TABLE 2 – Exigences liées à la définition de la chaîne de mesure

Exigence	**Description**	**Conformité**
EXP1	L'expérience doit faire l'objet d'un rapport détaillé.	Un rapport de vol sera rédigé.

TABLE 3 – Exigences liées à l'analyse des résultats de l'expérience

I.4.3 Exigences liées à l'intégration de l'expérience

Exigence	**Description**	**Conformité**
INT1	Les charges utiles doivent s'adapter à leurs compartiments dédiés (100x70x70 mm chacun).	Les charges utiles sont conçues pour satisfaire cette exigence.
INT2	La masse totale des charges utiles doit être inférieure à 1 kg.	Masse totale de 600 g.
INT3	Les charges utiles doivent avoir un interrupteur ON/OFF accessible depuis l'extérieur des charges utiles.	La conception des charges utiles satisfait cette exigence.
INT4	Les charges utiles doivent avoir leur propre source d'alimentation.	Chaque charge utile a sa propre batterie et système de régulation de l'alimentation.

TABLE 4 – Exigences liées à l'intégration de l'expérience

I.4.4 Exigences liées à la récupération de l'expérience

Exigence	**Description**	**Conformité**
REC1	Le Cansat doit être équipé d'un parachute permettant une vitesse de descente comprise entre 5 et 10 mètres par seconde.	Parachutes dimensionnés pour satisfaire cette exigence.

TABLE 5 – Exigences liées à la récupération de l'expérience

I.5 Budget

Nous avons essayé de maintenir le budget aussi serré que possible en réutilisant des composants déjà disponibles dans les locaux du club, provenant de projets plus anciens ou de ressources disponibles. La seule dépense inutile est l'achat des récepteurs GPS L76, car nous avons finalement utilisé des récepteurs Locosys 20031 trouvés dans les locaux du club, bien que nous utilisions le code du L76 pour faire fonctionner le LS20031.

Composant	**Coût**
Composants du détecteur de muons	300€
PCB du détecteur de muons	17€
Gel optique pour détecteur de muons	17€
Module de télémétrie et GPS	100€
Composants généraux (buzzer, connecteur de capteur de particules, etc.)	64€
PCB des charges utiles	17€
Total	**515€**

TABLE 6 – Budget des charges utiles

I.6 Impact environnemental

Il est difficile d'obtenir une estimation précise de l'empreinte carbone du projet, car nous avons commandé des composants auprès de différents fabricants aux États-Unis et en Chine. De plus, l'empreinte carbone du projet ne se compose pas seulement de celle des livraisons, mais aussi des ressources utilisées. Une première estimation consiste à considérer que nous avons commandé 1 kg de composants en provenance de Chine. Les composants ont été envoyés en France par avion, ce qui émet environ 500 g de CO2 par tonne par km. Une fois en France, ils ont été livrés par camion, ce qui ajoute 60 à 150 g de CO2 par tonne par km. La distance entre la France et la Chine est d'environ 9000 km, et nous considérons que la distance moyenne de livraison en France par camion est de 100 km.

Ces chiffres conduisent au calcul suivant pour l'empreinte carbone : $0.001 \times 9000 \times (0.5 + 0.06) = 4,506$ kg de CO2.

Nous avons essayé autant que possible d'utiliser des composants déjà détenus par le club. Ainsi, les microcontrôleurs, les récepteurs GPS finaux (Locosys 20031), les connecteurs de carte SD, les interrupteurs et les batteries étaient déjà en possession du club. Nous avons également fabriqué nous-mêmes les parachutes, avec des ressources du club.

Deuxième partie

Conception électronique

II.1 Composants matériels

II.1.1 Microcontrôleur

Nous utilisons un Raspberry Pi Pico comme microcontrôleur pour chacun des payloads. Ce choix est basé sur le fait que le club en possédait déjà, nous avions déjà de l'expérience avec ces microcontrôleurs grâce au projet Sparrow de la Section Espace de Supaéro, ils offrent de bonnes performances compte tenu de leur petite taille et ils sont programmés avec MicroPython, une variante de Python, qui est plus facile à utiliser que le C ou le C++.

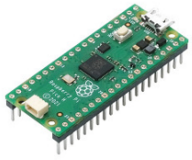


FIGURE 5 – Microcontrôleur Raspberry Pi Pico

II.1.2 Récepteur GPS

Au départ, nous avions l'intention d'utiliser des récepteurs Waveshare L76B, car ils étaient conçus pour le Raspberry Pi Pico et pouvaient être empilés en dessous, ce qui limiterait l'espace occupé par les récepteurs dans les charges utiles. Cependant, une fois que nous les avons reçus, nous nous sommes rendu compte que leurs antennes étaient plus longues et plus grandes que prévu, ce qui rendrait leur stockage difficile dans les charges utiles. Nous avons donc finalement opté pour les récepteurs GPS Locosys 20031 qui possédaient des antennes intégrées. Nous les avons trouvés par hasard dans les locaux du club et les avons fait fonctionner avec le programme initialement prévu pour le L76B. Cela est possible parce que les deux récepteurs utilisent la même puce MediaTek MT3339 et envoient des données avec le protocole NMEA0183. Les récepteurs GPS sont alimentés en 3V3 à partir du microcontrôleur et communiquent avec celui-ci via UART. Nous utilisons un débit en bauds de 57600, qui est le débit par défaut du Locosys 20031.

FIGURE 6 – Récepteur GPS Locosys 20031

II.1.3 Module de télémétrie

Seul AéroSat est équipé de télémétrie. Nous utilisons des modules Waveshare Pico-LoRa-SX1262 car ils sont conçus pour le Raspberry Pi Pico et parce que le Lorawan permet des communications efficaces (jusqu'à 10 km en terrain dégagé) à faible consommation. Pour recevoir les données, nous utilisons une station de base composée d'un autre module SX1262, connecté à un Raspberry Pi Pico, lui-même connecté à un ordinateur via le port USB. Le Pico recevant les données est également connecté à un connecteur de carte micro SD, afin que nous puissions sauvegarder les données reçues sur la carte SD. Les deux modules Lorawan (sur AéroSat et sur la station de base) sont connectés à leur Pico respectif via SPI. Nous limitons la puissance des modules à 10 pour nous conformer aux exigences du CNES qui limitent la puissance à 10 dBm. Le connecteur de carte micro SD de la station de base est également connecté à son Pico via SPI, avec un débit en bauds de 100000.

FIGURE 7 – Module de télémétrie Lorawan utilisé sur AéroSat

II.1.4 Carte micro SD

Nous utilisons des adaptateurs de carte Micro SD Catalex et des cartes Micro SD que nous avons trouvés dans les locaux du club. Sur les deux charges utiles, elles sont connectées au Pico via SPI 0 et utilisent les GPIO 4, 5, 6 et 7, avec un débit en bauds de 100000. Elles sont alimentées en 5V directement à partir du régulateur de tension.

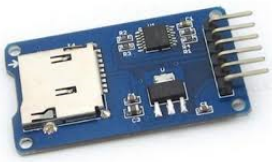


FIGURE 8 – Connecteur de carte Micro SD

II.1.5 Batterie

Nous utilisons des batteries LiPo qui offrent de bonnes performances pour des petites batteries. Nous avons d'abord recherché des batteries d'une capacité de 1000mAh en nous basant sur les estimations du pire cas de la consommation électrique des composants électroniques de la charge utile. Cependant, des tests ont montré qu'une batterie d'une capacité de 600mAh était suffisante pour alimenter AéroSat pendant plus de 3 heures. Nous avons donc opté pour les batteries LiPo utilisées par le projet Sparrow, d'une capacité de 600mAh.

FIGURE 9 – Batterie de 600mAh

II.1.6 Régulation de l'alimentation

Les batteries LiPo que nous utilisons fournissent entre 7,4 et 8V. Le Raspberry Pi Pico, le capteur de particules et le connecteur de carte micro SD fonctionnent avec 5V, tandis que le récepteur GPS nécessite 3,3V. C'est pourquoi nous utilisons un convertisseur DC/DC R-78E-0.5 qui prend la tension de la batterie en entrée et la régule à 5V pour alimenter le Raspberry Pi Pico, le connecteur de carte micro SD et le capteur de particules. Le récepteur GPS est alimenté par la sortie 3,3V du Pico. Nous plaçons un condensateur de 10µF connecté à la masse et à l'entrée du régulateur pour stabiliser le courant provenant de la batterie avant qu'il ne pénètre dans le régulateur. Le détecteur de muons est alimenté directement par la batterie, car il possède son propre microcontrôleur, un Arduino Nano, qui accepte jusqu'à 12V sur son broche d'entrée VIN.

FIGURE 10 – Convertisseur DC/DC R-78E-0.5

II.1.7 Buzzer

Nous utilisons un petit buzzer piézo passif de 5V compatible avec une plaque d'essai sur chaque charge utile, qui est contrôlé par le Pico. Il fonctionne avec PWM et est connecté au GPIO 21 du Pico.

FIGURE 11 – Buzzer de 5V compatible avec une plaque d'essai

II.1.8 Capteur de Particules

Le Capteur de Particules a été choisi par un ancien étudiant qui a travaillé sur le projet, il s'agit du PMS5003. Nous utilisons un breakout PIM477 qui convertit le câble de connecteur picoblade du capteur de particules PMS5003 en un connecteur mâle standard à pas de 2,54 mm pour connecter le capteur au PCB. Le PMS5003 communique avec le Pico de AéroSat via UART avec un débit en bauds de 9600.

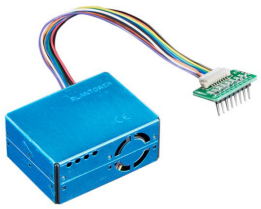


FIGURE 12 – PMS5003

II.1.9 Détecteur de muons

Nous utilisons un Détecteur de Muons Cosmic Watch pour MuonSat, qui est un détecteur développé par d'anciens étudiants du MIT à assembler soi-même. Nous n'avons pas construit l'enveloppe du détecteur, sinon elle aurait été trop grande pour tenir dans le compartiment du payload. Nous avons commandé les composants séparément et soudé le PCB nous-mêmes. Pour plus d'informations sur le détecteur, veuillez consulter le site Web Cosmic Watch.

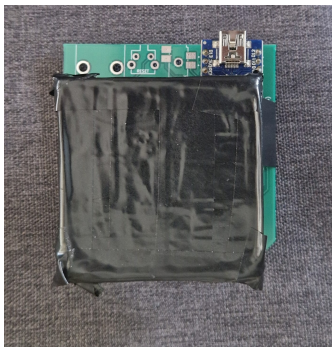


FIGURE 13 – Vue du dessus du détecteur de muons

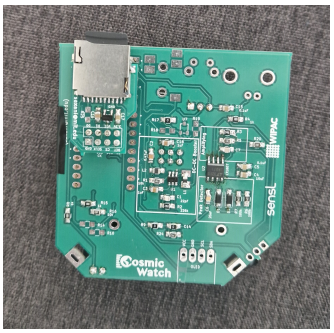


FIGURE 14 – Vue du dessous du détecteur de muons

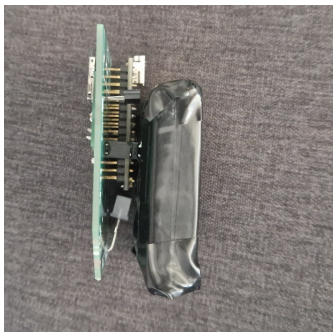


FIGURE 15 – Vue de gauche du détecteur de muons

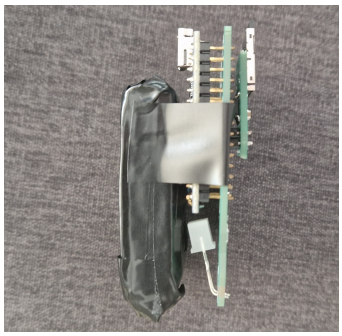


FIGURE 16 – Vue de droite du détecteur de muons

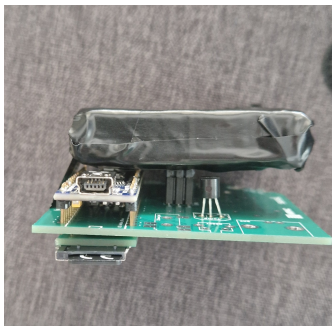


FIGURE 17 – Vue de l'avant du détecteur de muons

II.2 Conception des PCB

II.2.1 Processus de développement

Chaque fois que nous ajoutions un nouveau composant à nos charges utiles, nous testions son fonctionnement avec le microcontrôleur, et surtout comment il devait être alimenté. Nous avons donc testé le GPS, le module de télémétrie, le buzzer, le connecteur de carte SD et le capteur de particules individuellement pour nous assurer qu'ils fonctionnaient correctement. Ces tests ont été réalisés avec les composants connectés au microcontrôleur à l'aide de câbles et d'une breadboard.

Une fois les composants testés séparément, nous avons réalisé un test global avec tous les composants connectés au microcontrôleur, en simulant les opérations de vol réelles des charges utiles.

La réalisation de ce test global a permis de valider le brochage des microcontrôleurs et la répartition de l'alimentation des charges utiles. Nous avons donc pu passer à la phase de prototypage du PCB, c'est-à-dire à la fabrication d'un PCB avec la graveuse disponible à l'ISAE-SUPAERO. Cette graveuse ne pouvait fabriquer que des PCB à 2 couches de cuivre, c'est pourquoi nous n'avons pas pu l'utiliser pour le PCB final. En effet, nous ne voulions pas de plaques de cuivre exposées sur le PCB final sans couches de sérigraphie pour des raisons de sécurité (les parties métalliques en contact avec des couches de cuivre exposées pourraient provoquer des courts-circuits) et également pour des raisons de performance (les couches de cuivre exposées pourraient réfléchir le signal

transmis par le module de télémétrie et ainsi compromettre son fonctionnement).

Enfin, une fois le design du PCB validé avec le prototype, nous avons pu commander en toute confiance la version finale auprès d'un fabricant.

II.2.2 Schémas

Voici les schémas que nous utilisons pour le PCB final des Cansats. Cependant, il y a une erreur que nous n'avions pas remarquée avant de commander le PCB. Les broches 3V3_EN et RUN du RP Pico (respectivement la quatrième et la onzième broche à droite du RP Pico) ne doivent pas être connectées à la masse, mais laissées non connectées ; sinon, le RP Pico ne peut pas exécuter son programme.

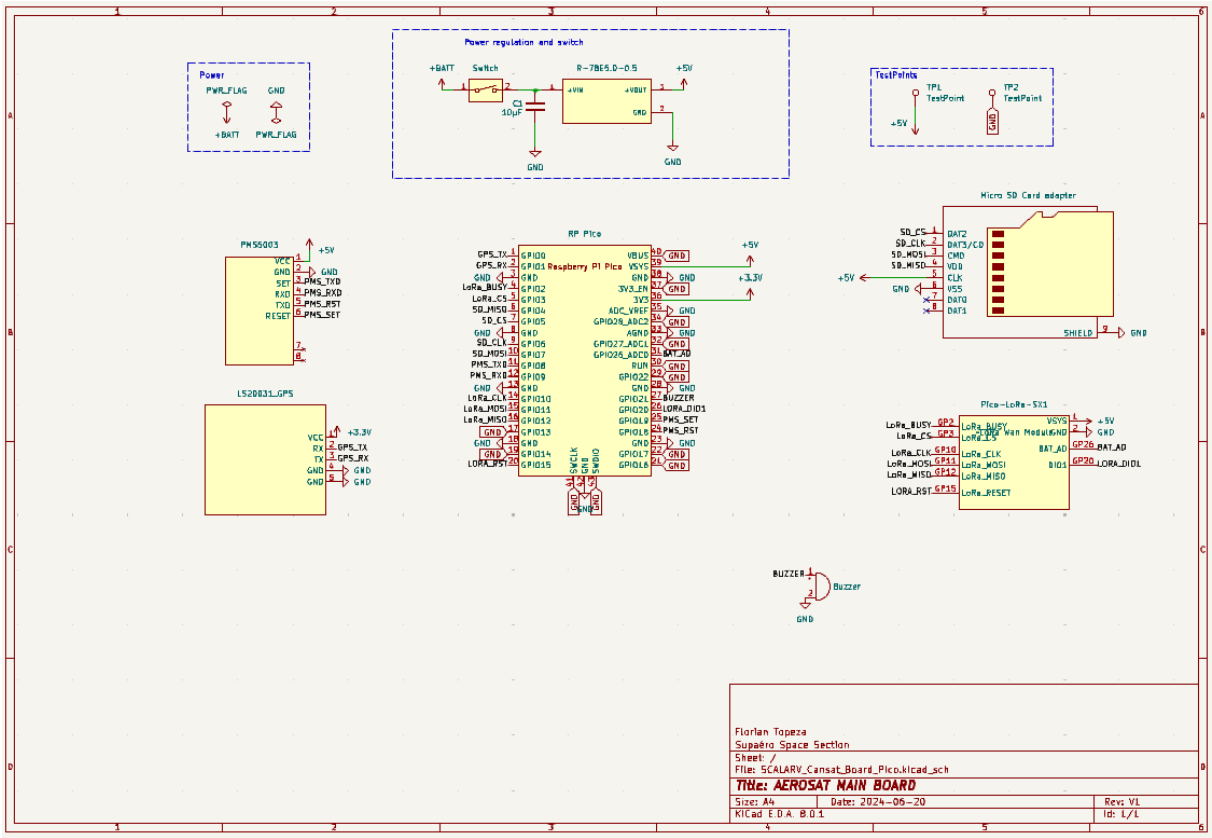


FIGURE 18 – Schémas de l'AéroSat

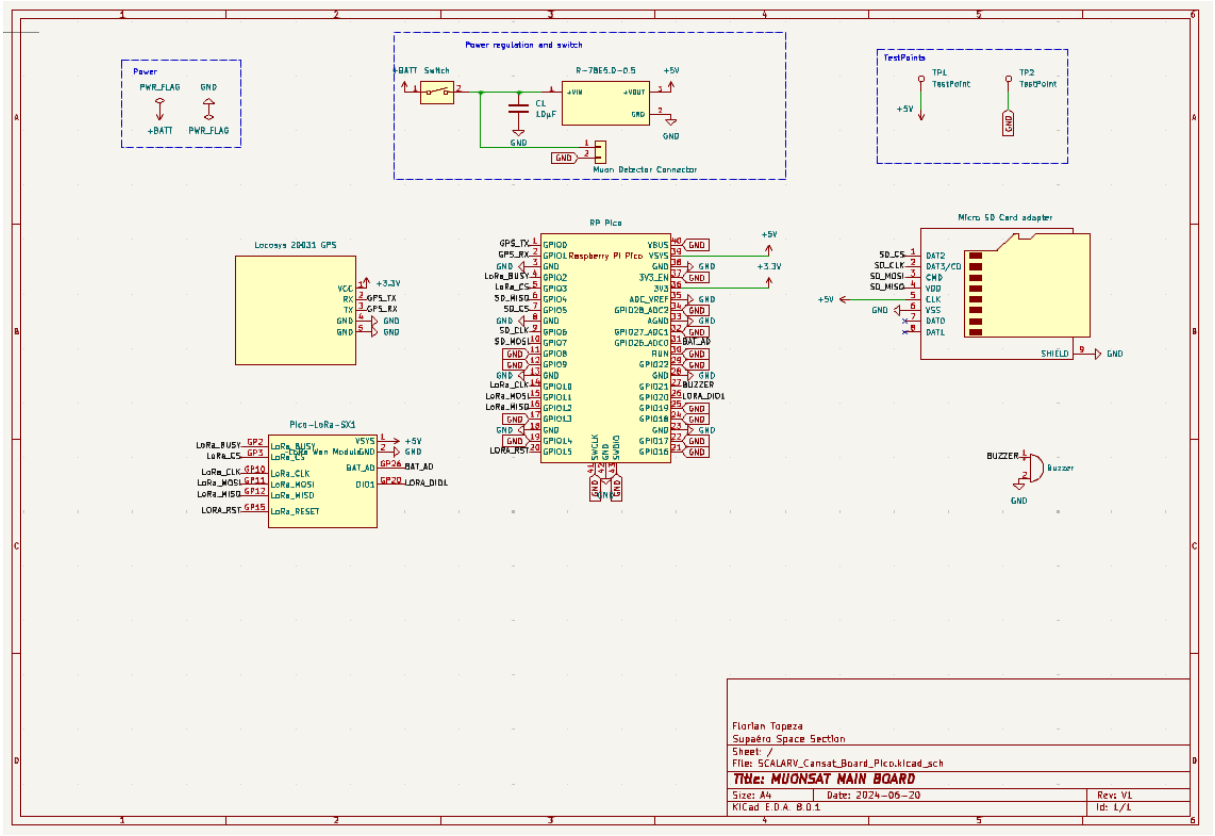


FIGURE 19 – Schémas du MuonSat

Broche Raspberry Pi Pico	**Composant connecté**
GPIO 0	GPS TX
GPIO 2	Lorawan Busy
GPIO 3	Lorawan CS
GPIO 4	SD MISO
GPIO 5	SD CS
GPIO 6	SD CLK
GPIO 7	SD MOSI
GPIO 10	Lorawan CLK
GPIO 11	Lorawan MOSI
GPIO 12	Lorawan MISO
GPIO 15	Lorawan RESET
VSYS	+5V Input
3V3	+3,3V Output
GPIO 21	BUZZER
GPIO 20	Lorawan DIO1

TABLE 7 – Configuration des broches du Raspberry Pi Pico pour AéroSat

Broche Raspberry Pi Pico	**Composant connecté**
GPIO 0	GPS TX
GPIO 4	SD MISO

GPIO 5	SD CS
GPIO 6	SD CLK
GPIO 7	SD MOSI
VSYS	+5V Input
3V3	+3,3V Output
GPIO 21	BUZZER

TABLE 8 – Configuration des broches du Raspberry Pi Pico pour MuonSat

II.2.3 Conceptions finales

Voici les conceptions finales des cartes. L'erreur mentionnée dans la section des schémas s'applique également ici, les broches 3V3_EN et RUN auraient dû être laissées non connectées au lieu d'être connectées à la masse.

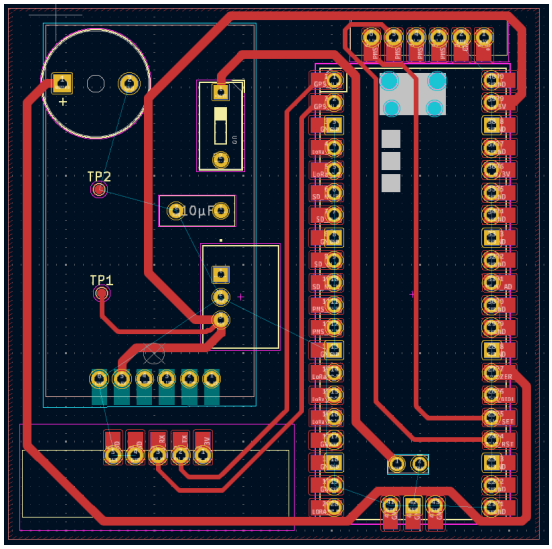


FIGURE 20 – Vue de la carte AéroSat par le dessus avec uniquement la couche de cuivre avant

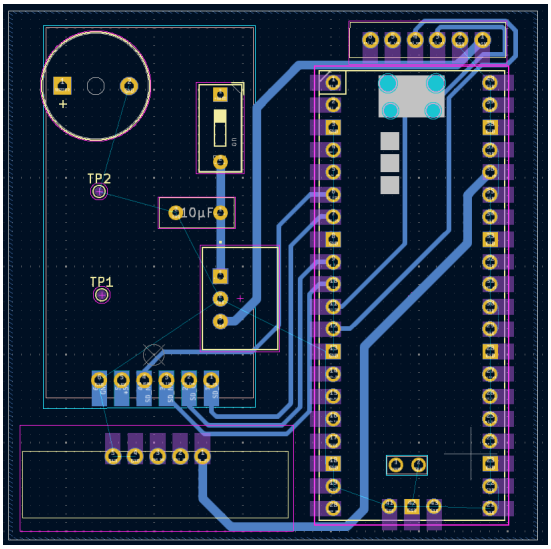


FIGURE 21 – Vue de la carte AéroSat par le dessus avec uniquement la couche de cuivre arrière

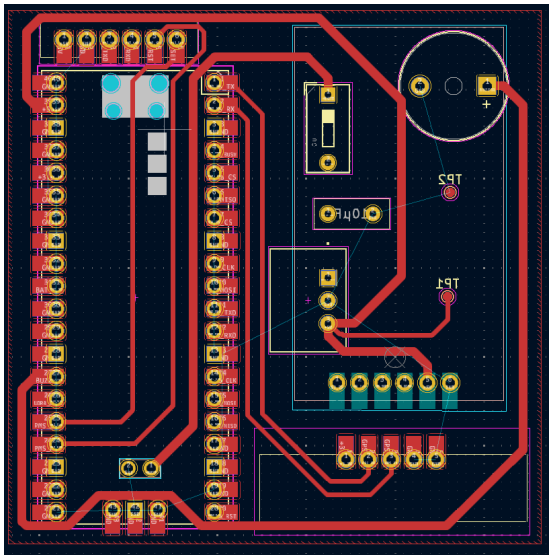


FIGURE 22 – Vue de la carte AéroSat par le dessous avec uniquement la couche de cuivre avant

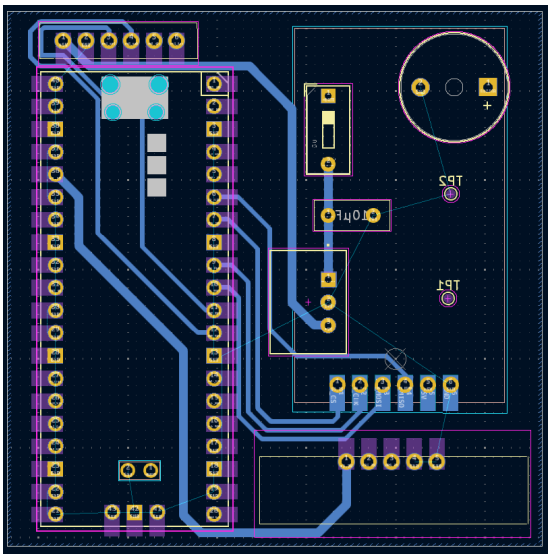


FIGURE 23 – Vue de la carte AéroSat par le dessous avec uniquement la couche de cuivre arrière

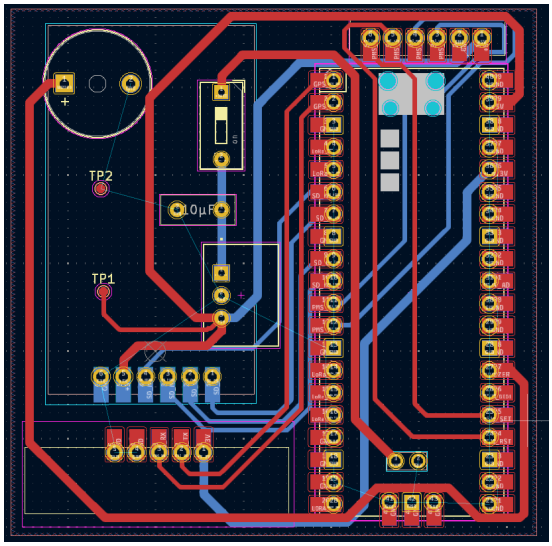


FIGURE 24 – Vue de la carte AéroSat par le dessus avec toutes les couches de cuivre

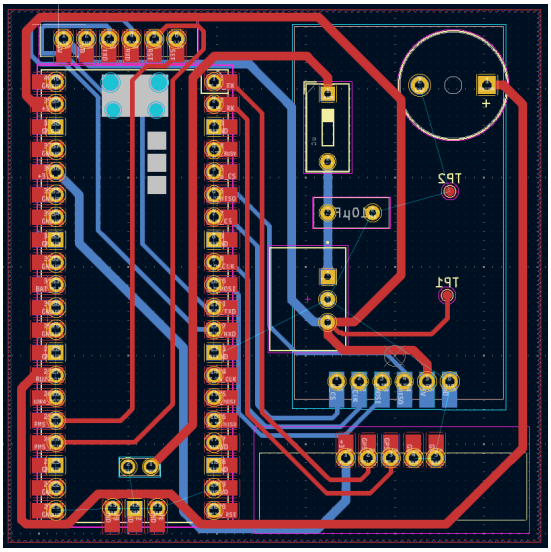


FIGURE 25 – Vue de la carte AéroSat par le dessous avec toutes les couches de cuivre

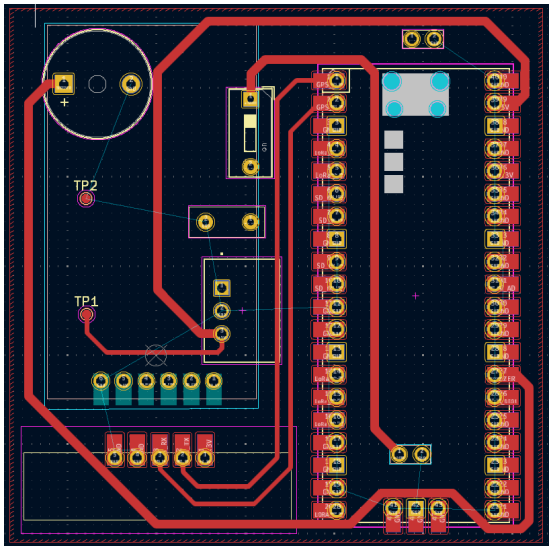


FIGURE 26 – Vue de la carte MuonSat par le dessus avec uniquement la couche de cuivre avant

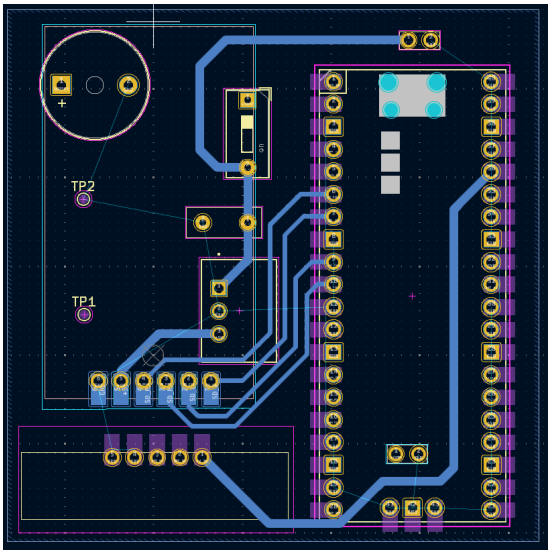


FIGURE 27 – Vue de la carte MuonSat par le dessus avec uniquement la couche de cuivre arrière

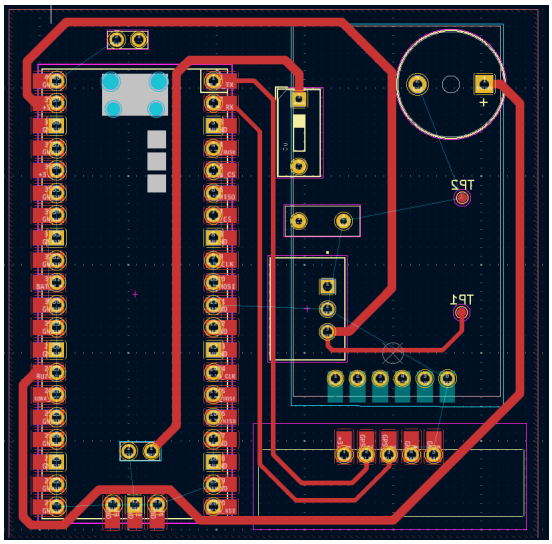


FIGURE 28 – Vue de la carte MuonSat par le dessous avec uniquement la couche de cuivre avant

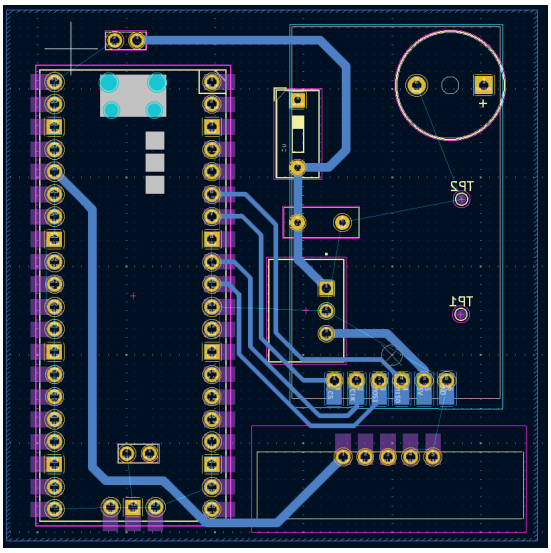


FIGURE 29 – Vue de la carte MuonSat par le dessous avec uniquement la couche de cuivre arrière

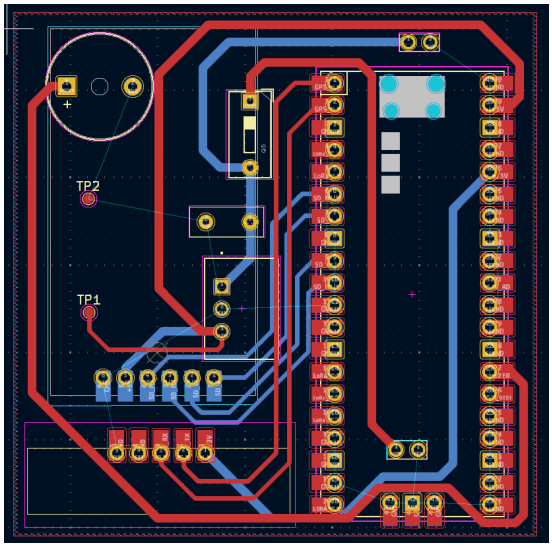


FIGURE 30 – Vue de la carte MuonSat par le dessus avec toutes les couches de cuivre

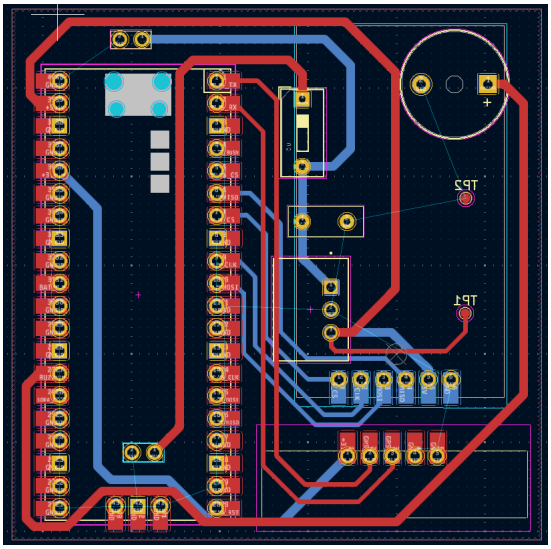


FIGURE 31 – Vue de la carte MuonSat par le dessous avec toutes les couches de cuivre

Troisième partie

Conception de l'ordinateur de bord

III.1 Bibliothèques

Pour utiliser les composants des charges utiles, tels que la carte SD ou le GPS, avec le Raspberry Pi Pico, nous avons besoin de bibliothèques qui fournissent les fonctions utilisées par le programme principal.

La bibliothèque pour le module LoRaWAN est composée de trois fichiers : sx126x.py, sx1262.py et _sx126x.py. Ils peuvent être téléchargés ici.

La bibliothèque pour le récepteur GPS est celle du récepteur Waveshare L76B, elle est contenue dans le dossier L76B et provient de ici.

La bibliothèque pour utiliser la carte micro SD est composée d'un seul fichier appelé sdcard.py, que l'on peut trouver ici.

Pour le buzzer, nous utilisons une bibliothèque fournie par le projet Sparrow de la Section Espace de Supaéro, le code peut être trouvé en Annexe A.

Enfin, le PMS5003 utilise le fichier pms5003.py que l'on peut trouver ici. Nous avons modifié la fonction repr dans le fichier original, afin de pouvoir obtenir les données dans un format simple pour les sauvegarder sur la carte micro SD. Ce changement est reporté en Annexe B.

III.2 Processus de développement

Nous avons adopté le même processus de développement que celui utilisé pour l'électronique, car le développement des programmes va de pair avec celui des composants électroniques. Nous avons testé chaque composant séparément avec un programme dédié, puis nous avons réalisé un test global avec tous les composants, pour enfin élaborer le programme final.

III.3 Programmes principaux

Les programmes principaux commencent par l'importation des fonctions nécessaires depuis les bibliothèques et par la configuration des broches pour les différents composants. Nous utilisons ensuite une boucle 'while True' pour récupérer les données du récepteur GPS (et du PMS5003 pour AéroSat), les sauvegarder sur la carte micro SD et les transmettre avec le module de télémétrie (uniquement pour AéroSat). Le code principal peut être trouvé en Annexe C.

Comme les données du détecteur de muons sont gérées par l'Arduino Nano, il n'est pas possible de les envoyer avec le module LoRaWAN conçu pour le RP Pico. C'est pourquoi nous ne transmettons pas les données du détecteur de muons, elles sont seulement sauvegardées sur la carte micro SD du détecteur de muons, tandis que les données GPS sont sauvegardées sur la carte micro SD de MuonSat. Aucune donnée n'est renvoyée au sol.

Quatrième partie

Conception mécanique

IV.1 Processus de développement

La manière la plus efficace et flexible pour concevoir la structure de la charge utile est d'utiliser les imprimantes 3D disponibles à l'ISAE-SUPAERO. Nous avons conçu la structure de manière itérative, en améliorant le design après chaque prototype pour aboutir au produit final.

Au départ, nous avions l'intention d'éjecter les deux Cansats depuis la fusée. Cependant, bien que le détecteur de muons soit suffisamment compact pour être stocké dans une structure de Cansat qui s'adapte au compartiment, la friction ne permet pas à ce Cansat de s'éjecter correctement. C'est pourquoi nous avons finalement décidé d'éjecter uniquement AéroSat. MuonSat n'est donc pas vraiment un Cansat, mais une expérience qui reste à bord de la fusée pendant tout le vol.

IV.2 Conception finale

La structure d'AéroSat est conçue pour s'adapter à son compartiment de Cansat et permettre un accès à l'interrupteur ON/OFF depuis l'extérieur du Cansat. Pour limiter la friction, les faces sont légèrement courbées de sorte que seule la partie centrale de chaque face touche la paroi. L'extrusion sur l'un des bords du couvercle permet à la trappe du Cansat de s'éjecter correctement, sinon le Cansat bloquerait la trappe lors de l'éjection. Le capteur de particules est placé au fond du Cansat et est maintenu en place par la structure. Le parachute est attaché au couvercle du Cansat, et les lignes du parachute sont fixées par paires dans quatre trous du couvercle.

Quant à MuonSat, sa structure est plus simple, elle consiste uniquement à maintenir l'électronique en place dans le compartiment. La structure est ouverte du côté du détecteur de muons, qui est logé au fond du compartiment. Nous avons percé un trou sur un côté de la structure de MuonSat pour faire passer une vis depuis l'extérieur du compartiment afin de maintenir MuonSat en place.

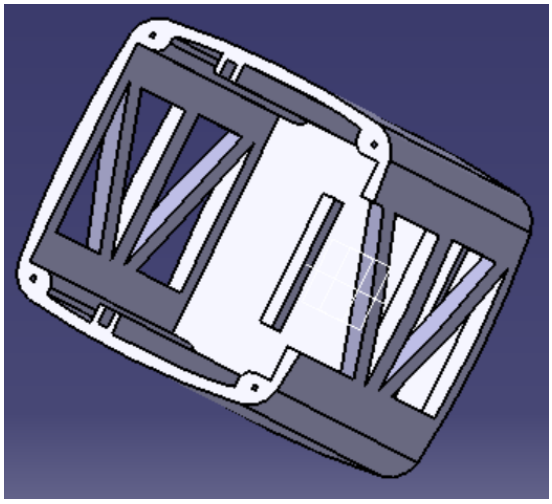


FIGURE 32 – Vue du dessus droit du corps d'AéroSat

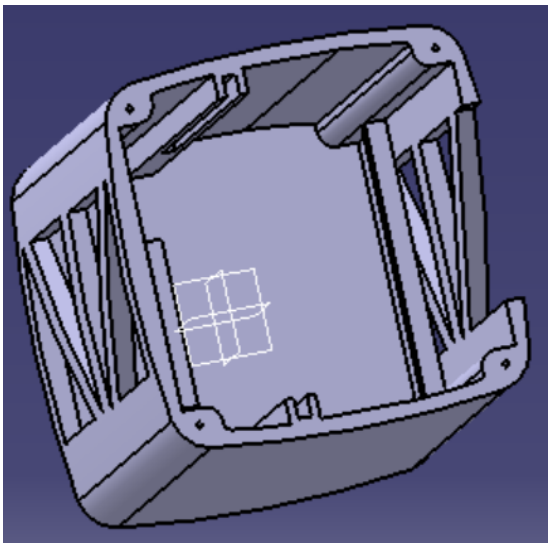


FIGURE 33 – Vue du dessus gauche du corps d'AéroSat

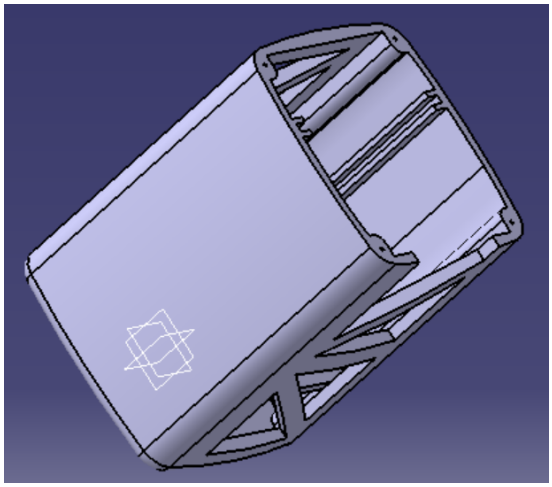


FIGURE 34 – Vue de côté du corps d'AéroSat

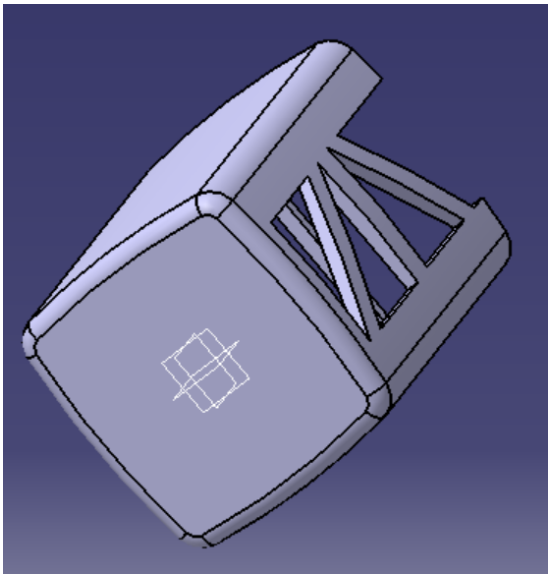


FIGURE 35 – Vue du dessous du corps d'AéroSat

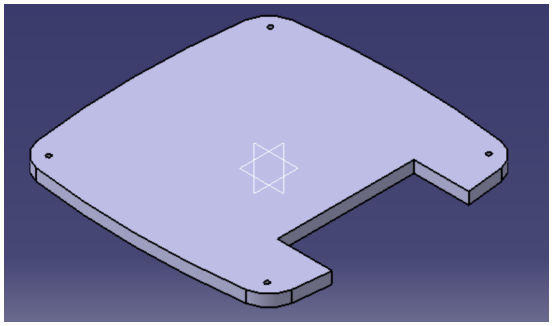


FIGURE 36 – Couvercle d'AéroSat

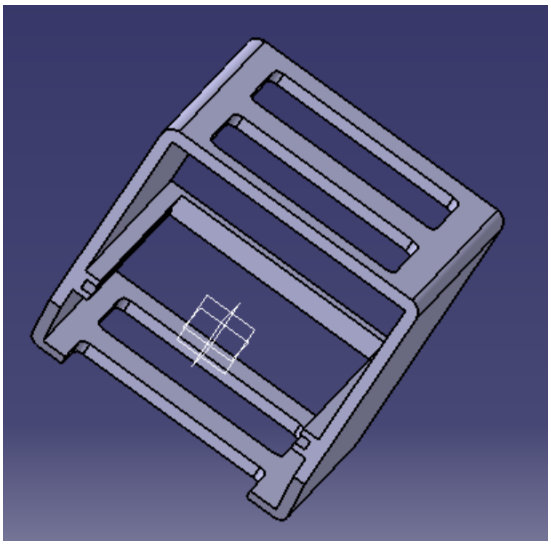


FIGURE 37 – Vue intérieure de MuonSat

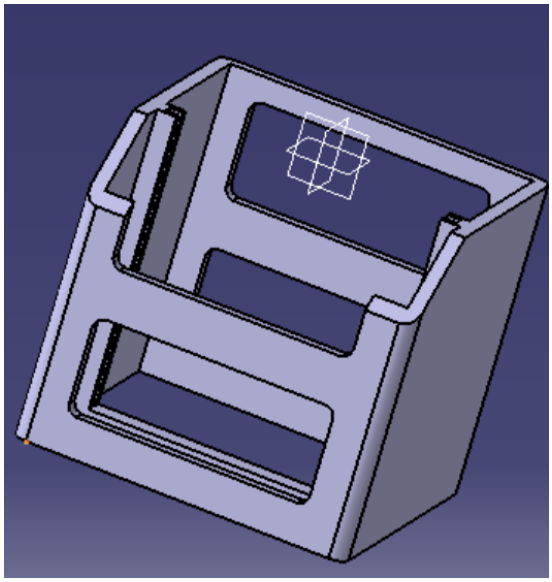


FIGURE 38 – Vue de côté de MuonSat

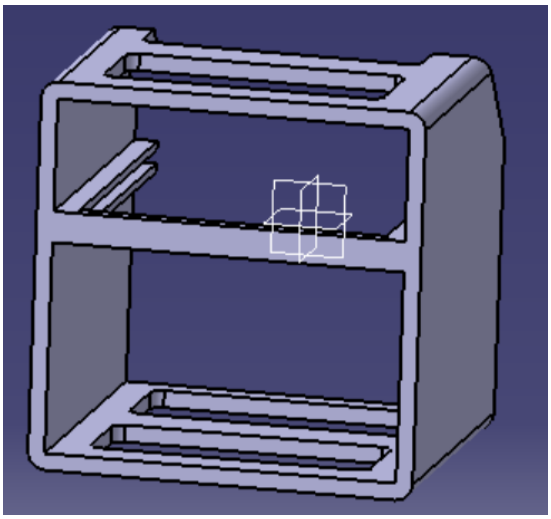


FIGURE 39 – Vue extérieure de MuonSat

Cinquième partie

Conception du parachute

V.1 Dimensionnement

La surface du parachute S_{para} est calculée avec la formule $V_{descente} = \sqrt{\frac{mg}{0.5\rho S_{para} C_{x_{para}}}}$.

$m = 0.300$ kg est la masse d'AéroSat (qui est la seule expérience à être éjectée de la fusée), $g = 9.81$ m.s^{-2} est la constante gravitationnelle, $C_{x_{para}} = 1$ est le coefficient de portance du parachute et $\rho = 1.225$ kg.m^{-3} est la densité de l'air.

Nous optons pour une vitesse de descente $V_{descente} = 7$ m.s^{-1} pour répondre aux exigences du CNES. Ainsi, $S_{para} = 0.098$ m^2. Cela correspond à un parachute en croix, chaque carré ayant un côté de 14 cm.

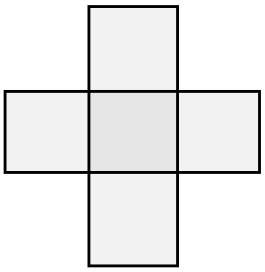


FIGURE 40 – Parachute

V.2 Fabrication

Pour fabriquer le parachute, nous avons découpé des morceaux de tissu rectangulaires de 16 cm de large et 54 cm de long dans une grande toile de parachute. Nous avons ensuite cousu les bords des panneaux de toile sur eux-mêmes pour les renforcer et obtenir des panneaux de 14 cm de large et 52 cm de long. Nous avons ensuite cousu les deux côtés ensemble pour former la croix. Enfin, nous avons ajouté des renforts aux coins pour faire passer les cordes et les attacher. Afin d'éviter que les cordes du parachute ne s'emmêlent pendant la descente, nous les avons regroupées à deux endroits différents sur les cordes avec des morceaux d'adhésif.

V.3 Essai

Pour tester les parachutes, nous avons simplement lesté la structure avec des masses factices pour simuler la masse réelle à l'intérieur de la structure et avons jeté ce modèle structurel avec le parachute depuis le quatrième étage de l'un des résidences étudiantes de l'ISAE-SUPAERO. Bien que la hauteur n'ait pas été suffisante pour observer un déploiement complet du parachute, les essais ont été suffisamment concluants pour garantir que le parachute se déploierait correctement.

Annexe

A Librairie du buzzer

```
# Berryrocket project
# Buzzer management
# Licence CC-BY-NC-SA
# Louis Barbier

import time
from machine import Pin,PWM,Timer

# Declaration d'un PWM pour le buzzer
buzzer = PWM(Pin(21))

# Declaration timer pour le buzzer et son extinction
timerBuzzer = Timer()
timerOffBuzzer = Timer()

# Variables pour le buzzer
freqBuzzer  = 500
tempsBuzzer = 0

# Management buzzer
def MgtBuzzer(timer):
    buzzer.freq(freqBuzzer)
    buzzer.duty_u16(32768) # Set to 50%
    timerOffBuzzer.init(freq=1.0/0.1, mode=Timer.ONE_SHOT,

    callback=SetOffBuzzer
    # Ring the buzzer for this time (0.1s)

def SetOffBuzzer(timer):
    buzzer.duty_u16(0) # Set to 0%

# Mets en marche le buzzer pour la frequence et la periode
indiquees
def SetBuzzer(enable=True, freq=500, tps=5.0):
    """Set the buzzer with a frequency and a time between
    ring"""

def SetBuzzer(enable=True, freq=500, tps=5.0):
    """Set the buzzer with a frequency and a time between
    ring"""
    global freqBuzzer
    freqBuzzer = freq
    timerBuzzer.deinit()
    if enable is True:
        timerBuzzer.init(freq=1.0/tps, mode=Timer.PERIODIC,

```

```
47          callback=MgtBuzzer)
```

B Fonction repr modifiée

```
def __repr__(self):

        csv_data = ','.join(map(str, self.data[:-2])) + '\n'
        return csv_data
```

C Programmes main

C.1 Program main d'AéroSat

```

"""
# SCALAR Va Charge Utile
# Programme principal d'AeroSat

"""


# Importations pour la carte SD
import os
from machine import Pin, SPI
import sdcard

# Importations pour le GPS
import time
from L76 import l76x
import math
import hashlib
from L76.micropyGPS.micropyGPS import MicropyGPS

# Importation pour le LoRa Wan
from sx1262 import SX1262

# Importations pour le PMS5003
from pms5003 import PMS5003

# Importations pour le buzzer
from buzzer import *

#Definition de la variable du buzzer
BUZZER_ENABLE = True

# Buzzer de debut d'initialisation
SetBuzzer(BUZZER_ENABLE, freq=800, tps=1)
time.sleep(1)
SetBuzzer(False)
```

```
37
38
39
40
41
42 """boucle"""
43
44 if __name__ == '__main__':
45
46
47
48     # Configuration de la carte SD
49     SD_CS_PIN = 5
50     SD_SCK_PIN = 6
51     SD_MOSI_PIN = 7
52     SD_MISO_PIN = 4
53     spi = SPI(0, baudrate=100000, sck=Pin(SD_SCK_PIN),
54
55     mosi=Pin(SD_MOSI_PIN), miso=Pin(SD_MISO_PIN))
56
57     sd = sdcard.SDCard(spi, Pin(SD_CS_PIN))
58     vfs = os.VfsFat(sd)
59     os.mount(vfs, '/sd')
60
61
62     # Configuration du GPS Locosys LS20031, le code utilise
63     est celui du GPS L76
64     UARTx = 0
65     BAUDRATE = 57600
66     #Il s'agit du baudrate adapte au module Locosys,
67     #pour le module L76, prendre BAUDRATE = 9600
68     gnss_l76b=l76x.L76X(uartx=UARTx,_baudrate = BAUDRATE)
69     gnss_l76b.l76x_exit_backup_mode()
70     gnss_l76b.l76x_send_command(gnss_l76b.SET_SYNC_PPS_NMEA_ON)
71     parser = MicropyGPS(location_formatting='dd')
72     sentence = ''
73
74
75     # Configuration du LoRa Wan
76
77     # La frequence d'emission du module est de 434MHz
78     # (norme EU433), avec un pas de 125kHz, un facteur
79     # d'etalement de 9, un taux de codage de 4/5, un mot
80     # de synchronisation de 0x12, une puissance de 10dBm,
81     # une limite de courant de 60mA, une longueur de
82     # preambule de 8 octets, une longueur d'implicite de
83     # 0xFF, un CRC active, une modulation IQ desactivee,
84     # un TCXO de 1.6V, un regulateur LDO desactive, et
```

```
    # un mode bloquant.

    # La puissance est limitee a 10dBm pour respecter le
    # cahier des charges du CNES.

    sx = SX1262(spi_bus=1, clk=10, mosi=11, miso=12, cs=3,
    irq=20, rst=15, gpio=2)

    sx.begin(freq=434.0, bw=125.0, sf=9, cr=7, syncWord=0x12,
            power=10, currentLimit=60.0, preambleLength=8,
            implicit=False, implicitLen=0xFF,
            crcOn=True, txIq=False, rxIq=False,
            tcxoVoltage=1.6, useRegulatorLDO=False,
            blocking=True)


    # Configuration du PMS5003
    pms5003 = PMS5003(uart=machine.UART(1, tx=machine.Pin(8),

    rx=machine.Pin(9),
    baudrate=9600),pin_enable=machine.Pin(19),pin_reset=machine.Pin(18)
    mode="active")


    # Les donnees du GPS et du PMS sont enregistrees sur la carte SD,
    #dans le fichier data.csv.
    file = open('/sd/data.csv', 'a')
    file.write("execution_time, WGS84 _latitude_parser,
    WGS84_latitude,
    WGS84_longitude_parser, WGS84_longitude, UTC_timestamp,
    Altitude, PM1.0,
    PM2.5, PM10, PM1.0, PM2.5, PM10,
    >0.3um in 0.1L air, >0.5um in 0.1L air, >1.0um in 0.1L air,
    >2.5um in 0.1L air, >5.0um in 0.1L air, >10um in 0.1L
    air\n")
    # En-tete du fichier CSV

    # Buzzer de fin d'initialisation
    SetBuzzer(BUZZER_ENABLE, freq=800, tps=0.2)
    time.sleep(0.6)

    # Initialisation du temps
    start_time = time.ticks_ms()

    # Boucle principale, qui recupere les donnees GPS et
    # PMS5003, et les envoie via LoRa Wan
    while True:

```

```
        # Buzzer de vol
        SetBuzzer(BUZZER_ENABLE, freq=1500, tps=1)

        # On recupere le temps d'execution
        end_time = time.ticks_ms()
        execution_time = time.ticks_diff(end_time,
        start_time)/1000

        # Recuperation des donnees GPS et du PMS5003
        if gnss_l76b.uart_any():
            sentence =
            parser.update(chr(gnss_l76b.uart_receive_byte()
            [0]))

            if sentence:

                PMS_data = str(pms5003.read())

                # Envoi des donnees
                sx.send(b"WGS84
                Coordinate:Latitude(%c),Longitude(%c)
                %.9f,%.9f"%

                (parser.latitude[1],parser.longitude[1],parser.
                latitude[0],
                parser.longitude[0]))
                sx.send(b"UTC Timestamp:%d:%d:%d"%

                (parser.timestamp[0],parser.timestamp[1],parser
                .timestamp[2]))
                sx.send(b"PMS data: %s"%PMS_data)

                # Ecriture des donnees sur la carte SD
                file.write(str(execution_time)
                + "," + str(parser.latitude[1]) + ","

                + str(parser.latitude[0]) + "," +
                str(parser.longitude[1])
                + "," +

                str(parser.longitude[0]) + "," +
                str(parser.timestamp[0])

                + ":" +

                str(parser.timestamp[1]) + ":" +
                str(parser.timestamp[2])
                + "," +
```

```
                str(parser.altitude) + "," + PMS_data + "\n")

                file.flush()


    # On ferme le fichier et on demonte la carte SD
    os.umount('/sd')
```

C.2 Programme main de MuonSat

```
"""
# SCALAR Va Charge Utile
# Programme principal de MuonSat

"""


# Importations pour la carte SD
import os
from machine import Pin, SPI
import sdcard

# Importations pour le GPS
import time
from L76 import l76x
import math
import hashlib
from L76.micropyGPS.micropyGPS import MicropyGPS

# Importations pour le buzzer
from buzzer import *

#Definition de la variable du buzzer
BUZZER_ENABLE = True

# Buzzer de debut d'initialisation
SetBuzzer(BUZZER_ENABLE, freq=800, tps=1)
time.sleep(1)
SetBuzzer(False)


"""boucle"""

if __name__ == '__main__':

```

```
37
38
39      # Configuration de la carte SD
40      SD_CS_PIN = 5
41      SD_SCK_PIN = 6
42      SD_MOSI_PIN = 7
43      SD_MISO_PIN = 4
44      spi = SPI(0, baudrate=100000, sck=Pin(SD_SCK_PIN),
45      mosi=Pin(SD_MOSI_PIN), miso=Pin(SD_MISO_PIN))
46
47      sd = sdcard.SDCard(spi, Pin(SD_CS_PIN))
48      vfs = os.VfsFat(sd)
49      os.mount(vfs, '/sd')
50
51
52      # Configuration du GPS Locosys LS20031,
53      #le code utilise est celui du GPS L76
54      UARTx = 0
55      BAUDRATE = 57600
56      #Il s'agit du baudrate adapte au module Locosys,
57      #pour le module L76, prendre BAUDRATE = 9600
58      gnss_l76b=l76x.L76X(uartx=UARTx,_baudrate = BAUDRATE)
59      gnss_l76b.l76x_exit_backup_mode()
60      gnss_l76b.l76x_send_command(gnss_l76b.SET_SYNC_PPS_NMEA_ON)
61      parser = MicropyGPS(location_formatting='dd')
62      sentence = ''
63
64      # Les donnees du GPS et du PMS sont enregistrees
65      # sur la carte SD,
66      #dans le fichier data.csv.
67      file = open('/sd/data.csv', 'a')
68      file.write("execution_time, WGS84 _latitude_parser,
69      WGS84_latitude,
70      WGS84_longitude_parser, WGS84_longitude, UTC_timestamp,
71      Altitude\n")
72
73
74      # Buzzer de fin d'initialisation
75      SetBuzzer(BUZZER_ENABLE, freq=800, tps=0.2)
76      time.sleep(0.6)
77
78      # Initialisation du temps
79      start_time = time.ticks_ms()
80
81      while True:
82
83          # Buzzer de vol
84          SetBuzzer(BUZZER_ENABLE, freq=1500, tps=1)
```

```
85
86
87          end_time = time.ticks_ms()
88          execution_time = time.ticks_diff(end_time,
89          start_time)/1000
90
91          # Recuperation des donnees GPS
92          if gnss_l76b.uart_any():
93              sentence =
94              parser.update(chr(gnss_l76b.uart_receive_byte()
95              [0]))
96
97              if sentence:
98
99                  # On ecrit les donnees sur la carte SD
100                 file.write(str(execution_time) + "," +
101                 str(parser.latitude[1])
102                 + ","
103
104                 + str(parser.latitude[0]) + "," +
105                 str(parser.longitude[1])
106                 + "," +
107
108                 str(parser.longitude[0]) + "," +
109                 str(parser.timestamp[0])
110                 + ":" +
111
112                 str(parser.timestamp[1]) + ":" +
113                 str(parser.timestamp[2])
114                 + "," +
115
116                 str(parser.altitude) + "\n")
117
118                 file.flush()
119
120
121
122     # Une fois qu'on a atterri, plus besoin d'enregistrer les
123     # donnees, on peut desactiver la carte SD.
124     os.umount('/sd')
```

D Photos

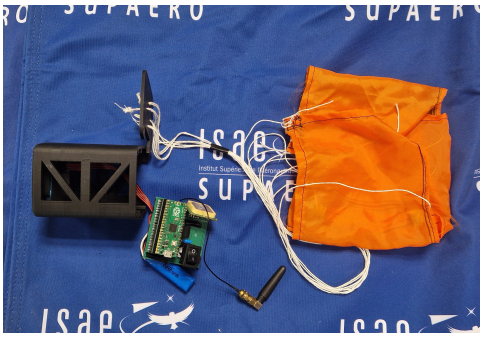


FIGURE 41 – AéroSat - 1

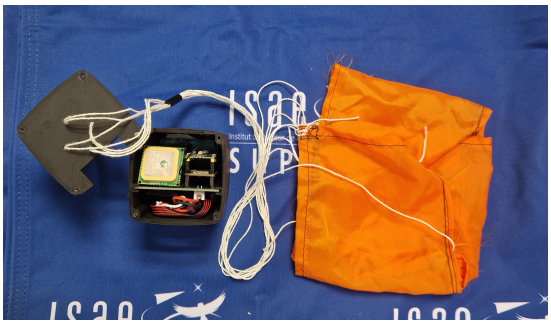


FIGURE 42 – AéroSat - 2

FIGURE 43 – AéroSat - 3

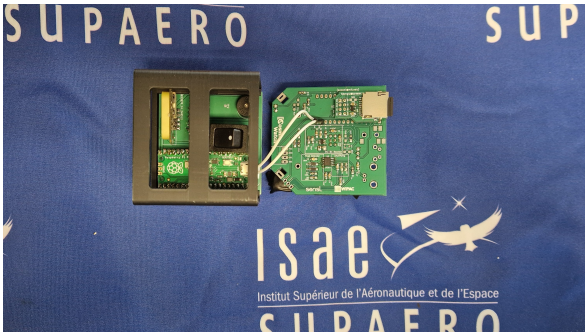


FIGURE 44 – MuonSat 1

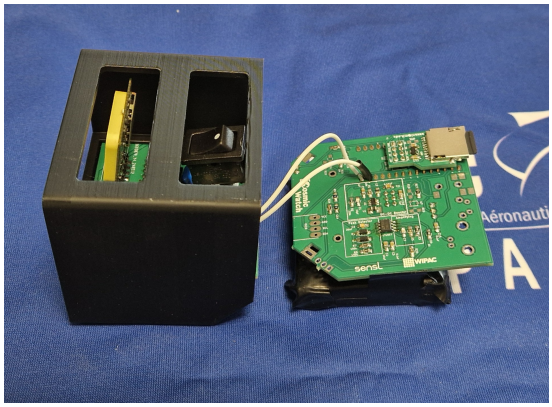


FIGURE 45 – MuonSat - 2

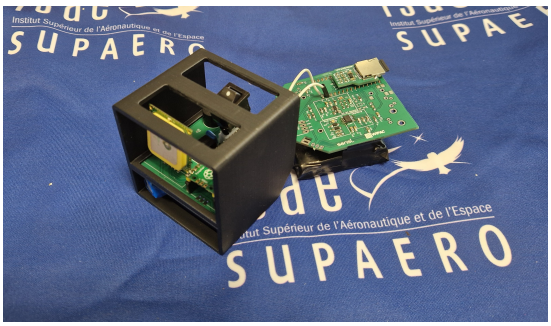


FIGURE 46 – MuonSat 3

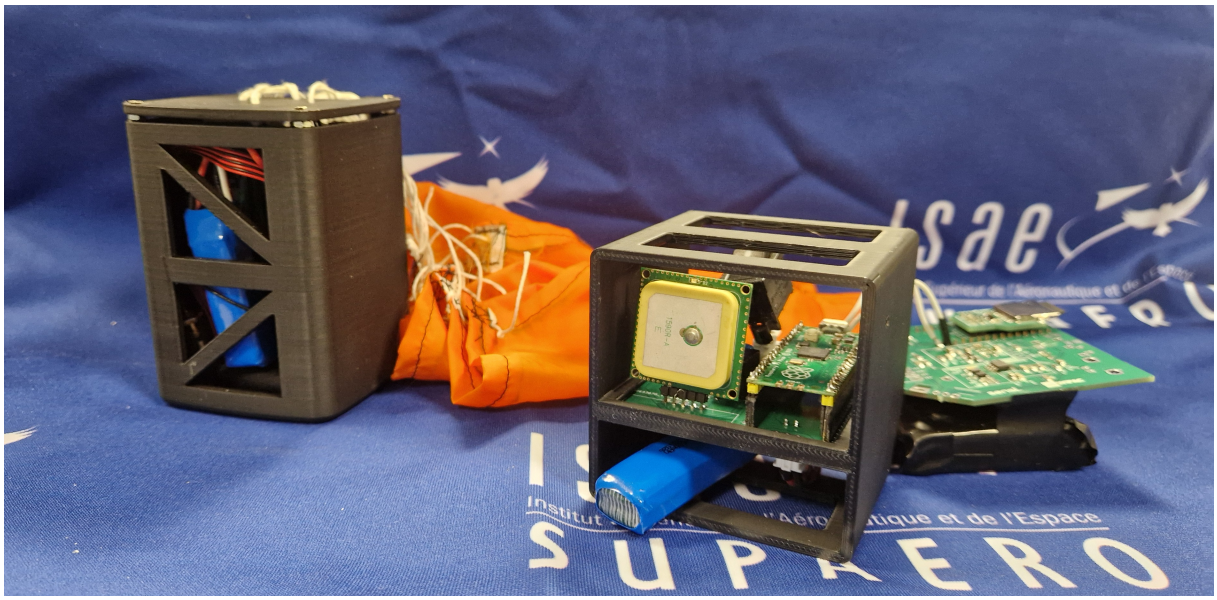


FIGURE 47 – SCALAR V-a full Payload