AERIS

Yo-Yo De-Spin Fusex

Rapport de projet 2016/2017

Remerciements

Nous remercions l'ESTACA, son Bureau des Elèves et l'Association des parents d'Elèves pour leur soutien financier. Remerciements particuliers à M. MANGIN pour son aide lors de l'impression de nos pièces en 3D mais aussi M.FAUX de l'atelier, qui nous a aidés dans la réalisation de nombreuses pièces mécaniques.

Merci à Planète Sciences et au CNES pour l'encadrement et le suivi de projet. Merci également au 1er RHP de nous avoir accueillis sur leur base et de nous avoir fourni la zone de lancement.

Merci à Serem74 et Gefi pour leur aide matérielle indispensable à l'aboutissement du projet.

Enfin, merci au club péruvien Cansat Peru, pour avoir prolongé le partenariat initié l'année dernière.

Sommaire

Table des matières

INTRODUCTION

Présentation de l'association ESTACA Space Odyssey

Association d'étudiants de l'école d'ingénieurs ESTACA., l'ESTACA Space Odyssey (ESO) a été créée en 1992. Simple association d'étudiants à ses débuts, l'ESO est depuis quelques années un club référent dans le domaine aérospatial en France. C'est de fait une des associations majeures de l'école et une de celles qui regroupe le plus d'étudiants.

Notre but est de promouvoir le domaine spatial au sein de notre école et auprès du grand public. Pour ce faire, nous concevons et réalisons chaque année différents types d'expériences liées à l'espace, notamment des fusées expérimentales, des mini-fusées, des ballons stratosphériques et des projets en microgravité. Ces projets sont réalisés pendant l'année scolaire au cours de campagnes de lancement comme le *C'Space*, organisé par le CNES ou au cours d'autres campagnes de lancement à l'international.

Depuis sa création l'association a lancé un peu plus de 90 projets, allant de la mini-fusée au ballon sonde en passant par les fusées expérimentales et bi-étage, ainsi que trois fusées supersoniques, trois fusées bi-étage (une inerte, deux propulsées), deux projets en microgravité et une multitude de fusées expérimentales et mini-fusées.

L'ESO est présente lors de deux événements majeurs du domaine aérospatial international. Présente à chaque Salon du Bourget tous les deux ans, au mois de juin, sur un stand ESTACA, elle était aussi présente au Congrès International d'Astronautique à Pékin en septembre 2013 et à Toronto en 2014.

Présentation de l'école d'ingénieurs ESTACA

L'ESTACA, Ecole supérieure des techniques aéronautiques et de construction automobile, forme des ingénieurs passionnés par les technologies, qui répondent aux besoins de nouvelles mobilités. À la pointe des technologies, elle offre un cadre privilégié pour une vie étudiante de qualité. Sa mission est de former des ingénieurs et de conduire une recherche appliquée au service de tous les acteurs des transports : aéronautique, automobile, spatial et transports urbains et ferroviaires. La formation répond aux nouveaux défis pour les transports de demain: respect de l'environnement, maîtrise énergétique, urbanisation croissante. Pour répondre à ces enjeux, l'ESTACA forme des ingénieurs multidisciplinaires, multiculturels qui sauront trouver des solutions technologiques innovantes pour répondre à la transformation profonde des modes de transport.

L'ESTACA, c'est :

- une formation ingénieur en 5 ans après le BAC,
- une école spécialisée en aéronautique, automobile, spatial et en transports ferroviaires,
- deux établissements : ESTACA-Paris et ESTACA Campus-Ouest à Laval en Mayenne,
- un projet de relocalisation du campus de Levallois vers Saint-Quentin-en-Yvelines en 2015,
- un enseignement personnalisé et de haut niveau qui répond aux besoins des entreprises,
- des équipes de recherche qui disposent d'équipements de haute technologie,
- une expérience internationale obligatoire,
- un diplôme habilité par la Commission des Titres d'Ingénieurs,
- une association de parents d'élèves associée à la vie de l'école.


A gauche : l'ESTACA campus Paris-Saclay ; à droite : l'ESTACA campus Ouest

Objectifs du projet

Initié en septembre 2016, le projet Aeris avait trois principaux objectifs :

Le premier était, sur le modèle des fusées sondes et des satellites, d'annuler le roulis de la fusée après la phase de propulsion, ceci afin de valider ou non l'efficacité de la technique dite de « yo-yo de-spin » dans l'atmosphère dense. Tout ceci validé à la fois par une centrale inertielle et par une caméra braqué sur la bague de-spin.

Le deuxième objectif était d'intégrer une télémesure accompagnée d'un GPS afin d'observer la trajectoire de la fusée du début du vol jusqu'à la retombée.

Enfin, nous avons souhaité renouveler la collaboration avec le club péruvien réalisant des Cansat (charge utile d'un volume approximatif d'un litre, embarquant diverses expériences et transportée par fusée : le Cansat peut éventuellement être larguée à l'apogée de cette dernière). Les péruviens viendraient alors au *C'space* édition 2017 avec leur cansat afin de l'embarquer dans la fusée expérimentale EOS et d'être largué à l'apogée.

Membres du projet

Nom	Rôle dans le projet	Année à l'ESTACA
Killian MELLE	Chef de projet	3ème année
Clément ROUSSEAU	Responsable pôle élec	5ème année
Augustin COUDRAY	Pôle élec	3ème année
Valentin CHAUVIN	Pôle élec	4ème année
Robin PIEBAC	Responsable pôle méca	3ème année
Pierre-Louis GAUTIER	Pôle méca	3ème année
Rémi CLAUDEL	Pôle méca	5ème année
Eric BARTH	Pôle méca	2ème année
Mathieu BARCLAY	Pôle méca	3ème année

EXPERIENCE PRINCIPALE : YO-YO DE-SPIN

Cette section a pour but de détailler le principe et le fonctionnement de l'expérience principale de la fusée expérimentale Aeris. Cette expérience, inédite au sein de l'ESO, a pour but d'annuler le roulis engendré qui fait suite au décollage de la fusex, par un système de libération de masses par déroulement.

Principe et fonctionnement du Yo-Yo De-Spin

- ***Principe***

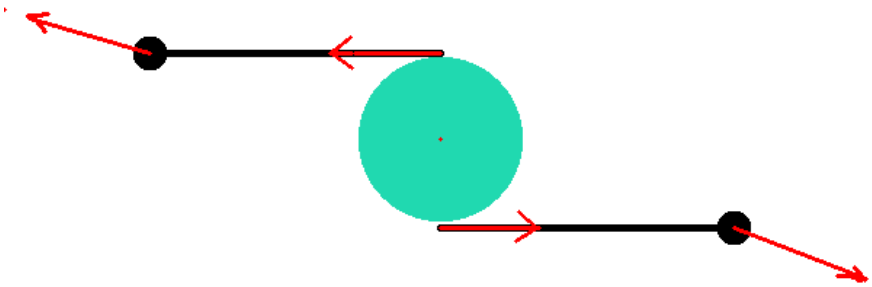
La méthode d'annulation de roulis par libération de masses est une méthode actuellement utilisée sur certains satellites spatiaux ou certaines fusées sondes, afin d'annuler leur vitesse angulaire, par une méthode simple qui répond au principe de conservation du moment cinétique.

C'est une méthode très intéressante pour le lancement de satellites dont l'étage les contenant a nécessité une stabilisation par spin forcé. Le satellite ne pouvant être libéré avec une rotation très élevée pour des raisons évidentes de contrôle de celui-ci, il est nécessaire d'annuler la rotation de l'étage avant de libérer le satellite.

Dans le cas des fusées sondes, celles-ci sont souvent stabilisées par un fort roulis (méthode simple et ne nécessitant pas de contrôle de stabilité électronique) durant leur phase de propulsion. Il est cependant nécessaire d'annuler ce roulis du mieux possible en fin de propulsion, et c'est à ce moment que la technique du de-spin entre en œuvre.

- ***Fonctionnement***

Le mécanisme se compose de deux longueurs de câble avec des poids sur les extrémités. Les câbles sont enroulés autour du dernier étage et / ou du satellite, à la manière d'un double yo-yo. Lorsque les poids sont libérés, le spin de la fusée les jette loin de l'axe de rotation. Ceci transfère suffisamment de moment angulaire aux poids pour réduire le spin du satellite à la valeur souhaitée.



Application à notre système

Cette technique est normalement utilisée hors-atmosphère (ou avec une faible présence d'atmosphère). En effet, l'absence de forces aérodynamiques et autres contraintes liées à celle-ci facilite grandement la libération des masses et le bon déroulement des câbles.

Notre expérience a pour but de tester l'impact des contraintes atmosphériques sur l'efficacité de ce système. Pour cela nous allons utiliser la technique de yo-yo de-spin de la même manière qu'un satellite en orbite.

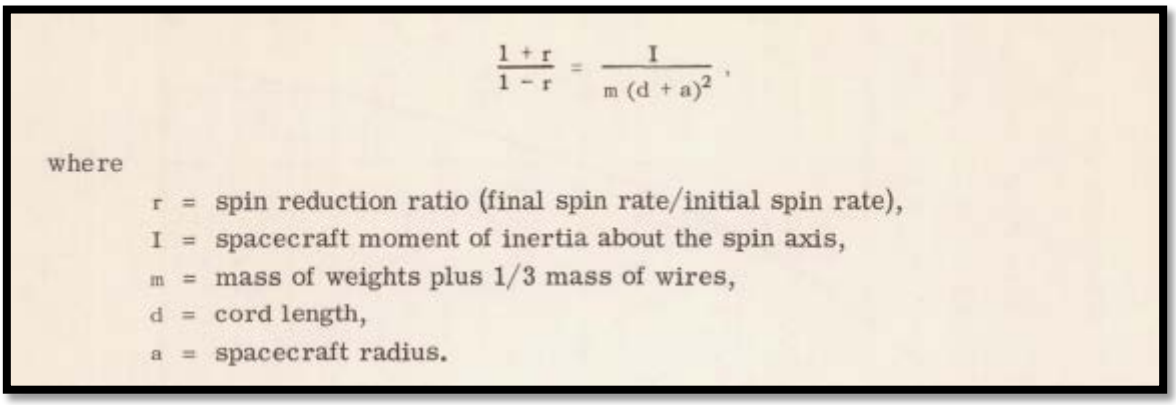
Pourquoi ?

La mise en roulis d'une fusée assure sa stabilité en vol, mais dès lors que la phase de propulsion est terminée, elle peut devenir gênante pour certaines expériences et/ou prises de données (largage de CanSat, vidéo embarquée etc.).

Dimensionnement du système

L'avantage de cette technique est que nos calculs ne dépendront pas de la vitesse angulaire de la fusée. Elle ne sera donc pas à calculer et n'interviendra pas dans nos calculs.

Pour dimensionner notre système, nous utiliserons les calculs mis en ligne par la NASA et initiés par Henry J. Cornille dans son dossier « *A method of accurately reducing the spin rate of a rotating spacecraft* », datant de 1962.



$$\frac{1+r}{1-r} = \frac{I}{m\,(d+a)^2}\,,$$

where

r = spin reduction ratio (final spin rate/initial spin rate),
I = spacecraft moment of inertia about the spin axis,
m = mass of weights plus 1/3 mass of wires,
d = cord length,
a = spacecraft radius.

Il est évident d'après l'équation que pour un moment d'inertie donné, le rapport de réduction de spin r est fixe lorsque la **masse m** et la **longueur d** du câble sont choisies. Le de-spin à zéro peut alors être effectué quelle que soit la vitesse de rotation initiale, à condition que le moment d'inertie ne varie avant libération des masses.

L'équation devient alors :

$$\frac{I}{m(d+a)^2} = 1$$

Pour déterminer la longueur du câble et les masses, nous avons mis en place un document Excel dans lequel est inscrit le moment d'inertie de la fusée (apparentée à un cylindre creux) mais également un nombre d'incréments de demi-périmètres. En effet la bague du yo-yo de-spin a un diamètre fixe et notre système de libération est également fixe (cf CAO). Ce système de maintien des masses et des câbles nous contraint dans la longueur de corde possible, par incréments de demi-périmètres.

Ainsi nous déterminerons les masses en fonctions des différentes longueurs de câbles possibles.

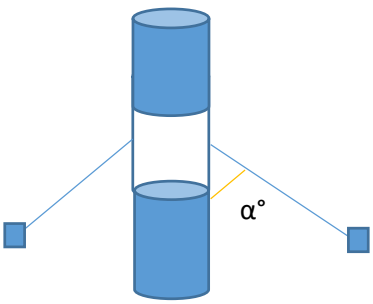
	Dimensionnement fil/masses De-Spin				
Nombre d'incréments de demi-périmètre	Longueur de Câble (m)	Masse Déduite (kg)			
1	0,2355	0,37758		Demi périmètre (m) =	0,157
2	0,3925	0,17522			
3	0,5495	0,10075		Diamètre Fusée (m) =	0,1
4	0,7065	0,06534		Inertie Fusée (kg/m²) =	0,0425
5	0,8635	0,04578			
6	1,0205	0,03385			
7	1,1775	0,02604			
8	1,3345	0,02065			
9	1,4915	0,01678			
10	1,6485	0,01390			
11	1,8055	0,01170			
12	1,9625	0,00999			
13	2,1195	0,00863			
14	2,2765	0,00753			
15	2,4335	0,00662			

Erreurs dues à la présence de forces aérodynamiques

Nous sommes conscients que la présence d'atmosphère ne facilite pas la tâche de cette expérience. Après avoir pris contact avec un professeur de l'université du Colorado, qui a lui-même travaillé sur le système de yo-yo de-spin, celui-ci nous a confirmé que les calculs de la NASA étaient trop simplistes dans le cas d'une utilisation telle que la nôtre. Il faudrait prendre en compte bien des phénomènes qui rendraient l'expérience trop compliquée. En effet, sur banc d'essai au sol (voir section 5), le système fonctionnera parfaitement car aucune vitesse verticale n'est présente et les forces qui résisteraient au déploiement des masses sont négligeables.

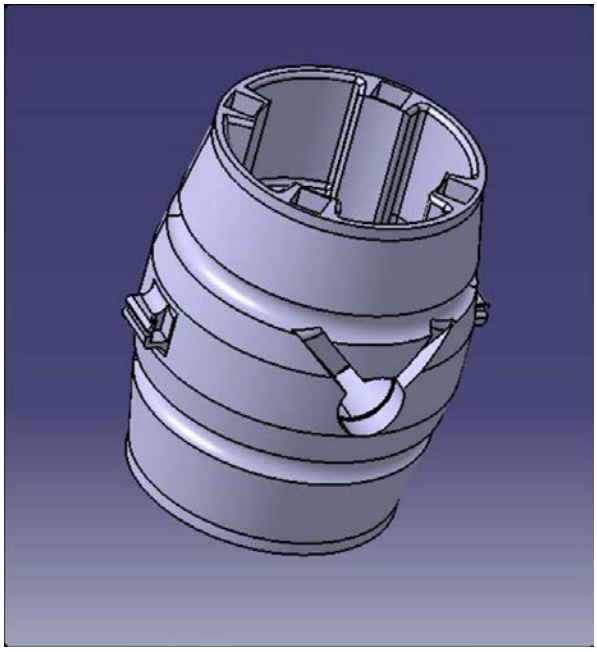
Alors que sur une fusée qui vient d'être propulsée et soumise aux contraintes de l'atmosphère, les masses et leurs fils peuvent se comporter autrement.

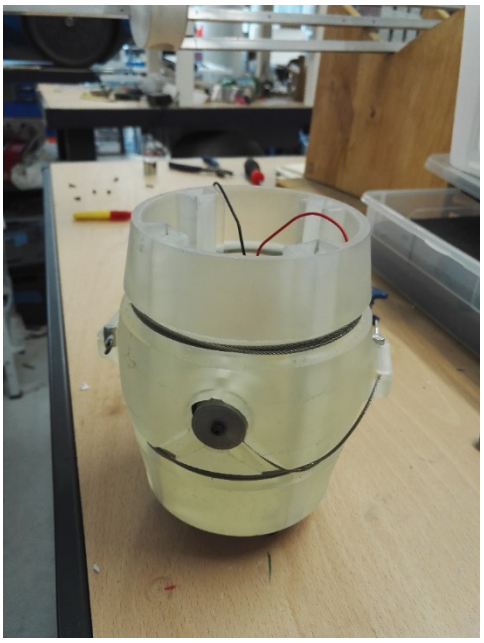
Pour parfaire le dimensionnement de notre système, il faudrait prendre en compte un phénomène de repliement des cordes vers le bas qui empêchent les masses d'être normales au plan de la bague de de-spin (voir illustration).



Il faudrait prendre en compte un angle α dans nos calculs et c'est justement ce phénomène que nous allons négliger pour observer son impact sur le bon déroulement des masses et le bon arrêt du roulis de la fusée.

CAO & rendu final de la bague De-Spin

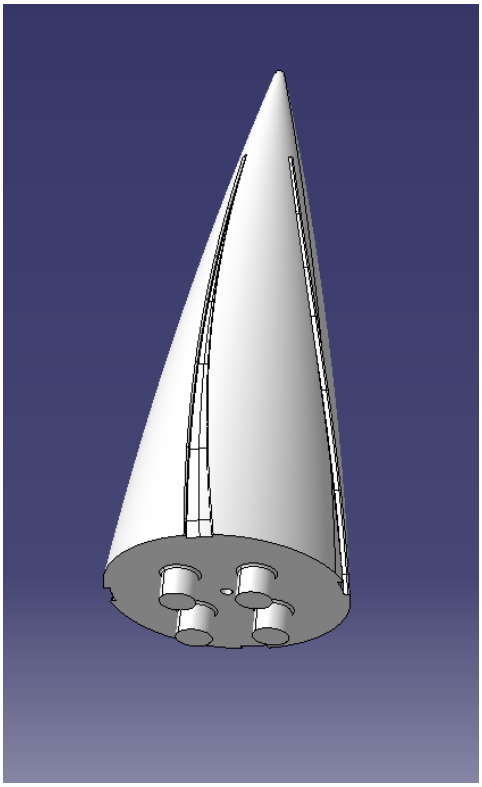


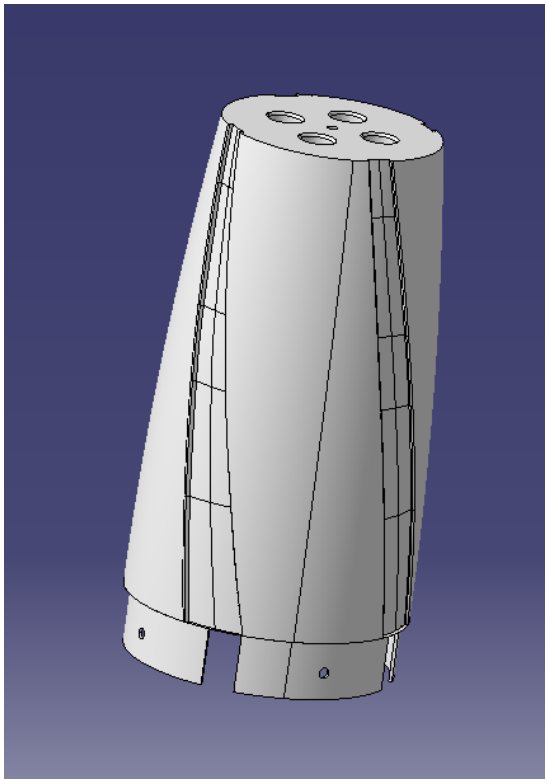


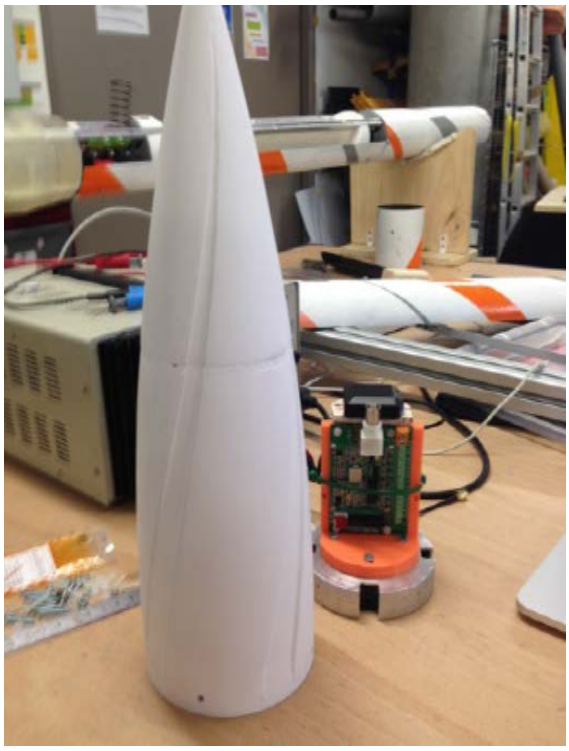
La bague De-Spin a été imprimée grâce à l'utilisation d'une imprimante à Stéréolithographie (aussi appelé « par extraction »). Ceci a permis d'obtenir une pièce d'une propreté inégalable (par rapport à une imprimante 3D classique à filament) et très solide (bien que sensible aux chocs) grâce aux propriétés de la résine utilisée.

Voici une vidéo en time lapse de l'impression de la bague.

CAO & rendu final de l'ogive







Nous avons choisi d'imprimer l'ogive et d'y ajouter un petit plus à savoir des rainures qui, nous le pensons, pourraient inciter la fusée à tourner dans un sens préférable à l'expérience. Cependant, aucun test en soufflerie n'a été effectué et le fonctionnement ou non de ces rainures était incertain et le demeure encore aujourd'hui malgré l'appui vidéo récupéré suite au lancement. Cette ogive fera partie des améliorations qui verront le jour sur Aeris II.

BANC D'ESSAI

Pour valider nos calculs vis-à-vis de l'efficacité de la bague De-Spin, nous avons conçu un banc d'essai tournant qui nous permet de simuler les conditions de vol.

Après de gros problèmes de stabilité et de vibrations, le banc d'essai a prouvé son efficacité et nous a permis d'être confiants quant à nos calculs. La seule incertitude demeurante fut l'inertie de celui-ci qui n'était pas forcément représentatif de celle de la fusée. Ce « problème » constitue également une voie d'amélioration qui sera abordée et étudiée sur Aeris II.

Voici une vidéo d'un des derniers tests qui fut effectué sur le banc d'essai.

DOSSIER DE SECURITE

Cette section a pour but de détailler les étapes de l'expérience du yo-yo de-spin ainsi que les critères assurant son bon déroulement. Le déroulement de fils autour de la fusée avant l'éjection du parachute étant une étape délicate, celle-ci nécessite une bonne préparation et une anticipation des risques occasionnels.

Déclenchement idéal

Le moment idéal pour libérer les masses serait le moment où la fusex possède une vitesse la plus faible possible. Ceci correspond à l'apogée mais il est exclu de libérer les masses à ce moment-là, pour des raisons qui seront expliquées ci-après. Nous nous baserons donc sur la chronologie qui suit :

- A t+0s le moteur est allumé, la fusée sort de rampe
- A t+14s le satellite péruvien est largué, vitesse = 31m/s
- A t+16s, le parachute est libéré, vitesse = 31m/s

L'étape primordiale à protéger de toute influence externe est bien sûr l'ouverture du parachute, qui est un critère de sécurité essentiel.

Il sera nécessaire de déclencher le système de yo-yo de-spin bien avant, tout en essayant d'avoir une vitesse faible, afin d'éviter le moindre risque d'emmêlement des câbles du yo-yo de-spin avec le corps de la fusée ou le parachute.

Sachant que le temps de déploiement des masses est **inférieur à 1 seconde** (fait qui sera confirmé grâce au banc d'essai), en prenant une marge de sécurité d'1 seconde supplémentaire (mauvais déploiement, déploiement tardif…), cela nous amènerait à libérer les masses à **t+11s** où la vitesse sera alors de 40m/s.

Ceci laisse **3 secondes** entre la libération des masses et l'éjection du parachute.

Les raisons d'un déploiement tardif/mauvais seraient :

- Des forces aérodynamiques créant un angle trop important entre la ligne de déploiement horizontal du de-spin et les masses (voir dossier d'expérience), qui feraient s'enrouler les câbles autour de la fusex.

- Un problème dans l'exécution du code de l'Arduino

- Un problème de minuterie analogique pour la case parachute (l'incertitude des composants de la minuterie sur son temps de réponse seront donc précisés et pris en compte)

- Un mauvais décollement des masses aux électro aimants (ex : une masse reste accrochée à son électro aimant et pas l'autre)

Incertitudes sur les composants électroniques

Yo-Yo De-Spin

Le déclenchement du de-spin et de l'ouverture de la trappe Cansat sont pilotés par une Arduino UNO, mais cette carte gère également l'acquisition et l'enregistrement des données de la centrale inertielle. Pour que le de-spin se déclenche en temps voulu, nous avons donc implémenté dans le code des interruptions qui nous permettent d'être sûrs de la durée de temporisation.

La détection de sortie de rampe (capteur à effet hall) génère une interruption qui déclenche la temporisation. Ensuite, à T+11s, une nouvelle interruption est générée (cette fois ci par le timer de l'Arduino) pour alimenter l'actionneur du de-spin, et de même pour l'éjection cansat à T+14s. Ainsi même en cas de boucle infinie ou de durées d'exécution anormales, on s'assure que la temporisation sera bonne.

Minuterie

Sur la minuterie analogique, les sources d'incertitudes de la temporisation se situent au niveau du circuit RC et de la résistance variable. En effet la température peut modifier les valeurs de résistance ou de capacité de ces composants ce qui entraîne un décalage de la minuterie.

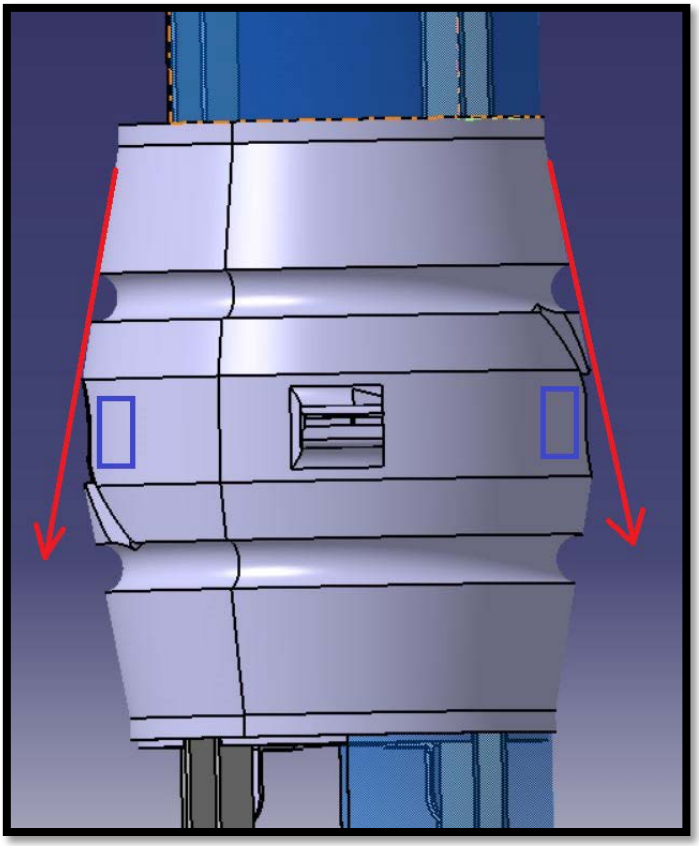
Si les résistances simples et les résistances variables sont peu affectées par cette effet (moins de 300 ppm/°C dans notre cas), le condensateur y est plus sensible. Sur notre minuterie nous possédons un condensateur présentant un décalage dû à la température de 2000°C/ppm. Etant donné que la minuterie est assez proche du propulseur, si l'on considère une température de 50°C dans la fusée contre 20°C au sol, la capacité du condensateur varie de : $2000\times(50-20)=6\%$. La temporisation sera alors de 16 ±1.08s.

Largage des masses

Nous avons discuté avec Planète Sciences du largage des masses (d'à peine 15 grammes chacune) sans système de récupération de celles-ci (qui serait très contraignant).

Les masses ne peuvent normalement pas se détacher au cours du vol sous l'effet de la poussée/des forces aérodynamiques, celles-ci étant maintenues par des électro aimants de 120N de tenu et soutenues par la bague de-spin qui limite leur exposition et les tient en place pour empêcher tout translation verticale.

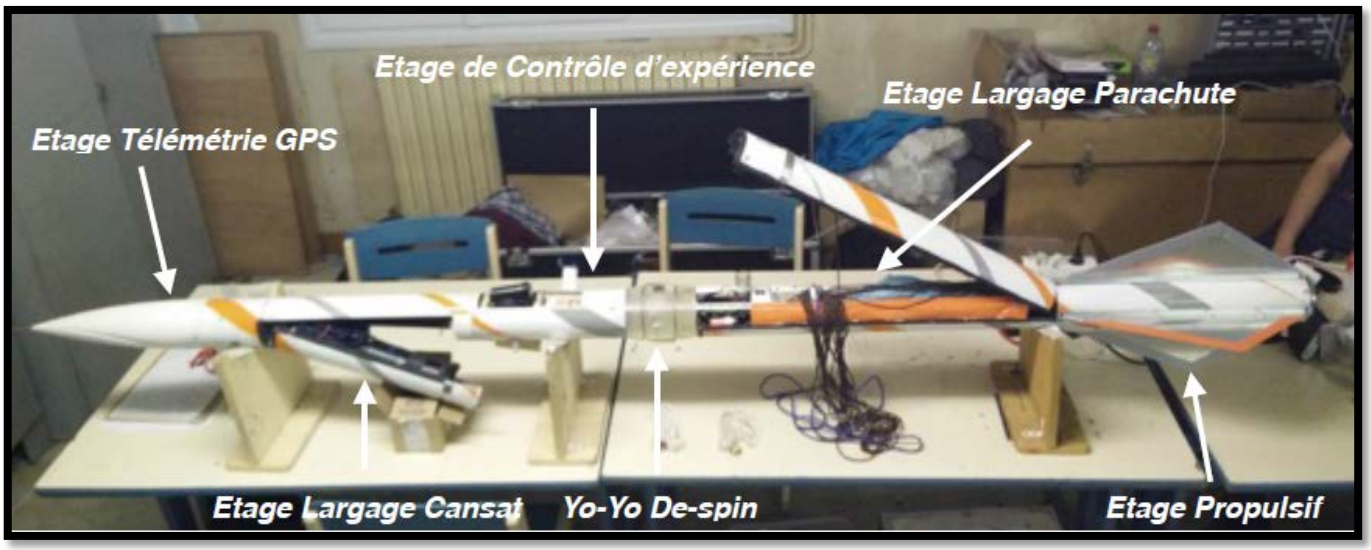
Le schéma ci-dessous représente assez bien l'écoulement de l'air (en rouge) et la déflection opérée par la bague, tout en gardant protégées les masses (rectangles bleus) :

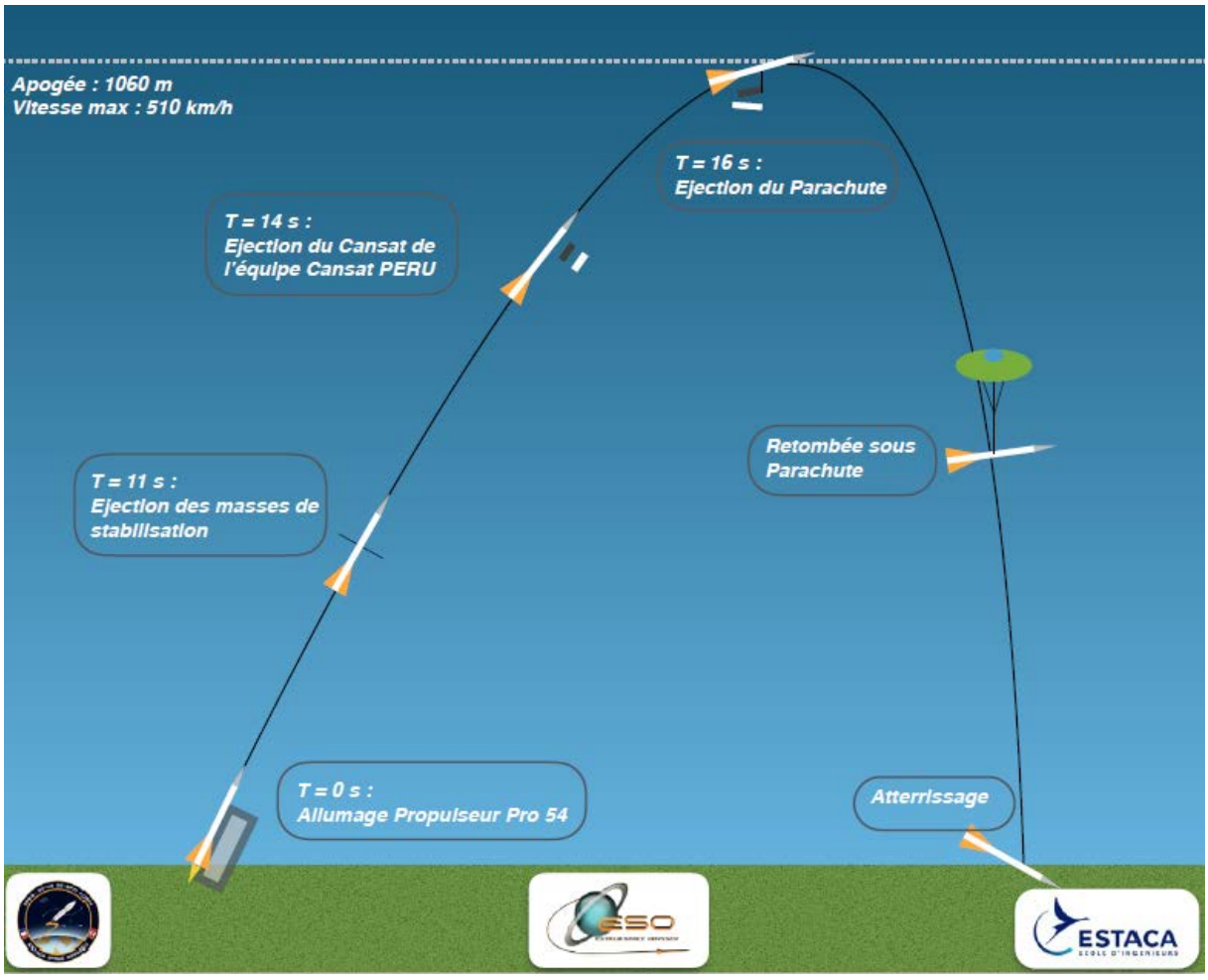


Les critères de sécurité vérifiables ont été testés via le banc d'essai. Pour les autres, seule une marge de sécurité et un bon respect de la chronologie de vol auraient assuré leur réussite.

ARCHITECTURE FINALE & PLAN DE VOL

Voici une photo explicative de la fusée finale, à la veille de son lancement.





MINUTERIE

Pour respecter le cahier des charges fusex, rédigé par le CNES et Planète Sciences, Aeris doit posséder un système de récupération. Le choix s'est porté sur un système classique de parachute avec ouverture d'une porte latérale, déclenchée par une minuterie analogique. Cette minuterie a été réalisée par des membres du projet, de la conception du schéma électrique sur logiciel jusqu'au réglage du temps de la minuterie, en passant par la réalisation du circuit imprimé. La partie conception du circuit et de la carte s'est faite sur ordinateur, en utilisant le logiciel Proteus 8 et ses outils ISIS (pour le schéma électrique) et ARES (pour obtenir le circuit à imprimer sur une plaque de cuivre).

Tous les composants de la minuterie sont reliés à la carte. En effet, pour faciliter l'accès à la carte, les piles sont sur des supports eux-mêmes fixés à la carte, les LED, sont également directement attachés à la carte. Enfin pour plus de simplicité, certains éléments, fixés ou non à la carte, y sont reliés par des connecteurs. Nous pouvons ainsi aisément retirer la carte de la fusée en cas de problème, pour régler le temps de la minuterie ou pour changer les piles par exemple.

Dans ce compte rendu, nous détaillerons toutes les étapes de réalisation de cette minuterie.

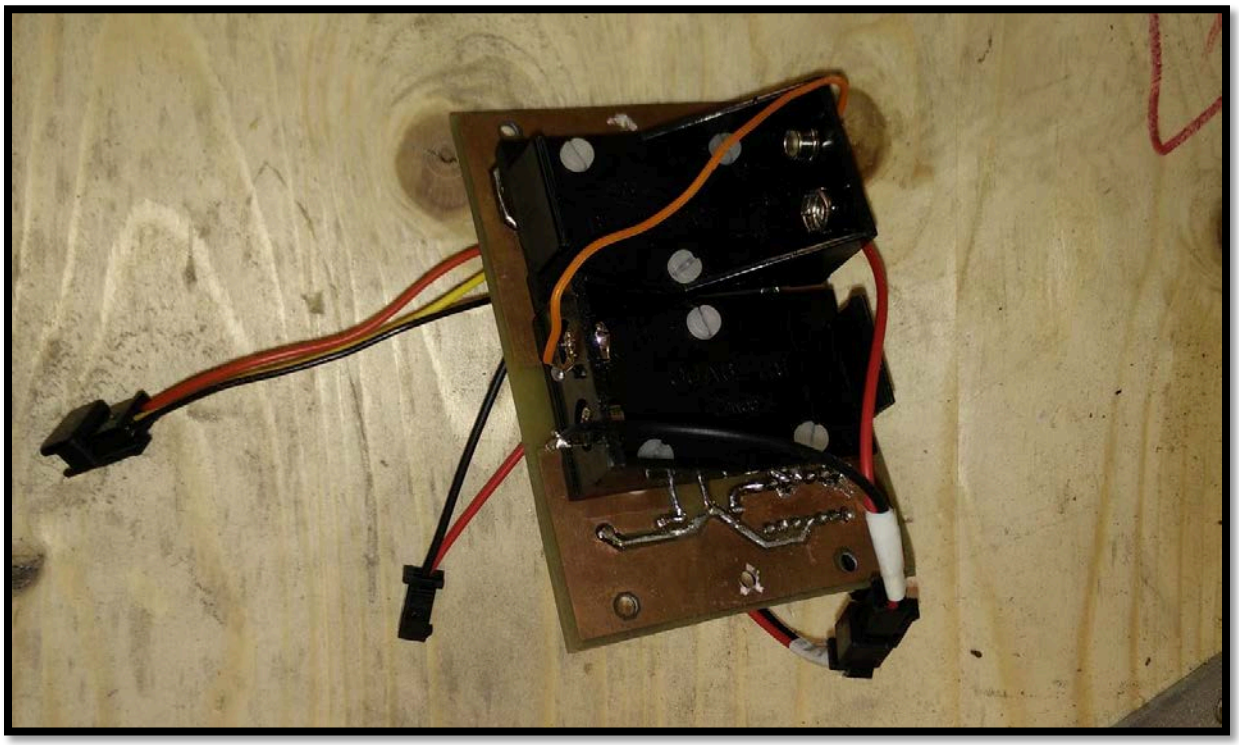


Figure 1: Minuterie avec ses piles fixées au dos de la carte

Définition du schéma électrique

La minuterie que nous utilisons déclenche l'ouverture de la porte du parachute par un système d'électro-aimant et de ressort. Cette méthode a déjà été utilisée au sein de l'association, nous avons donc pris la base des circuits utilisés dans de précédents projets. En utilisant ISIS, le circuit a été modifié afin de le rendre plus fiable et efficace. Par exemple, la minuterie sera alimentée par deux piles de 9V, soit une tension de 18V, or l'électro-aimant a un fonctionnement optimal lorsqu'il est alimenté en 12V. Nous avons donc placé un régulateur de tension qui permet de descendre la tension d'alimentation de 18 à 12V.

Pour la détection de sortie de rampe, la minuterie ainsi que l'expérience utilisent des capteurs à effet hall. La sortie de ce capteur est à 12V (sa tension d'alimentation) par défaut et passe à 0V lorsque l'on approche un aimant suffisamment près. En plaçant un aimant attaché à la rampe sur la fusée, on peut alors détecter la sortie de rampe en pilotant un transistor à l'aide de capteur qui relie le circuit à la masse et déclenche la minuterie.

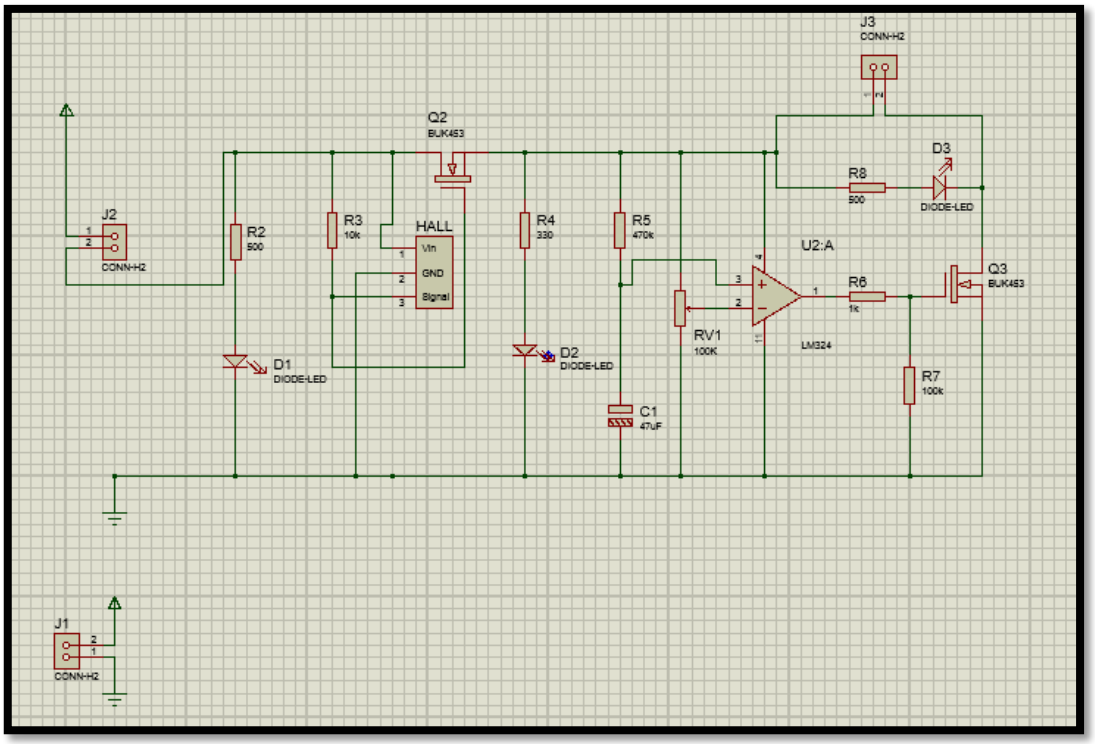


Figure 2: Schéma du circuit initial

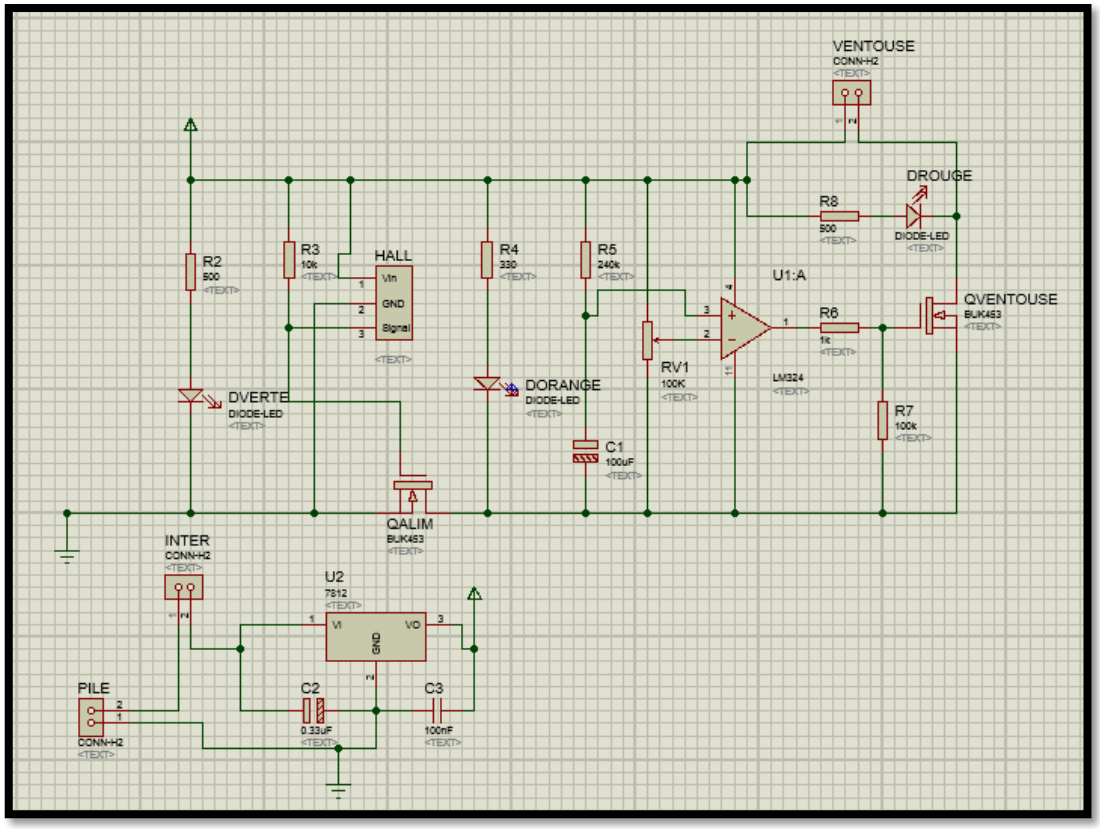


Figure 3: Schéma électrique final de la minuterie sous ISIS

Lorsque le circuit est défini, nous le réalisons sur une plaque d'essai pour tester son bon fonctionnement. Cela nous permet de repérer les problèmes et de les corriger avant de faire une plaque de cuivre. Nous avons ainsi changé l'AOP LM324N initial par un TL082, permettant un gain de place, mais les tests ont révélé un problème lié au TL082. En effet, l'AOP conserve de l'énergie lorsque l'alimentation est coupée, et déclenche le système d'ouverture dès que l'alimentation est à nouveau branchée, alors que le courant ne devrait pas atteindre la zone de l'électro-aimant. Nous sommes donc revenus à l'utilisation d'un LM324N.

Design de la carte avec ARES

Une fois le circuit définitif validé, les composants nécessaires déterminés, on se prépare à réaliser la carte. Nous utiliserons des plaques de cuivre que nous passons à l'insoleuse puis dans différents produits afin d'y faire apparaitre les pistes qui permettront de relier les composants entre eux. Pour obtenir ces pistes, on utilise l'outil ARES du logiciel Proteus 8. A partir du circuit défini sur ISIS, ARES créé un routage des pistes, il propose une disposition des composants pour permettre un routage efficace. Une fois l'opération automatique effectuée, nous pouvons si besoin modifier le résultat obtenu, pour baisser le nombre de pistes ou finir le routage (ARES n'étant pas toujours capable de trouver une solution pour relier tous les points). Nous ajoutons également des plans de masse de chaque côté de la carte. Voici le résultat obtenu pour la minuterie d'Aeris :

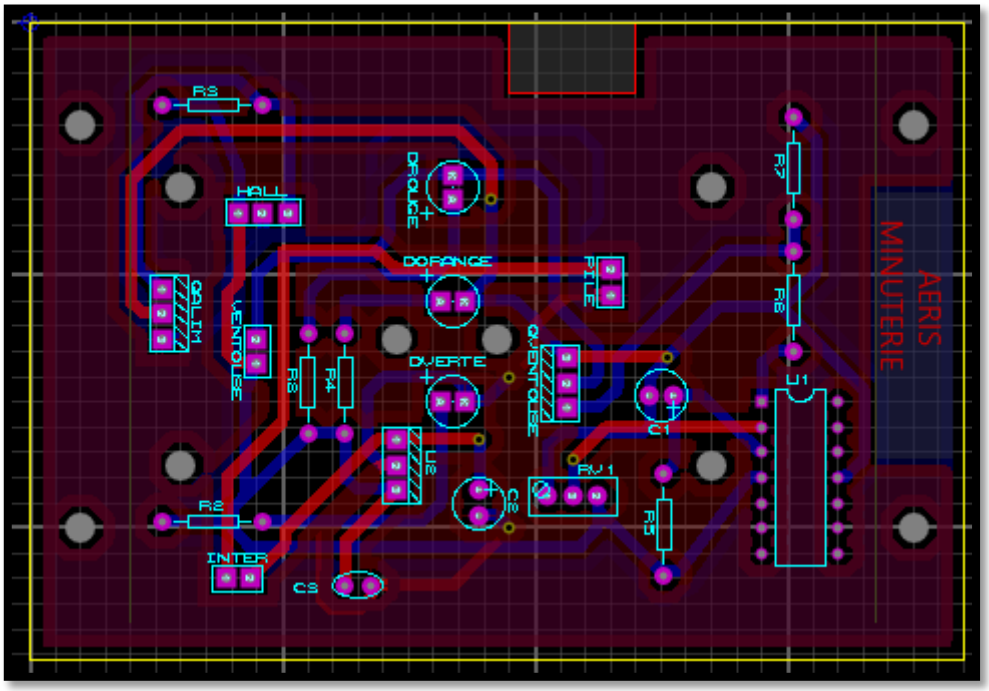


Figure 4: Routage sur ARES

EXPERIENCE

Aeris contient deux expériences : l'expérience principale étant le De-spin, et la seconde l'éjection du Cansat péruvien. Pour la première, nous avons besoin de lâcher deux masses avec des électro-aimants et une carte micro-SD Adafruit vient récupérer les informations sur la position et la rotation selon x, y et z données par une centrale inertielle Adafruit. La seconde ne demande que l'utilisation d'un électro-aimant afin de libérer la trappe pour le Cansat. Ces deux expériences sont contrôlées par une seule carte utilisant une Arduino UNO (dont le code sera exposé par la suite), celle-ci commande et alimente la centrale inertielle ainsi que le port micro-SD. Une minuterie est utilisée pour déclencher le De-spin et l'éjection du Cansat, un led rouge sur la carte s'allume lorsque le De-spin est déclenché et une led bleue pour l'éjection du Cansat. Le temps de déclenchement pour les minuteries sont données dans le rapport de sécurité et ceux exposés dans le code sont à titre indicatif ou pour des essais. La carte utilise au final 5 piles 9V, une pour l'Arduino, deux pour les électro-aimants du De-spin et deux pour l'électro-aimant du Cansat. De plus les électro-aimants nécessitants une tension de 12V, une carte de régulation a été faite pour ces derniers.

Explications du fonctionnement

L'Arduino est d'abord alimentée par une pile 9V. Le code de l'Arduino permet de, lorsque le capteur à effet hall switch (sortie de rampe), lancer un timer pour le nombre de secondes désirées (dans le code), dans le cas du code ci-présent, au bout de 11sec, la led rouge s'allume via un changement d'état de pin correspondante de l'Arduino, pendant 11sec déclenchant ainsi le relâchement des masses du yoyo, de même, une led bleue s'allume au bout de 14sec déclenchant l'éjection du Cansat et reste déclenché pendant 10sec. De plus, à partir du moment où l'Arduino est alimenté, la centrale inertielle est active et renvoie les informations à l'Arduino qui les enregistre en fichier .txt dans la carte microSD.

La carte de régulation est simplement composée de deux condensateurs, et un régulateur 7812 pour sortir du 12V.

Etalonnage de la centrale inertielle

Nous avons effectué l'étalonnage avec la centrale inertielle intégrée dans la fusée. Pour cela nous avons disposé la fusée dans différentes positions de telle sorte à exposer chaque axe de l'accéléromètre à -1, 0 et +1g pendant quelques secondes. Le tableau ci-dessous donne la moyenne des mesures sur cette période (valeurs avec coefficients correctifs) :

		Mesuré		
		Axe X	Axe Y	Axe Z
Théorique	-1	-0,97	-1,00	-1,03
	0	0,00	0,03	0,02
	1	1,02	0,98	1,01

Figure 5: Etalonnage des accéléromètres

Pour l'étalonnage des gyromètres, nous avons effectué des rotations de 90° sur chaque axe. En intégrant les mesures de vitesse de rotations on obtient alors :

Axe de mesure	Théorique	Mesuré	Erreur relative
X	90	92.44	2.7%
Y	-90	-89.47	0.6%
Z	90	87.16	-3.2%

Figure 6: Etalonnage des gyros

TELEMESURE

Aeris utilise la télémesure pour transmettre au sol des coordonnées GPS permettant d'estimer la trajectoire de la fusée (bien que la fréquence de mesure soit faible) mais surtout de pouvoir connaître sa dernière position avant l'atterrissage, facilitant ainsi sa localisation.

Le circuit utilisé pour réaliser cette fonction a été conçu pour être le plus simple possible. Il est composé d'un PIC qui traite les données provenant du module GPS, conserve les informations qui nous intéressent et envoie celles-ci à un modulateur. Le modulateur met en forme le signal qu'il envoie directement à l'émetteur, le KIWI.

Le module GPS

Le module que nous avons choisi provient d'Adafruit : https://www.adafruit.com/product/746. Ce GPS à une fréquence de sortie de 1 Hz en nominal (la fréquence que nous utilisons) et 10 Hz au maximum. Par défaut, le module produit quatre trames successives contenant différentes informations.

```
$GPGSA,A,3,14,25,31,29,32,,,,,,,,2.05,1.82,0.94*09
$GPRMC,194509.000,A,4042.6142,N,07400.4168,W,2.03,221.11,160412,,,A*77
$GPVTG,221.11,T,,M,2.03,N,3.77,K,A*3E
$GPGGA,194510.000,4042.6127,N,07400.4184,W,1,5,1.82,143.2,M,-34.2,M,,*6E
$GPGSA,A,3,14,25,31,29,32,,,,,,,,2.05,1.82,0.94*09
$GPRMC,194510.000,A,4042.6127,N,07400.4184,W,1.60,212.71,160412,,,A*7E
$GPVTG,212.71,T,,M,1.60,N,2.97,K,A*31
$GPGGA,194511.000,4042.6118,N,07400.4196,W,1,5,1.82,142.6,M,-34.2,M,,*65
$GPGSA,A,3,14,25,31,29,32,,,,,,,,2.05,1.82,0.94*09
$GPRMC,194511.000,A,4042.6118,N,07400.4196,W,1.24,213.26,160412,,,A*73
$GPVTG,213.26,T,,M,1.24,N,2.29,K,A*37
```

Figure 7: Trames GPS

Nous nous intéresserons ici à la trame GPGGA qui contient entre autres le temps GMT, la longitude, la latitude, la qualité du signal et l'altitude.

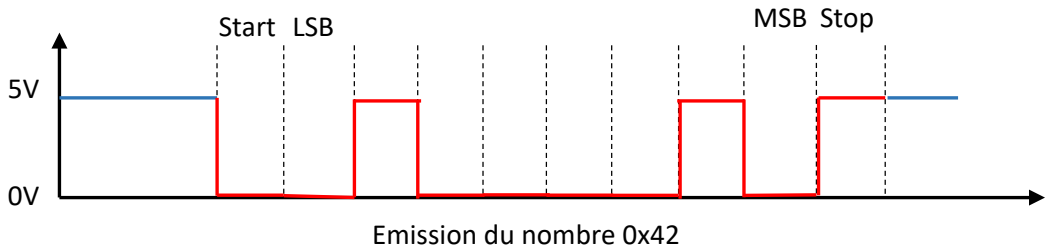
Le PIC

Le PIC que nous utilisons est un PIC 16F688, fabriqué par Microchip : http://www.microchip.com/wwwproducts/en/PIC16F688. Nous avons porté notre choix sur ce composant car c'est le plus petit PIC équipé d'un UART (Universal Asynchronous Receiver Transmitter), permettant d'émettre ou de recevoir des données par liaison série. C'est donc cette fonctionnalité que nous utilisons pour recevoir des données GPS.

A chaque détection de début de trame GPS (détecté par les caractères « $GPGGA »), le PIC stocke dans un tableau les valeurs qui nous intéressent (voir paragraphe précédent). Dès la fin de la trame GPS, le PIC envoie les données enregistrées dans ce tableau.

Pour l'envoi des données, il aurait simplement suffi d'utiliser l'UART présent sur le PIC, mais à des fins d'expérimentation, il a été décidé d'implanter un protocole (très simple) de communication série sur le logiciel. Pour cela, tout octet envoyé est précédé d'un bit de start et suivi d'un bit de stop. De plus, on insert à la fin de chaque trame les caractères « \r » et « \n », qui marquent le début d'une nouvelle ligne.

Cette trame est générée sur un port I/O classique et relié au circuit de modulation. Toute la difficulté de cette méthode est de régler correctement la durée de chaque bit émis pour respecter le débit de communication (4800 bits/s). Ce réglage est fait à l'aide d'un oscilloscope analyseur de trame.



Emission du nombre 0x42

L'exemple ci-dessus représente l'émission du nombre 0x42 (ou 01000010 en binaire). Lorsqu'aucun caractère n'est émis, la sortie est maintenue à 5V. L'émission d'un octet commence par un bit de start (niveau logique 0) puis l'octet bit à bit en commençant par le bit de poids faible puis pour finir un bit de stop (niveau logique 1).

La carte comporte un connecteur SUB-D à neuf broches (voir annexe 2) qui permet de connecter la carte au GPS et éventuellement au programmateur du PIC (PICKIT2). Il y a deux configurations possibles :

- Configuration de vol : la carte est reliée au GPS uniquement. SWITCH en position 2, l'alimentation se fait par la pile 9V et un régulateur de tension 5V.
- Configuration de programmation/test : la carte est reliée au GPS et au PICKIT2 (possibilité d'ajouter en plus un oscilloscope ou un analyseur de trame pour la sortie du PIC). SWITCH en position 1, l'alimentation se fait via le PICKIT.

Les pins de 1 à 5 sont dédiés au PICKIT. Elles ont pour fonctions :

- 1 -> VPP ou Master Clear, servant à la réinitialisation et la détection du type de PIC
- 2 -> VDD, l'alimentation
- 3 -> VSS, la masse du circuit
- 4 -> ICSPDAT
- 5 -> ICSPCLK utilisés pour la programmation

Les pins de 6 à 8 sont dédiés au GPS.

- 6 -> +5V, l'alimentation du module
- 7 -> GND la masse
- 8 -> TX la patte de transmission du GPS

La configuration par défaut du GPS étant satisfaisante dans notre cas, nous n'avons donc pas besoin d'utiliser le pin RX du GPS qui permet une transmission montante du PIC vers le GPS. La pin 9 peut alors être utilisée pour sonder la sortie du PIC (pour régler la durée d'un bit par exemple).

La modulation

Le circuit de modulation est celui proposé par Planètes Sciences dans le cadre de la formation télémesure. Ce circuit est construit autour du générateur de fonction XR2206. Le circuit est conçu de sorte à réaliser une modulation de fréquence de type FSK (Frequency Shift Keying). Le principe est de générer deux fréquences correspondant respectivement à un 0 ou 1 logique.

Suivant les standards de Planètes Sciences et dans le but de maximiser la bande passante tout en conservant une transmission efficace, nous avons choisi d'utiliser un débit de 4800 bauds. Les deux fréquences qui doivent être générées sont alors de 15 kHz pour un niveau logique 0 et 9 kHz pour un niveau logique 1, avec une amplitude crête à crête de 1.5V.

Les deux fréquences sont déterminées par le condensateur C4 et les résistances fixes RF1 et RF2 et les résistances variables RVF1 et RVF2, ces deux dernières permettant un réglage fin des fréquences (voir annexe 2), selon les formules :

$$F_1 = \frac{1}{(RF_1 + RVF_1) * C_4} \; et \; F_2 = \frac{1}{(RF_2 + RVF_2) * C_4}$$

La sortie de l'oscillateur est à la fréquence F1 lorsque l'entrée est supérieure à 2V (1 logique) et à la fréquence F2 pour une entrée inférieure à 1V (0 logique). L'amplitude est quant à elle ajustée à l'aide du potentiomètre RVAMP.

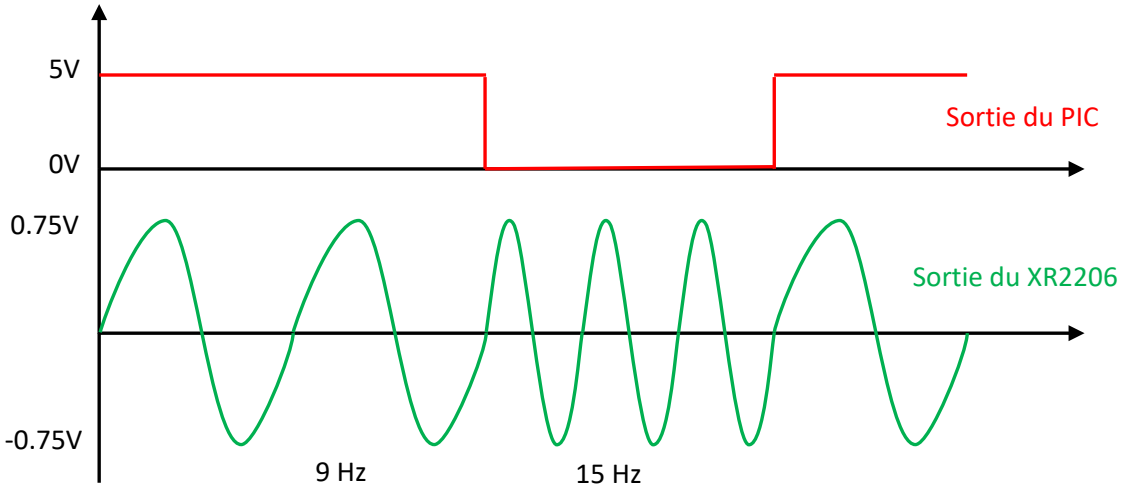


Figure 8: Principe de la modulation FSK

Réalisation des cartes

A partir du routage sur ARES, nous imprimons les pistes sur des feuilles transparentes. Pour fabriquer les cartes électroniques, nous utilisons, en plus de ces feuilles : une plaque de cuivre photosensible, une insoleuse, une bassine de révélateur et un bac de perchlorate de fer avec un radiateur.

On commence par se placer dans un endroit sombre pour passer la plaque à l'insoleuse : une fois les feuilles de routage (inférieure et supérieure) placée de chaque côté de la plaque, celle-ci est exposée à une lumière vive, le cuivre exposé réagis aux UV. La carte est ensuite plongée dans le révélateur, qui permet d'ôter la protection du cuivre exposée aux UV. Les pistes commencent à se dessiner. Enfin, nous plongeons la carte dans le bain de perchlorate de fer chauffé autour de 30°. Le perchlorate réagis alors avec le cuivre et le retire de la plaque, ne laissant plus que les pistes et plans de masse utiles au fonctionnement de la carte. Nous perçons les trous nécessaires à la mise en place des composants, et la fixation de la carte, il ne reste plus qu'à nettoyer la carte à l'acétone, enlevant ainsi la protection recouvrant le cuivre resté en place, nous pourrons ainsi souder les composants dessus.

La dernière étape consiste donc à souder l'ensemble des composants, définis sur le schéma électrique sur ISIS, en suivant le plan de routage réalisé sur ARES. Nous nous assurons ensuite du bon fonctionnement de la carte, et lorsque ce n'est pas le cas, nous cherchons les origines des problèmes pour les corriger et obtenir une carte fonctionnelle.

ANALYSE DES RESULTATS

Expérience

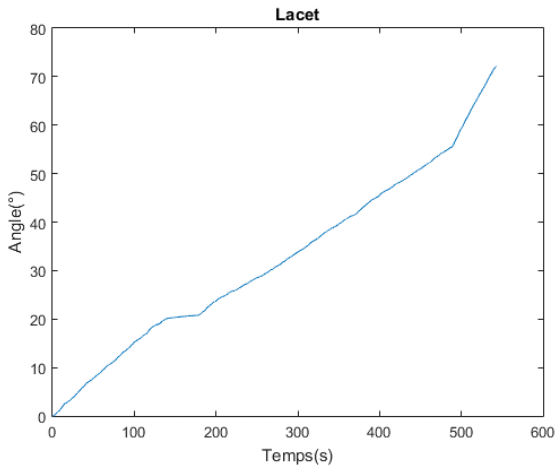
A notre grande déception, le yo-yo de-spin n'a pas fonctionné car la sortie de rampe n'a pas été détectée. Comme on peut le voir sur la vidéo, les deux LED sont restées allumées fixes ce qui signifie qu'aucun changement d'état du capteur à effet hall n'a été relevé par l'arduino.

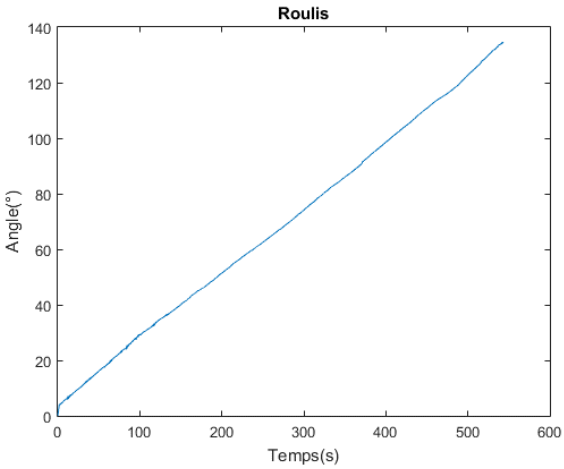
En effet, le code a été conçu de telle sorte à ce que la minuterie soit lancée lorsque la sortie du capteur à effet hall change d'état (passage de 0V à 5V ou inversement). En cas de mauvais placement de l'aimant sur la fusée ou de problème électronique, le capteur ne change pas d'état lors de la sortie de rampe et ce code ne permet pas d'informer l'équipe du problème. Il aurait été donc été plus judicieux de détecter l'état 0 ou 5V du capteur.

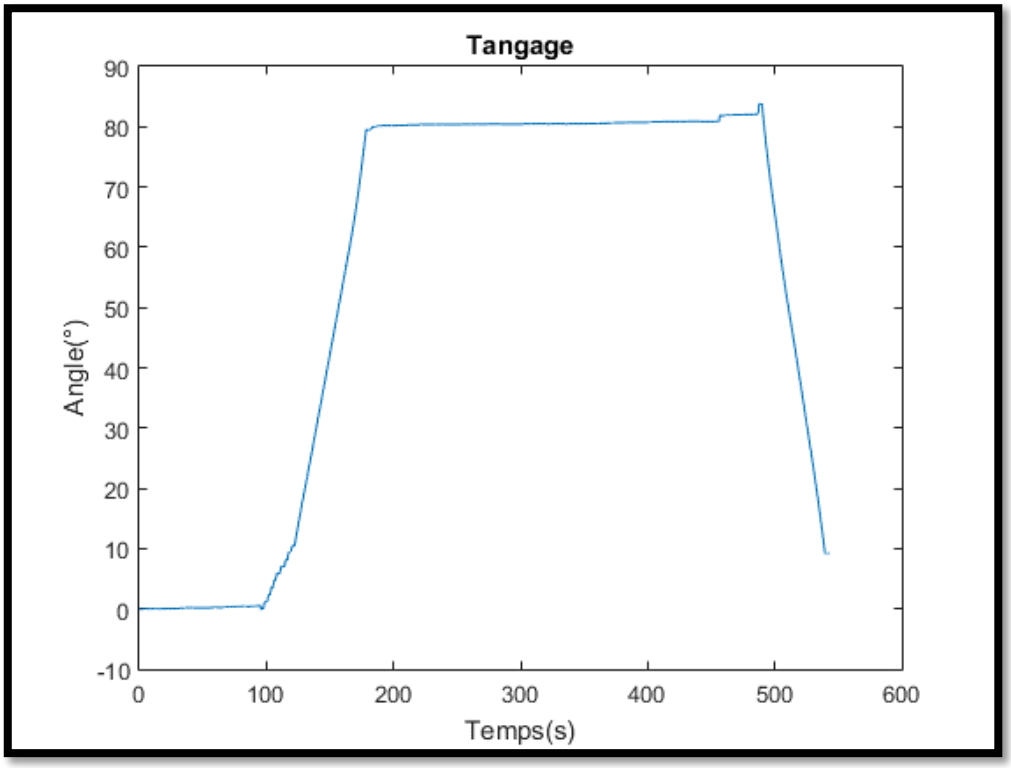
Au final les ventouses magnétiques n'ont pas libéré les poids et la porte cansat ne s'est pas ouverte.

Quant aux données de la centrale inertielle, elles ont pu être enregistrées sur la carte SD car l'enregistrement commençait dès la mise sous tension de la carte mais pour une raison inconnue, le vol ne figure pas sur les données enregistrées. Il faut noter que la carte a été éteinte et rallumée plusieurs fois en rampe dû à des problèmes au niveau du spatiobus. Très peu d'attention a été donné à l'enregistrement sur carte SD lors du développement du code, ainsi celui-ci consiste simplement à enregistrer les données sur un fichier et lors d'un rallumage de la carte, les nouvelles données sont simplement écrites à la suite du même fichier. Il est donc possible que la taille du fichier par exemple ai pu limiter la capacité d'enregistrement. Aucun test n'avait été mené sur une durée représentative du vol.

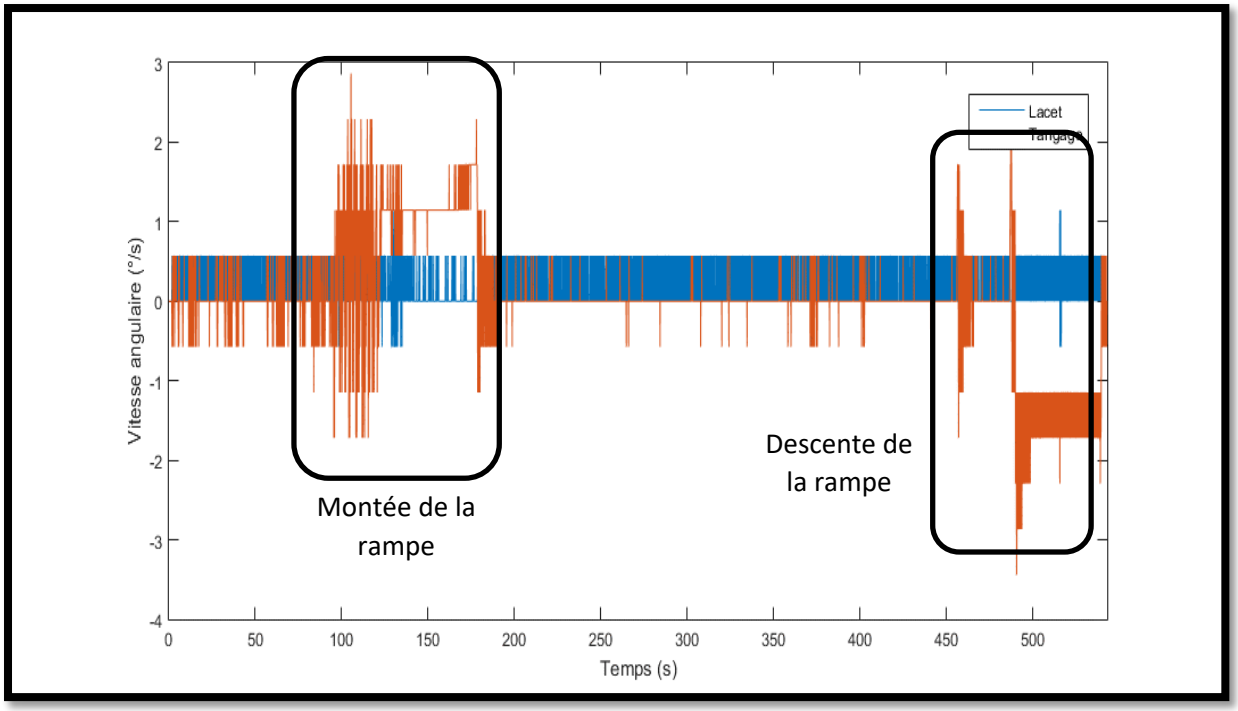
Malgré cela des données ont été enregistrées pendant la première mise en rampe de la fusée. En effet, en arrivant sur la zone de lancement, nous avions mis la fusée en rampe, érigé la rampe puis finalement redescendu la rampe à cause des problèmes de télémesure. Cet enregistrement dure environ 9 minutes. Les axes de la centrale inertielle étant alignés avec la rampe, on peut simplement intégrer les valeurs de vitesse angulaire pour obtenir le tangage (axe Y de la centrale inertielle), le roulis (axe Z) et le lacet (axe X). On obtient les résultats suivants :







On observe une très forte dérive sur les axes de roulis et lacet mais une excellente stabilité du tangage qui permet même d'observer la montée et la descente de la rampe. Si l'on regarde les données brutes de vitesse angulaires enregistrées pour comparer le lacet et le tangage (voir courbe ci-dessous), on remarque que la résolution de la mesure est très faible, l'écart minimum entre deux valeurs (environ 0.57°/s) est pratiquement du même ordre de grandeur que la vitesse angulaire mesurée lors de la phase de montée ou de descente de la rampe.



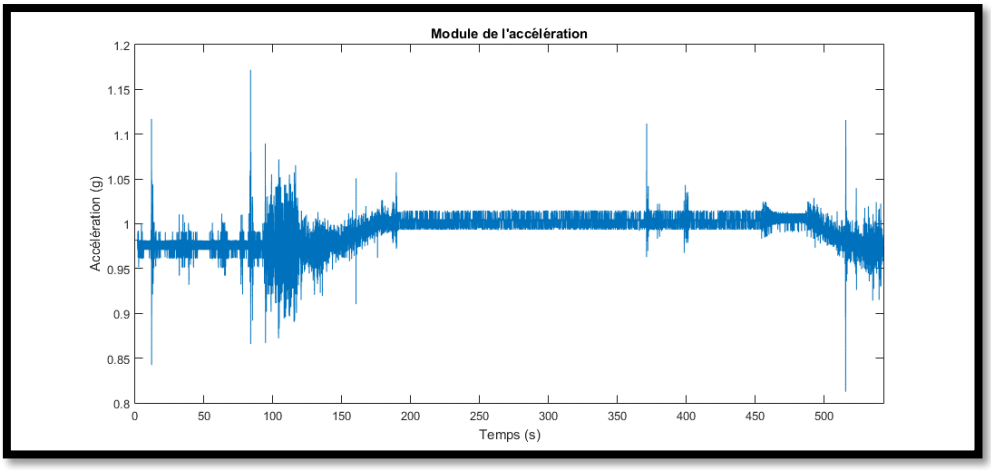
En réalité, cette imprécision est due au code et à la manière dont l'arduino enregistre des données sur la carte SD. En effet, le code converti les données brutes fournies par le gyromètre (des nombres compris entre 0 et 65535) en radians par seconde, car le code était initialement prévu pour effectuer d'autres calculs à partir de ces valeurs. Ces résultats sont ensuite directement envoyés sur la carte SD sous forme de chiffre à virgule mais l'arduino limite les nombres envoyés à 2 chiffres après la virgule. On a ainsi une précision de 0.01 rad/s, soit environ 0.57 °/s alors que la précision de la centrale inertielle est de 0.06 °/s. Ainsi, pour le lacet, la vitesse angulaire varie entre 0 et 0.01 rad/s ce qui suffit à faire dévier de 70° en 9 minutes et de 135° pour le roulis, dont la mesure est plus bruitée.

Cette réduction de la précision, qui n'a pas été anticipée, est aussi présente sur les résultats des accéléromètres. Les données d'accélération enregistrées sont en g et on a donc une précision de 0.01g au lieu des 0.0005g du capteur.

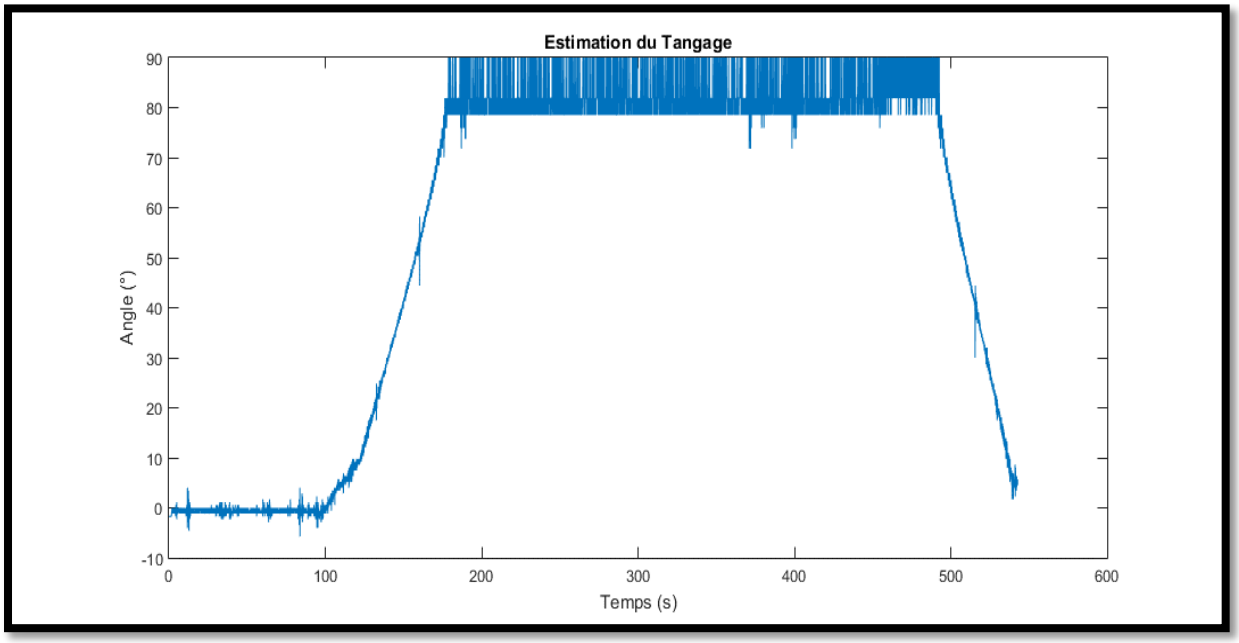
On peut obtenir une estimation du tangage à l'aide de la mesure de l'accélération dans l'axe de la fusée (axe Z de la centrale inertielle) avec la formule suivante :

$$\theta = \arcsin(g_z)$$

Cette méthode permet de vérifier les valeurs obtenues à partir des gyromètres et est souvent utilisée pour recaler ces derniers car elle n'utilise pas d'intégration et évite donc l'accumulation d'erreurs de mesures. Cependant elle ne peut être utilisée que lorsque la centrale inertielle est immobile ou pour des déplacements très faibles, ce qui est le cas sur la rampe (voir ci-dessous) mais pas en vol.



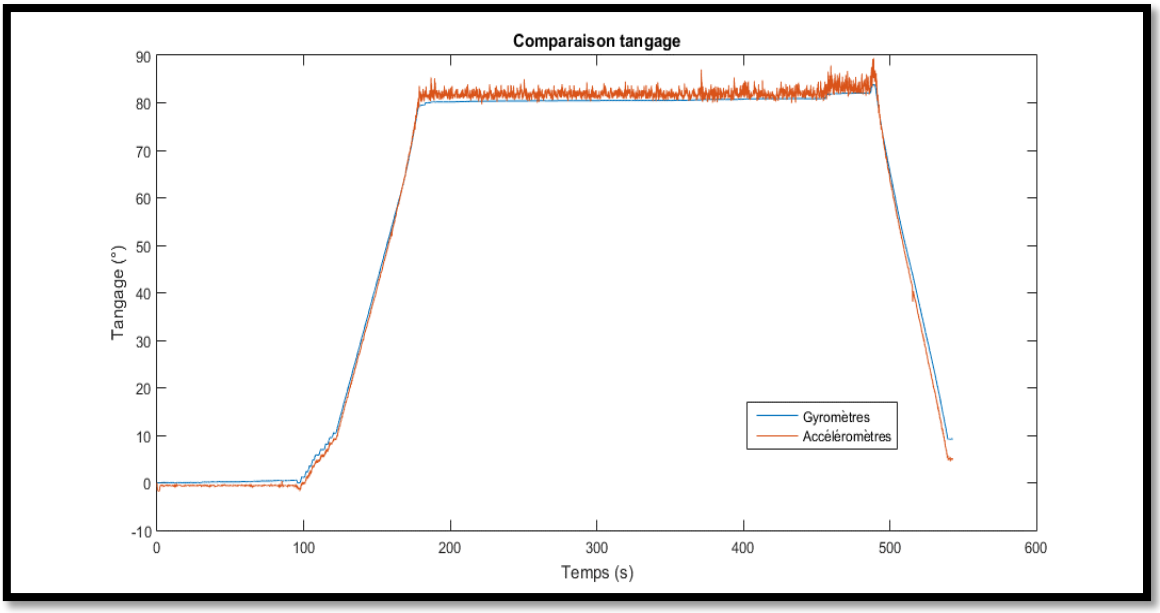
En appliquant cette formule aux données accélérométriques on obtient :



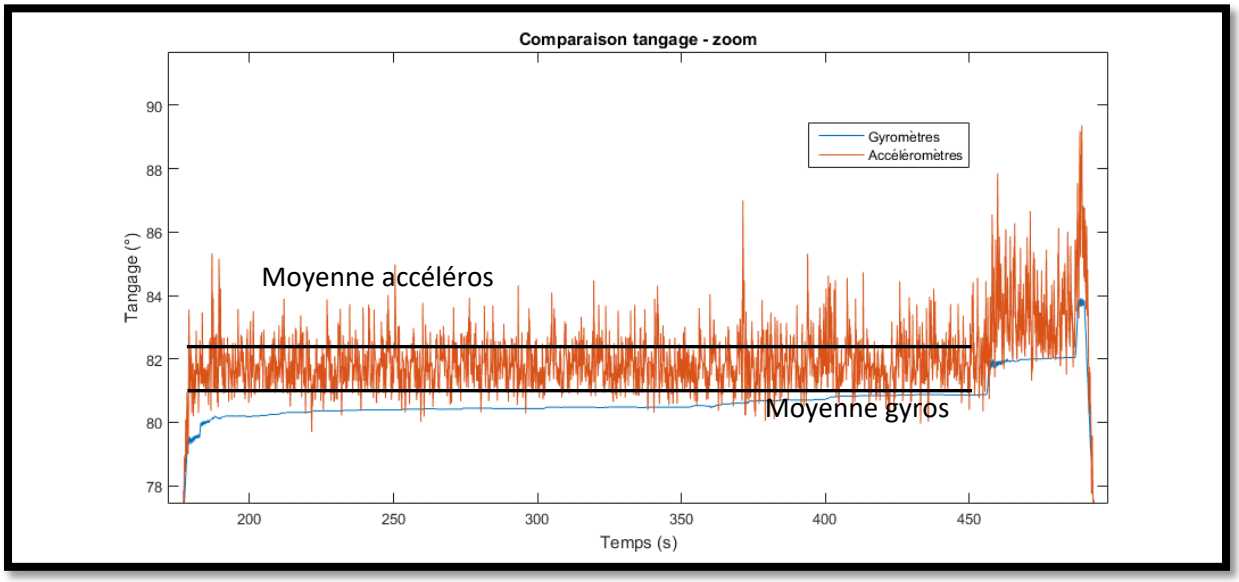
Tout d'abord on constate que la courbe est comparable à celle obtenue par intégration de la vitesse angulaire, ce qui renforce l'idée que nous observons bien la mise en rampe de la fusée et que cette courbe n'est pas due au bruit.

En pratique, pour estimer le tangage à l'aide de l'accélération longitudinale on applique un filtre passe bas sur les mesures pour éliminer les vibrations dues à la manipulation de la fusée et ne conserver que la composante continue de l'accélération de la pesanteur. Mais ces résultats permettent d'observer les problèmes liés à la précision de mesure. En effet, lorsque la rampe est complètement érigée, les valeurs oscillent entre 0.98 et 0.99 principalement et parfois 1g et ces valeurs correspondent à des angles de 78.5, 81.9 et 90° respectivement. On a donc une précision très faible !

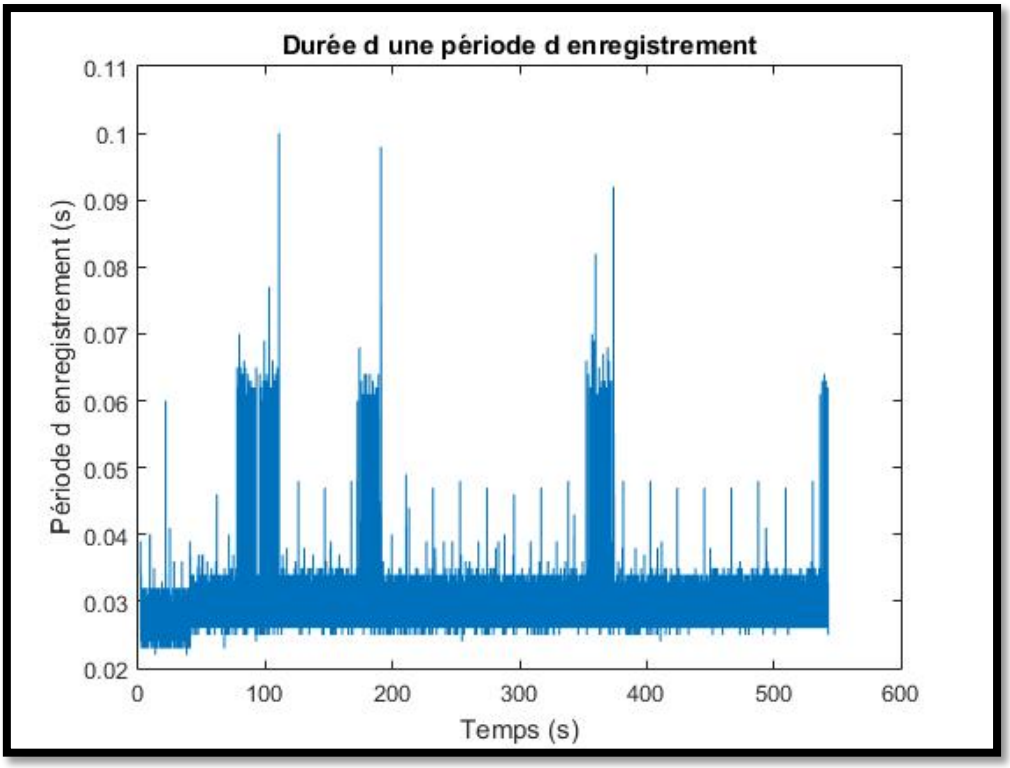
Ainsi, comme on peut le voir le voir sur la courbe ci-dessous, même les données filtrées (fréquence de coupure du filtre : 1Hz) sont très instables dans cette zone. La valeur moyenne de tangage est alors d'environ 82° et de 80° pour l'intégration de la vitesse angulaire.



Ce graphique représente en rouge le tangage estimé à partir des données accélérométriques (filtrées) et en bleu à partir des données gyrométriques. Les deux sont très proches notamment lors de la phase rampe érigée (voir ci-dessous), même si l'on observe une légère déviation du tangage estimé à partir des mesures gyrométriques (environ 1° sur 5 minutes).



Enfin, on peut constater d'après le graph ci-dessous que la période d'enregistrement est d'environ 28ms (soit 36Hz). Cette fréquence est relativement faible, les fréquences d'enregistrement minimales conseillées pour faire de la trajectographie inertielle sont de 150 - 200 Hz et on observe également que cette fréquence est assez instable dans notre cas, avec des pics de lag très importants. Il est donc nécessaire de travailler sur ce point, en se focalisant entre autres sur la vitesse d'enregistrement de la carte SD qui est un des principaux facteurs limitant la fréquence d'enregistrement.



Télémesure

Lors de la mise en rampe, des données GPS ont été reçues pendant quelques minutes puis la télémesure a émis « NO SIGNAL » jusqu'à la fin du vol, signifiant normalement que le GPS n'a pas de fix. En effet, le code implémenté sur le PIC utilise un caractère spécial de la trame GPS qui indique si le GPS possède un fix ou non. Si oui, il envoie les données provenant du GPS, sinon il envoie « NO SIGNAL ». Il est peu probable que le problème soit lié au GPS car il a correctement fonctionné lors de nombreux tests dont un durant 1 heure.

Malheureusement, l'erreur commise est d'avoir changé le code après ces tests pour exploiter une autre trame GPS. Cette trame GPS ne possédant pas le même code de détection de fix, il est possible que son fonctionnement ait été mal compris, entraînant alors l'émission systématique d'un « NO SIGNAL » quelques minutes après la mise sous tension de la carte.

RETOUR D'EXPÉRIENCE

Tout d'abord, il est évident que les problèmes décrits dans les précédents paragraphes auraient pu être évités avec plus d'organisation et moins de précipitation lors du C'Space. Ces problèmes sont dans les deux cas liés au code implémenté sur l'Arduino et le PIC :

- Dans le cas de l'Arduino, le problème est dû à une mauvaise anticipation des causes possibles de défaillance. En effet, il aurait fallu une indication visuelle indiquant que la présence de l'aimant n'est pas détectée. Une analyse des modes défaillance, même relativement simpliste, permettrai d'éviter ce genre de problème.
- Dans le cas du PIC, le logiciel a été changé au dernier moment et sans réaliser de tests approfondis et cette précipitation est probablement la cause de l'échec de la télémesure. Il est donc nécessaire d'être prudent lors de modifications du code et de réaliser des tests représentatifs du vol pour chaque version du logiciel.

Au cours de l'année, nous avons rencontré divers problèmes plus « classiques » :

Les connecteurs ont été une source incessante de problèmes. Nous avons utilisé pour toutes nos cartes des connecteurs « volants », qui ne sont pas fixé sur la carte ou sur la structure et reliés à la carte par des fils multibrins classiques. Ces connecteurs ont généré à répétition des ruptures de soudure, soit au niveau du connecteur à force de manipulation régulières, soit au niveau de la soudure des fils sur la carte à force de tirer dessus à chaque branchement/débranchement. Il serait probablement plus intéressant d'utiliser des connecteurs soudés sur la carte ou même de réaliser une boîte pour chaque carte électronique, avec des connecteurs fixés sur la boîte.

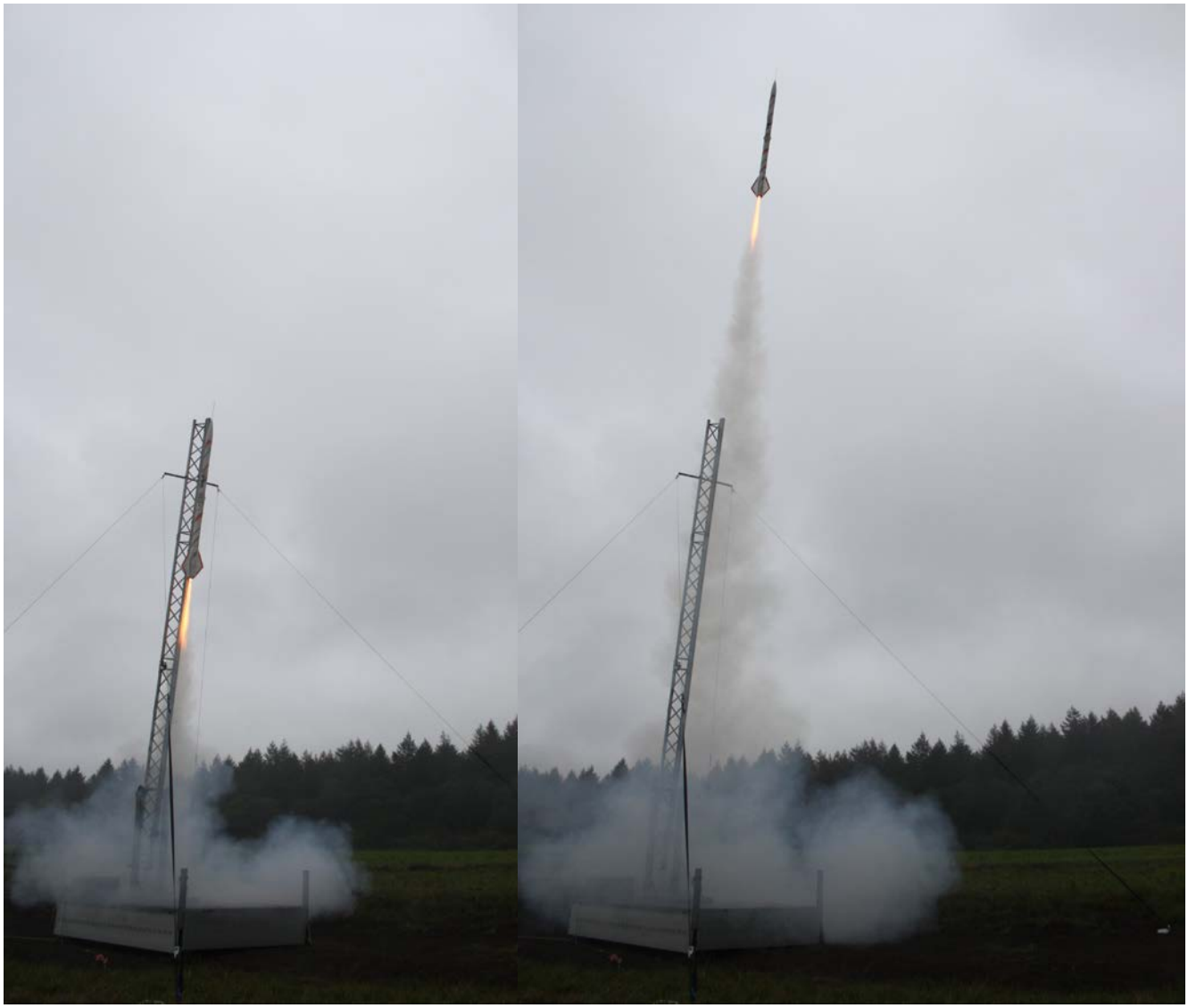
Aucun plan de câblage n'a été fait dans l'année et c'est donc sans aucune anticipation que l'électronique a été câblée lors de l'intégration. Cela a entraîné une perte de temps et le résultat, plutôt chaotique, peut être la source de problèmes électroniques.

Enfin l'intégration de l'électronique a été effectuée très tard dans l'année et le temps nécessaire pour cette étape a largement été sous-estimé. Les problèmes apparus lors de l'intégration n'ont pas pu être résolus à temps avant que les membres de l'équipe électronique partent en stage. Nous avons donc dû finir l'intégration au C'Space.

Du côté mécanique, des erreurs de conception ont été faîtes sur les treillis porteurs de la fusée (treillis U 11,5x11,5 aluminium de 2m50). AERIS I avait initialement trop de flèche et des renforts en acier ont été nécessaires afin de respecter la tolérance mise en place par le cahier des charges de l'organisme qui nous encadre durant la campagne de lancements. D'autres modifications mineures ont aussi été nécessaires durant la semaine précédant le lancement.

Finalement, malgré l'échec de l'expérience, nous avons beaucoup appris en termes d'organisation et de rigueur. Forts de cette expérience, les membres d'AERIS sont donc prêts à relancer un yoyo despin avec AERIS II en 2018 !

CAMPAGNE DE LANCEMENT



CHRONOLOGIE

Chronologie Aeris (1/4) :

VEILLE DU LANCEMENT - Lecteur : Killian	
Caméra	
→ Vider carte SD et assurer son bon fonctionnement	
→ Régler la caméra sur le mode voulu (120 fps)	
→ Recharger batterie caméra	
De-Spin	
→ Vérifier état du câble (en recouper si besoin est)	
→ Vérifier la présence des deux masses	
Récupération	
→ Vérifier tout le cordage (attachement parachute, portes)	
→ Vérifier tout le cordage (Maintien des pièces avec la fusée)	
→ Vérifier la présence des ressorts de l'étage Cansat	
→ Vérifier la présence du ressort de l'étage parachute	
Etage parachute	
→ Ouvrir la porte parachute	
→ Plier le parachute (en accordéon)	
→ Introduire le parachute	
→ Apposer la tige sur l'électro-aimant puis appuyer la porte et v	
→ Vérifier que rien ne dépasse et secouer	
Etage Cansat (test)	
→ Ouvrir la porte Cansat	
→ Plier le parachute du Cansat	
→ Introduire le Cansat	
→ Fermer la porte Cansat soigneusement (rien ne dépasse)	
Alimentation générale	
→ Insérer 10 piles 9V	
→ Tester tension piles	
Méca	
→ Resserrer toutes les vis	
→ Vérifier maintien des électro-aimants	
Expérience	
→ Mettre carte SD	

Chronologie Aeris (2/4) :

Matos à prendre	
→ 2 jeux de clé Allen (3 et 4)	
→ Jeux de tournevis plats (un petit pour dominos, un pour vis M4)	
→ Jeu de tournevis cruciforme	
→ Clés de 5,5	
→ Clés de 7	
→ Vis M3/M4 têtes plates	
→ Vis M3/M4 têtes bombées	
→ Rondelles diverses	
→ Ecrous M3/M4	
→ Lime	
→ Corde	
→ Grosse pince	
→ Patin de rechange	
→ Multimètre	
→ Pinces multiprise	
→ 5 piles 9V neuves	
→ Insert maintien Pro-54	
→ Jeux de clés mixtes	
→ Scotch (élec + peintre)	
→ Marker	
→ Ciseaux	

Chronologie Aeris (3/4) :

EN TENTE CLUB - Lecteur : Killian		
→ Placer aimant Hall sur la rondelle (minuterie para)	Pierre-Louis	
→ Placer aimant Hall sur la rondelle (minuterie expé)	Pierre-Louis	
→ Allumer interrupteur minuterie parachute	Pierre-Louis	
→ Vérifier LED verte minuterie parachute	Pierre-Louis	
→ Allumer interrupteur expé	Pierre-Louis	
→ Attendre allumage fixe LED bleue ET led rouge expé	Pierre-Louis	
→ Tenir porte Cansat	Robin	
→ Tenir les masses et légèrement les soulever	Rémi	
→ Retirer les aimants (ATTENTION à la bague en résine !!!)	Valentin et Augustin	
→ Vérifier clignotement leds rouge et bleue puis : - led rouge allumée et led bleue éteinte avec libération des poids à 11 sec - led rouge allumée et led bleue allumée avec éjection cansat à 14 sec - extinction led rouge à 21 sec et led bleue à 24 sec	Augustin	
→ Vérifier LED jaune minuterie parachute puis led rouge et éjection para après 16 secondes	Valentin	
→ Mettre tous les interrupteurs sur OFF	Valentin	
→ Plier parachute et vérifier que rien ne dépasse	Rémi	
→ Fermer la porte parachute	Robin	
→ Retourner fusée, case Cansat vers le haut	Tout le monde	
Cansat		
→ Ouvrir porte Cansat	Robin	
→ Allumer Cansat	Williams	
→ Plier parachute Cansat	Rémi et Williams	
→ Introduire Cansat Ol Del Paso	Rémi et Williams	
→ Apposer la tige sur l'électro-aimant puis appuyer la porte et visser (vérifier pas de suspente coincée)	Robin	
→ Vérifier que rien ne dépasse et secouer	Robin	
De-Spin		
→ Enrouler les masses autour de la bague et les fixer sur les	Pierre-Louis	
→ Malmener les masses pour vérifier la bonne fixation de	Pierre-Louis	

Chronologie Aeris (4/4) :

EN RAMPE - Lecteur : Rémi		
/!\ Suivre la chronologie calmement, pas de précipitation	Tout le monde	
Allumer caméra ! #NotAgain	Killian	
Vérifier mode	Killian	
Commencer enregistrement et vérifier LED rouge	Killian	
→ Positionner interrupteurs télem en position ON	Clément	
→ Positionner interrupteur PIC en postion VOL	Clément	
→ Tester la réception de la télémesure avec le spatiobus	Clément	
→ Mettre la fusée en rampe (axe des patins dans le profilé !!!)	Tout le monde	
→ Regarder si la fusée coulisse dans le rail en la poussant	Tout le monde	
→ Poser l'aimant "Hall parachute" sur la rondelle	Clément	
→ Poser l'aimant "Hall expérience" sur la rondelle	Clément	
→ Accrocher la corde de l'aimant "Hall parachute" à la rampe (nœud coulant)	Killian	
→ Accrocher la corde de l'aimant "Hall expérience" à la rampe (nœud coulant)	Killian	
→ Allumer interrupteur minuterie parachute	Valentin	
→ Vérifier LED verte minuterie parachute	Valentin	
→ Allumer interrupteur expérience	Augustin	
→ Attendre allumage fixe LED bleue ET led rouge expé	Augustin	
Etage propulsif		
→ Montrer au pyro la méthode d'insertion de la goupille de maintien	Robin	

Annexe 1 : schémas de la carte expérience

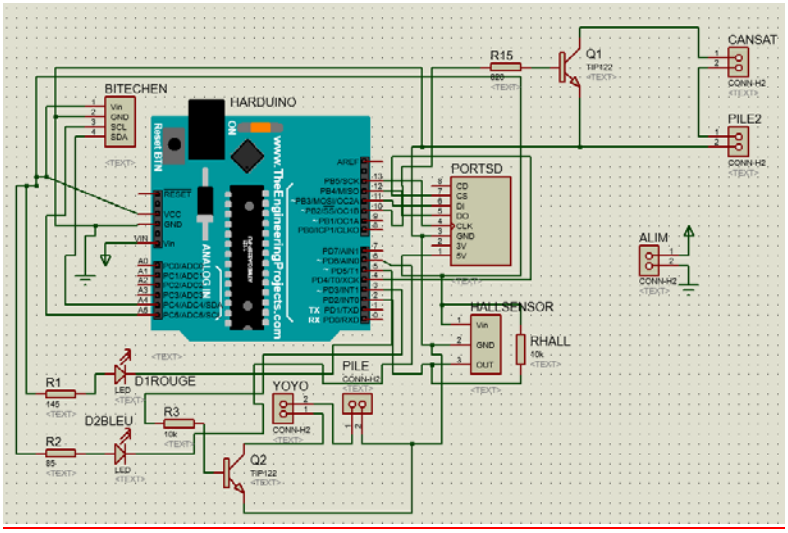


Figure 9: Carte Arduino

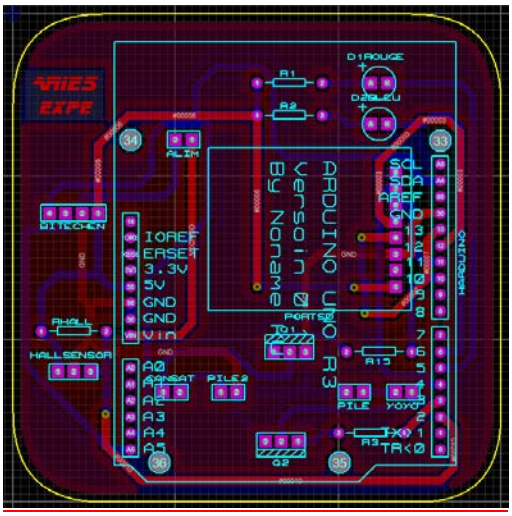


Figure 10: Carte Arduino - routage

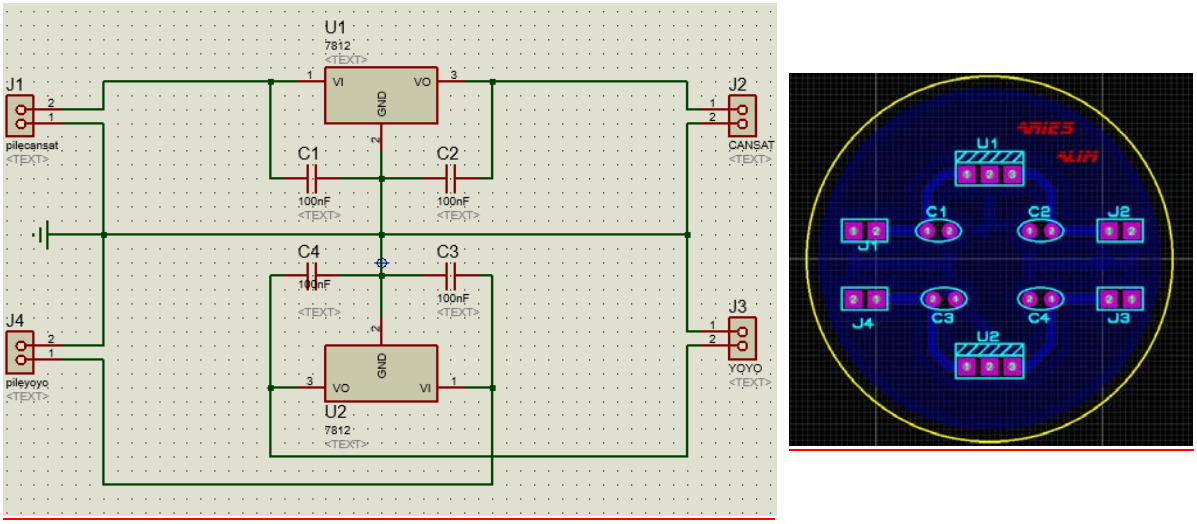
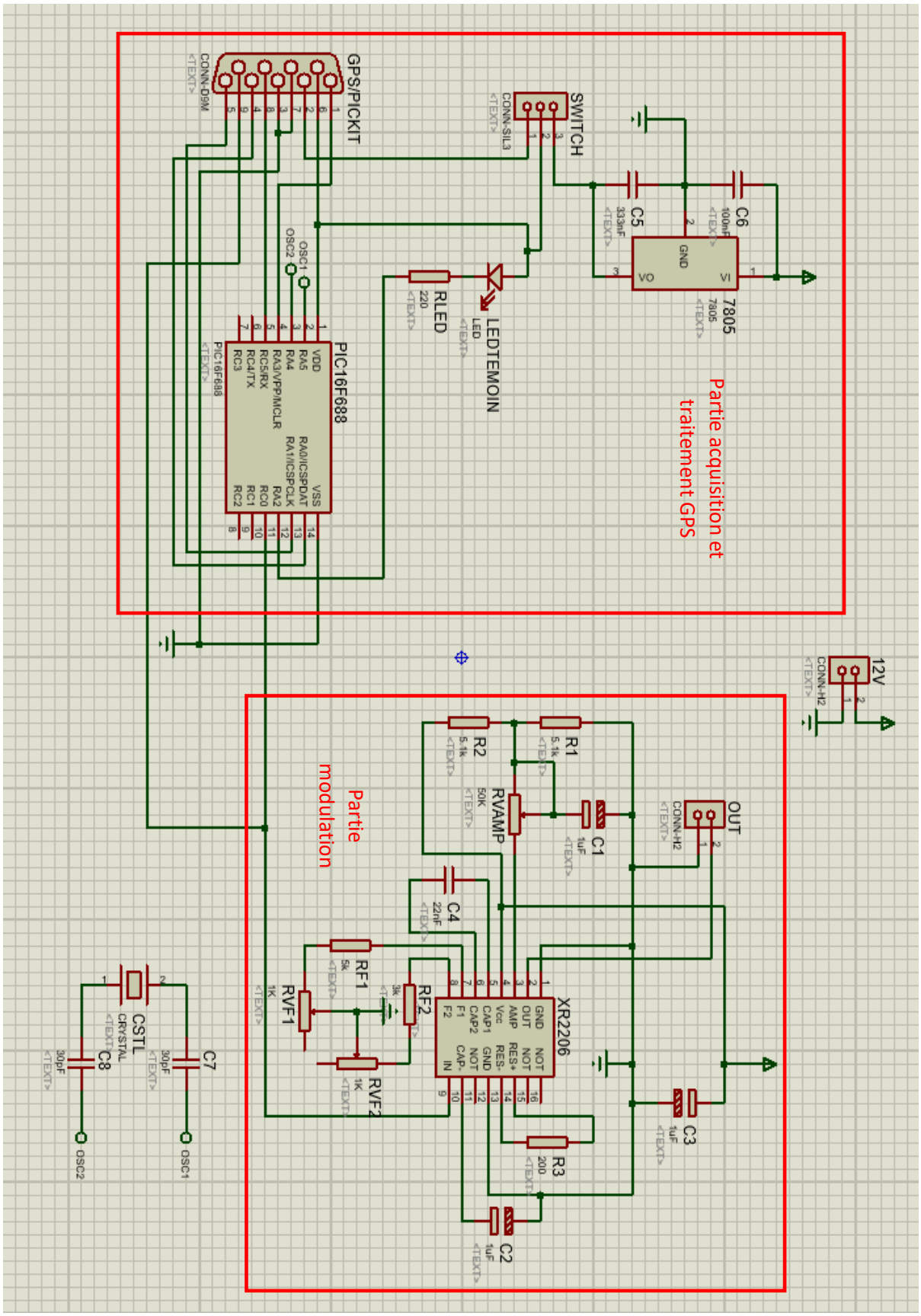


Figure 11: Schémas de la carte d'alimentation

Annexe 2 : schémas de la carte télémesure



Annexe 3 : Code Arduino

```
#include <Wire.h>
#include <math.h>
#include <SPI.h>
#include <SD.h>

//SD
File myFile;

//Kalman - coefficients
double K, R=0.001, Q=0.005;
double PGx=1, PGy=1, PGz=1;
double PAx=1, PAy=1, PAz=1;

//Kalman - sorties
double rX=0, rY=0, rZ=0;
double aX=0, aY=0, aZ=0;

//Mesures
double rX_mes=0, rY_mes=0, rZ_mes=0;
double aX_mes=0, aY_mes=0, aZ_mes=0;

//Minuterie
int cansat = 10;
int yoyo = 3;
int rouge = 5;
int bleu = 6;
const byte interruptPin=2;
boolean ledBlink = false;
int state = 0;
```

```
unsigned long compteur=0;
unsigned char phase=0;
unsigned long carotte=0;
unsigned long SDtimer=0;

void setup() {
 pinMode(interruptPin,INPUT);
 pinMode(rouge, OUTPUT);
 pinMode(bleu, OUTPUT);
 pinMode(yoyo,OUTPUT);
 pinMode(cansat,OUTPUT);
 digitalWrite(rouge,HIGH);
 digitalWrite(bleu,HIGH);
 digitalWrite(yoyo,LOW);
 digitalWrite(cansat,LOW);

 //Initialisation MPU
 digitalWrite(rouge, LOW); //LED rouge allumée signifie que l'initialisation MPU est en cours
 digitalWrite(bleu, HIGH);
 Wire.begin();
 setupMPU();
 for(int i=0;i<500;i++) //On fait tourner le Kalman pour le faire converger
 {
  getGyro();
  kalmanG();
  getAccel();
  kalmanA();
 }

 //Initialisation SD
 digitalWrite(rouge, HIGH);
```

```
digitalWrite(bleu, LOW); //LED bleue allumée signifie que l'initialisation SD est en cours
if (!SD.begin(4)) // Si l'initialisation échoue, arrêt du programme et clignottement de la LED bleue
{
 compteur=millis();
 while(1)
 {
  if((millis()-compteur)<500)
  {
   digitalWrite(bleu, LOW);
  }
  else if((millis()-compteur)>500)
  {
   digitalWrite(bleu, HIGH);
  }
  if((millis()-compteur)>1000)
  {
   compteur=millis();
  }
 }
}

myFile = SD.open("Rslt.txt", FILE_WRITE);
if (myFile)
{
 myFile.println("-----     Début du programme     -----");
}
else
{
 compteur=millis();
 while(1)
 {
```

```
    if((millis()-compteur)<500)
    {
      digitalWrite(bleu, LOW);
    }
    else if((millis()-compteur)>500)
    {
      digitalWrite(bleu, HIGH);
    }
    if((millis()-compteur)>1000)
    {
      compteur=millis();
    }
   }
  }
  state = digitalRead(interruptPin); //Détection de l'état initial du capteur à effet hall
  SDtimer = millis();
}

void loop() {
  switch (phase) {
    case 0 : //Phase 0 -> en rampe
      digitalWrite(rouge, LOW);
      digitalWrite(bleu, LOW);
      break;
    case 1 : //Phase 1 -> sortie de rampe
      if((millis()-carotte)>11000)
      {
        ledBlink=false;
        digitalWrite(rouge, LOW);
        digitalWrite(bleu, HIGH);
        digitalWrite(yoyo, HIGH);
```

```
   phase=2;
   myFile.println("-----     T0+11s     -----");
  }
  break;
case 2 : //Phase 2 -> après déclenchement yoyo
  if((millis()-carotte)>14000)
  {
   digitalWrite(bleu, LOW);
   digitalWrite(cansat, HIGH);
   phase=3;
   myFile.println("-----     T0+14s     -----");
  }
  break;
case 3 : //Phase 3 -> après éjection cansat
  if((millis()-carotte)>21000)
  {
   digitalWrite(rouge, HIGH);
   digitalWrite(yoyo, LOW);
   phase=4;
   myFile.println("-----     T0+21s     -----");
  }
  break;
case 4 : //Phase 4 -> après arrêt de l'alimentation yoyo
  if((millis()-carotte)>24000)
  {
   digitalWrite(bleu, HIGH);
   digitalWrite(cansat, LOW);
   phase=5;
   myFile.println("-----     T0+24s     -----");
  }
  break;
```

```
  case 5 : //Phase 5 -> après l'arrêt de l'alimentation ventouse cansat
    if((millis()-carotte)>180000)
    {
      digitalWrite(bleu, HIGH);
      digitalWrite(cansat, LOW);
      myFile.println("-----     FIN DU PROGRAMME     -----");
      myFile.close();
      while(1);
    }
    break;
}

if(ledBlink) //Clignottement des leds pour indiquer que la sortie de rampe est détectée
{
  if((millis()-compteur)<500)
  {
    digitalWrite(bleu, HIGH);
    digitalWrite(rouge, HIGH);
  }
  else if((millis()-compteur)>500)
  {
    digitalWrite(bleu, LOW);
    digitalWrite(rouge, LOW);
  }
  if((millis()-compteur)>1000)
  {
    compteur=millis();
  }
}

if(digitalRead(interruptPin) != state) //Détection de changement d'état capteur à effet hall
```

```
 {
  if(phase == 0) //Si on est en rampe alors on déclenche la minuterie
  {
   myFile.println("-----     T0     -----");
   carotte=millis();
   phase=1;
   ledBlink=true;
   compteur=millis();
  }
  else //Sinon on réinitialise la minuterie et retour en phase 0 (utilisé pour tests au sol)
  {
   myFile.println("-----     Reset     -----");
   digitalWrite(rouge,HIGH);
   digitalWrite(bleu,HIGH);
   digitalWrite(yoyo,LOW);
   digitalWrite(cansat,LOW);
   phase=0;
   ledBlink=false;
  }
  state = digitalRead(interruptPin);
 }

 WriteMPU();
 if((millis()-SDtimer)<5000) //Fermeture périodique du fichier pour enregistrement
 {
  myFile.close();
  myFile = SD.open("Rslt.txt", FILE_WRITE);
  SDtimer = millis();
 }
}
```

```
void setupMPU() { //Initialisation des paramètres du MPU
  Wire.beginTransmission(0b1101000);
  Wire.write(0x6B); // Registre PWR_MGMT_1
  Wire.write(0x00); // Sortie du sleep mode
  Wire.endTransmission();

  Wire.beginTransmission(0b1101000);
  Wire.write(0x1B); // Registre GYRO_CONFIG
  Wire.write(0b00011000); // Full Scale à +-2000°/s
  Wire.endTransmission();

  Wire.beginTransmission(0b1101000);
  Wire.write(0x1C); // Registre ACCEL_CONFIG
  Wire.write(0b00011000); // Full Scale à +-16g
  Wire.endTransmission();
}

void getGyro() {
  Wire.beginTransmission(0b1101000);
  Wire.write(0x43); // Début des registres GYRO_OUT
  Wire.endTransmission();
  Wire.requestFrom(0b1101000,6); // Registre GYRO_XOUT_H à GYRO_ZOUT_L
  while(Wire.available() < 6);
  rX_mes = ((Wire.read()<<8|Wire.read())*M_PI/2952)+0.03102; //+0.03102
  rY_mes = ((Wire.read()<<8|Wire.read())*M_PI/2952)-0.008975; //-0.008975
  rZ_mes = ((Wire.read()<<8|Wire.read())*M_PI/2952)+0.0104; //+0.0104
}

void kalmanG() {
  K=(PGx+Q)/(PGx+Q+R);
  rX=rX+K*(rX_mes-rX);
```

```
  PGx=(1-K)*(PGx+Q);
  K=(PGy+Q)/(PGy+Q+R);
  rY=rY+K*(rY_mes-rY);
  PGy=(1-K)*(PGy+Q);
  K=(PGz+Q)/(PGz+Q+R);
  rZ=rZ+K*(rZ_mes-rZ);
  PGz=(1-K)*(PGz+Q);
}

void getAccel() {
  Wire.beginTransmission(0b1101000);
  Wire.write(0x3B); // Début des registres ACCEL_OUT
  Wire.endTransmission();
  Wire.requestFrom(0b1101000,6); // Registre ACCEL_XOUT_H à ACCEL_ZOUT_L
  while(Wire.available() < 6);
  aX_mes = ((Wire.read()<<8|Wire.read())/2048.0);
  aY_mes = ((Wire.read()<<8|Wire.read())/2048.0);
  aZ_mes = ((Wire.read()<<8|Wire.read())/2048.0);
}

void kalmanA() {
  K=(PAx+Q)/(PAx+Q+R);
  aX=aX+K*(aX_mes-aX);
  PAx=(1-K)*(PAx+Q);
  K=(PAy+Q)/(PAy+Q+R);
  aY=aY+K*(aY_mes-aY);
  PAy=(1-K)*(PAy+Q);
  K=(PAz+Q)/(PAz+Q+R);
  aZ=aZ+K*(aZ_mes-aZ);
  PAz=(1-K)*(PAz+Q);
}
```

```
void WriteMPU()
{
 getGyro();
 kalmanG();
 getAccel();
 kalmanA();
 if(phase == 0)
 {
  myFile.print(phase); myFile.print("\t");
  myFile.print(millis()); myFile.print("\t");
  myFile.print(aX); myFile.print("\t");
  myFile.print(aY); myFile.print("\t");
  myFile.print(aZ); myFile.print("\t");
  myFile.print(rX); myFile.print("\t");
  myFile.print(rY); myFile.print("\t");
  myFile.println(rZ);
 }
 else
 {
  myFile.print(phase); myFile.print("\t");
  myFile.print(millis()-carotte); myFile.print("\t");
  myFile.print(aX); myFile.print("\t");
  myFile.print(aY); myFile.print("\t");
  myFile.print(aZ); myFile.print("\t");
  myFile.print(rX); myFile.print("\t");
  myFile.print(rY); myFile.print("\t");
  myFile.println(rZ);
 }
}
```

Annexe 4 : Code PIC

```
/*
 * File:   newmain.c
 * Author: ClémentROUSSEAU
 *
 * Created on 12 décembre 2016, 17:31
 */

#pragma config FOSC = INTOSCCLK  // Oscillator Selection bits (INTOSCIO oscillator: I/O function on RA4/OSC2/CLKOUT pin, I/O function on RA5/OSC1/CLKIN)
#pragma config WDTE = OFF       // Watchdog Timer Enable bit (WDT disabled)
#pragma config PWRTE = OFF      // Power-up Timer Enable bit (PWRT disabled)
#pragma config MCLRE = ON       // MCLR Pin Function Select bit (MCLR pin function is MCLR)
#pragma config CP = OFF         // Code Protection bit (Program memory code protection is disabled)
#pragma config CPD = OFF        // Data Code Protection bit (Data memory code protection is disabled)
#pragma config BOREN = ON       // Brown Out Detect (BOR enabled)
#pragma config IESO = ON        // Internal External Switchover bit (Internal External Switchover mode is enabled)
#pragma config FCMEN = ON       // Fail-Safe Clock Monitor Enabled bit (Fail-Safe Clock Monitor is enabled)

#include <xc.h> //On utilise le compilateur xc8

void initUSART();
void initClock();
void initTimer();
void initIO();
void wait(unsigned char j);
void send(unsigned char caractere);
void skipChar(unsigned char n);

void initUSART() //Paramétrage de l'USART pour la réception
```

```
{
   TXSTAbits.SYNC=0;
   RCSTAbits.SPEN=1;
   BAUDCTLbits.BRG16=0;
   TXSTAbits.BRGH=0;
   SPBRGH=12; //
   SPBRG=12;  //9600 bauds
   PIE1bits.RCIE=1;
   RCSTAbits.CREN=1;
}

void initClock() //Paramétrage de l'oscillateur
{
   OSCCONbits.IRCF2=1;
   OSCCONbits.IRCF1=1;
   OSCCONbits.IRCF0=1;
}

void initTimer() //Paramétrage de Timer0
{
   OPTION_REGbits.T0CS=0;
   OPTION_REGbits.PSA=1;
}

void initIO() //Initialisation des entrées/sorties
{
   TRISCbits.TRISC0=0;
   PORTCbits.RC0=0;
   TRISAbits.TRISA2=0;
   PORTAbits.RA2=1;
}
```

```
void wait(unsigned char j) //Délai permettant de régler finement la durée d'émission d'un bit
{
   TMR0=-j; //Valeur à modifier pour régler la durée d'attente
   INTCONbits.T0IF=0; //Mise à zéro du flag d'overflow de Timer0
   while(INTCONbits.T0IF == 0); //Lorsque TMR0=255, T0IF=1
}

void send(unsigned char caractere) //Transmet un caractère au protocole UART
{
   unsigned char delay=195;
   PORTCbits.RC0=0; //Start
   wait(255); //
   wait(111); //Délai nécessaire au bit de start
   for(unsigned char i=0;i<8;i++)
   {
      if((caractere >> i) & 1) //Test du 'ième' bit
      {
         PORTCbits.RC0=1; //Emission d'un 1 logique
         wait(155);
         wait(delay);
         delay -= 6; //Diminution du délai à chaque itération car
      }           //le nombre de décalages nécessaires augmente à chaque boucle
      else
      {
         PORTCbits.RC0=0; //Emission d'un 0 logique
         wait(155);
         wait(delay);
         delay -= 6;
      }
   }
```

```
    wait(42);
    PORTCbits.RC0=1; //Stop
    wait(255); //Délai nécessaire au bit de stop
    wait(124);
}

void skipChar(unsigned char n) //Passe les caractères non désirés
{
    for(int i=0;i<n;i++)
    {
        while(PIR1bits.RCIF == 0); //Lorsque l'on détecte un caractère dans le buffer
        RCREG; //On le lit pour effacer le buffer
    }
}

void main(void) {
    initUSART();
    initClock();
    initTimer();
    initIO();
    int i=0;
    unsigned char message[55]={0}; //Contient les données à envoyer

    while(1)
    {
        if(PIR1bits.RCIF == 1 && RCSTAbits.OERR ==0) //Détection de la présence d'un caractère dans le buffer UART
        {
            if(RCREG == '$') //Détection de début de trame GPS
            {
                skipChar(3);
```

```
while(PIR1bits.RCIF == 0);
if(RCREG == 'G') //Détection de la trame GPGGA
{
   skipChar(2);
   while(PIR1bits.RCIF == 0);
   for(i=0;i<36;i++) //heure, latitude, longitude
   {
      while(PIR1bits.RCIF == 0);
      message[i]=RCREG;
   }
   while(PIR1bits.RCIF == 0);
   if(RCREG == '1') //détection du fix GPS
   {
      PORTAbits.RA2=0;
      skipChar(1);
      for(i=36;i<38;i++) //nombre de satellites
      {
         while(PIR1bits.RCIF == 0);
         message[i]=RCREG;
      }
      skipChar(5);
      for(i=38;i<53;i++) //altitude mer et altitude terrain
      {
         while(PIR1bits.RCIF == 0);
         message[i]=RCREG;
      }
      for(i=0;i<53;i++)
      {
         send(message[i]); //envoi des données
      }
      send('\r'); //fin de trame
```

```
                send('\n');
            }
            else //Pas de fix GPS
            {
                PORTAbits.RA2=1;
                send('N');
                send('O');
                send(' ');
                send('S');
                send('I');
                send('G');
                send('N');
                send('A');
                send('L');
                send('\r');
                send('\n');
            }
          }
        }
      }
      else if(RCSTAbits.OERR == 1) //Détection d'overflow du buffer UART
      {
        RCSTAbits.CREN=0; //Réinitialisation de l'UART
        RCSTAbits.CREN=1;
      }
    }

    return;
}
```