

Rapport de projet

Fusée expérimentale Bellatrix Club INS'ASTRO

Sommaire

Sommaire	1
1) Introduction	3
2) Expérience	4
3) Mécanique	5
a) Structure :	5
b) Aérodynamique :	8
Ogive :	8
Ailerons	9
Parachute	9
4) Electronique	12
a) Séquenceur	12
b) chaîne de mesure	15
5) Coûts	18
a) prévisionnel	18
b) Financement	18
6) Cspace	19
7) Résultats	20
a) Vol	20
b) Expérience	20
c) Recommendations	20
9) Conclusion	21

1)Introduction

INS'ASTRO est le club astronomie et espace de l'INSA Rouen Normandie.

Au début de cette année, nous avons décidé d'avoir pour objectif la création d'un engin spatial réutilisable, une fusée expérimentale (FUSEX), et de la lancer au C'Space, un événement organisé annuellement par le CNES.

Notre projet est de concevoir, fabriquer et tester les différentes pièces composant la fusée, (structure, parachute, chaîne de mesure) les assembler, puis utiliser le vecteur pour réaliser différentes expériences. Dans notre cas, étant donné que ce sera le premier lancement, nous avons décidé de réaliser une étude sur le vecteur afin de connaître précisément ses propriétés. Nous avons également pour idée de tester le confort à l'intérieur du compartiment de charge utile, en disposant un œuf à l'intérieur et en s'assurant qu'il revienne dans un état similaire à celui antérieur au lancement.

Notre fusée devra être réutilisable afin de permettre son optimisation par les futures générations d'étudiants qui voudront se prêter à l'expérience, cette idée nous guidera durant la totalité de notre conception.

Notre équipe est composée d'une dizaine d'étudiants de différents départements afin de répondre à la diversité des compétences techniques (mécanique, aérodynamique, électronique, informatique) et organisationnelles nécessaires à la bonne réalisation du projet.

2)Expérience

Notre expérience principale est l'étude et la détermination des propriétés de notre fusex.

Pour cela nous avons besoin de connaître plusieurs paramètres: la position, la vitesse, l'accélération.

Afin de prendre les mesures nécessaires plusieurs capteurs sont installés à l'intérieur de la fusée: baromètre, accéléromètre et gyromètre, leur mise en œuvre, la récolte et le transfert de données est détaillée dans la partie "chaîne de mesure" de ce document.

Grâce aux données collectées nous pourrons

- déterminer la trajectoire de notre vol
- estimer le coefficient de frottement de la fusée
- vérifier la poussée réelle du moteur

3) Mécanique

a) Structure :

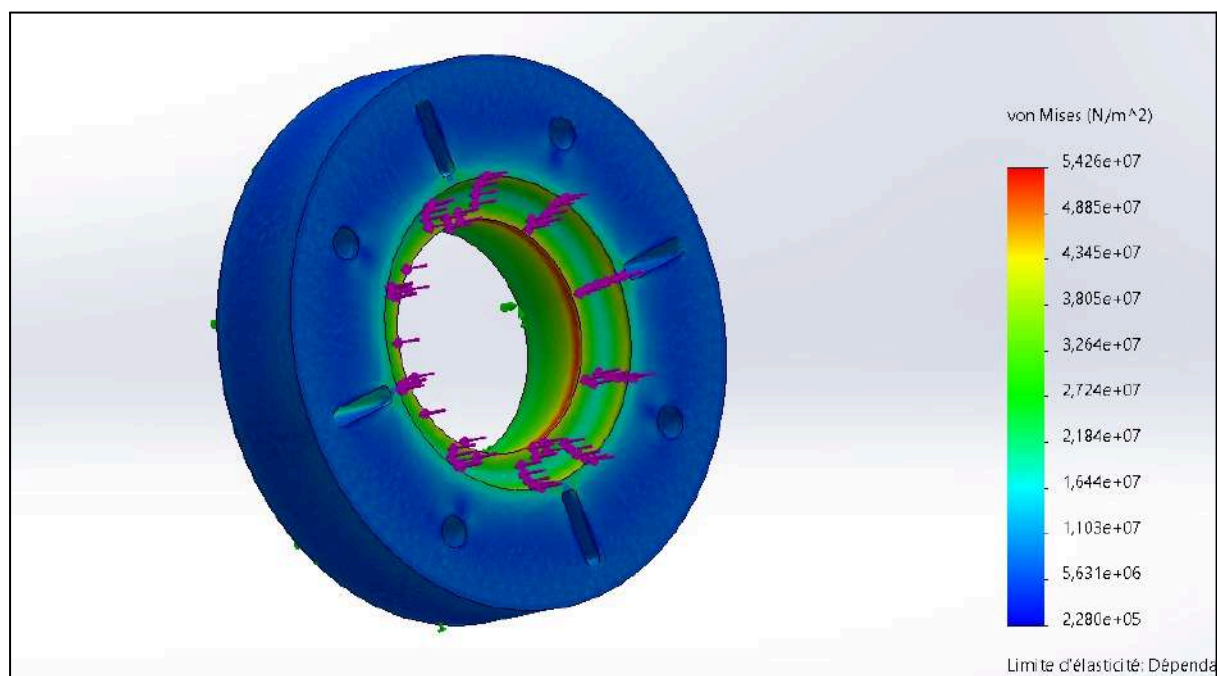
Les objectifs du pôle structure sont :

- Concevoir le montage de la fusée et sa mécanique
- Modéliser les pièces en 3D via CAO sur Catia
- Vérifier la tenue mécanique de chacun des composants de la fusée

La fusée repose structurellement sur le tube en aluminium et sur 4 tiges filetées dans la partie inférieure.

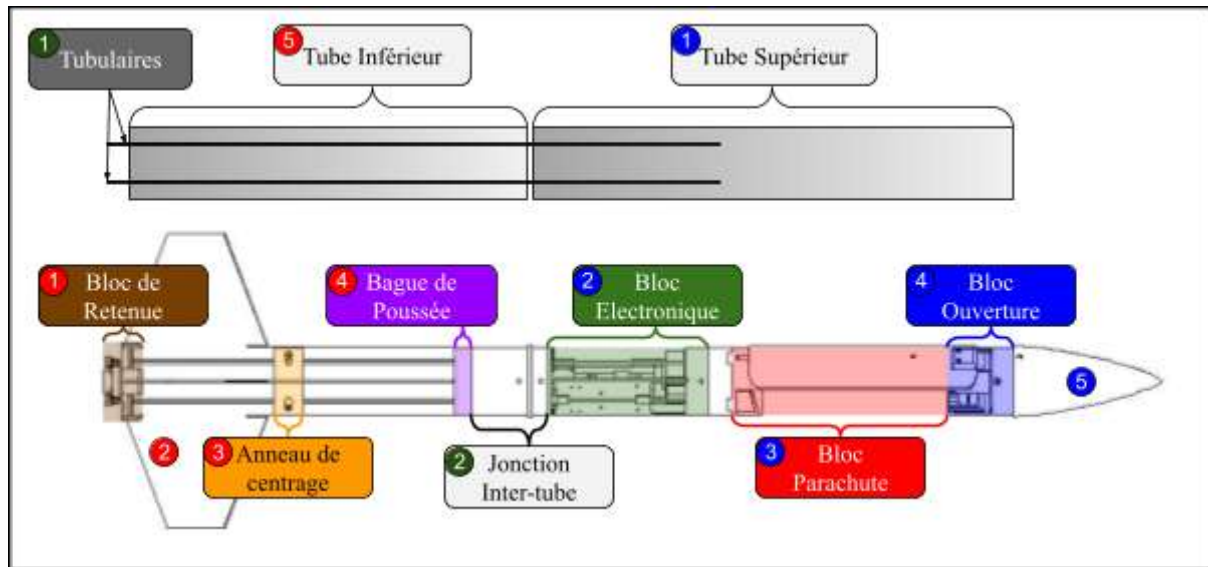
Le tube qui, suite à des contraintes d'usinage, se divise en deux parties, qui sont solidarisiées entre elles par visserie sur une pièce intermédiaire, elle-même vissée sur les tubulaires. Une version avec des boutons presseur a été envisagée mais abandonnée suite à un excès de contrainte par cette méthode.

La pièce maîtresse de la fusée est la bague de poussée qui est l'origine de la fusée. Envisagée en Aluminium pour une reprise par le bas, elle fut finalement modifiée pour une reprise par le haut en ABS.



Simulation compression de la bague de pression (ABS, 50°C, 11 400 N Force)

La conception de pièce se fait sur CATIA V5 et la simulation sur Solidworks via une conversion des fichiers issu de CATIA V5 en ".stp".



Partie Inférieure

(1) Bloc de retenue (ABS) : Quand le moteur est à l'arrêt, il vient s'appuyer sur quatre brides fixées sur les tubulaires. La grande partie en imprimée ABS sert à maintenir fixement les ailerons dans l'axe.

(2) Ailerons (Aluminium) : Les ailerons s'assemblent par assemblage à mi-bois¹ sur le bloc de retenue et le tube inférieur, renforcés par deux attaches vissés sur l'anneau de centrage et par insertion dans la bague de poussée.

(3) Anneau de centrage (ABS) : Creusée en son centre, cette pièce a pour but d'assurer le centrage du moteur par rapport à la fusée. Elle est maintenue en position par des sur-tubulaires en ABS (longitudinalement) et par des groupes palier/écrous vissés sur les tubulaires (radialement et longitudinalement).

(4) Bague de Poussée (ABS) : Le moteur vient s'appuyer lors de la phase de propulsion sur cette dernière, c'est la base de la fusée. Bien que cette dernière soit actuellement en ABS, une version plus résistante en aluminium doit être envisagée.

(5) Tube inférieur (Aluminium) : Usiné à partir d'un tube de 3m de long, de 100mm de diamètre et de 2mm d'épaisseur, ce dernier contient des fentes en fourche en "U" pour accueillir les ailerons ainsi que 4 alésages conçus pour recevoir des vis M5L20 assurant la jonction inter-groupe.

¹ Voir [wikipédia](https://fr.wikipedia.org/wiki/Assemblage_%C3%A0_mi-bois)

Jonction intergroupe

(1) Tubulaires (Acier) : Ces 4 tiges filetées M5 longues de 1 m chacune, assurant la cohésion interne de la fusex, permettent de renforcer la structure et de limiter les alésages de fixation des éléments intérieurs. Les éléments internes de la fusex sont principalement encastrés par vissage à ces tubulaires.

(2) Jonction (ABS) : Cette éléments permet la jonction des deux tubes, par visserie M5L20, aux tubulaires rendant ces trois éléments solidaire les uns des autres.

Groupe électro-parachute

Tube supérieur (Aluminium) : Usiné à partir du même tube de 3m de long, de 100mm de diamètre et de 2mm d'épaisseur, ce dernier contient 4 alésages conçus pour recevoir des vis M5L20 assurant la jonction inter-groupe. 4 alésages supplémentaires pour le maintien de la coiffe et du bloc ouverture. Deux ouvertures sont présente sur les côtés : une pour l'éjection du parachute, une pour l'accès aux commandes électroniques

Bloc électronique (ABS) : Ce module contient le cerveau de la fusée, situé en amont du parachute et en aval du moteur il contient l'électronique nécessaire au bon fonctionnement de la fusex. Une ouverture sur le côté permet d'accéder aux commandes d'ouverture de trappe pour les tests au sol.

Bloc parachute (ABS) : Zone de rangement du parachute, il est situé en aval de l'ancrage du parachute. Une pente en ABS permet de faciliter la sortie du parachute.

Bloc ouverture (ABS) : Ce module contient une servomoteur ainsi que les systèmes mécaniques d'ouverture et d'éjection de la trappe.

Coiffe (ABS) : Conçue par le pôle aérodynamique et modélisée par le pôle mécanique, cette dernière est creuse permettant d'accueillir une partie du parachute. Le parachute devrait sortir par le haut de la fusex.

b) Aérodynamique :

L'objectif du pôle est d'optimiser la forme de la fusée et de réaliser une étude sur l'écoulement de l'air autour de celle-ci.

Ogive :

L'ogive a pour objectif d'améliorer l'aérodynamisme de la fusée

Nous avons choisi une ogive de forme parabolique pour maximiser la place laissée au parachute qui y sera installé en partie.

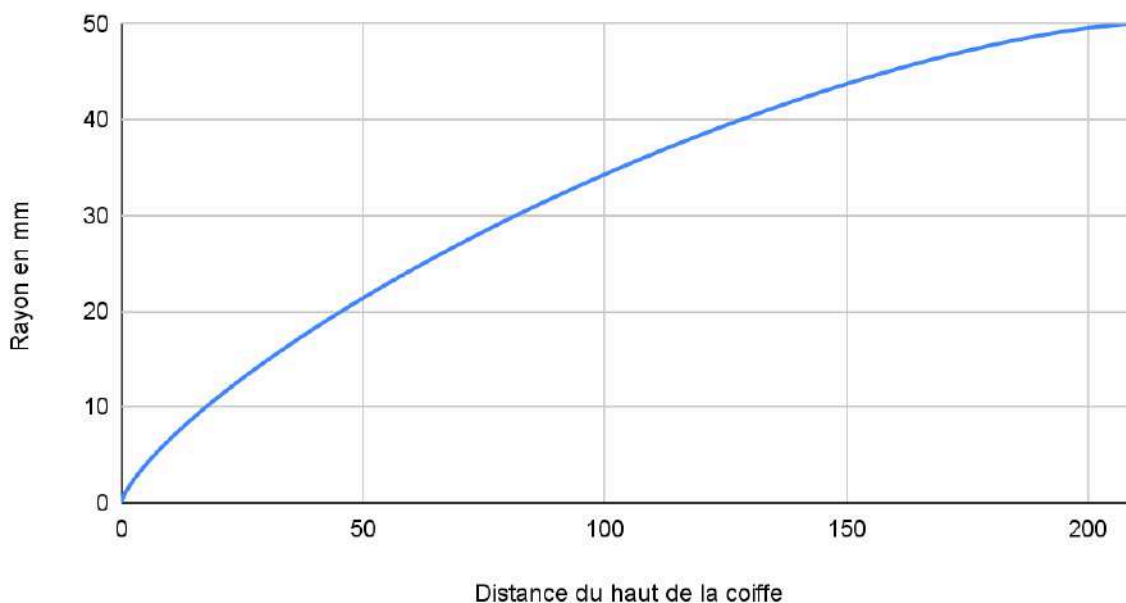
nous ferons notre ogive selon le modèle de Von karman (C=0)

[Nose cone design - Wikipedia](#) Haack series - optimise la réduction de traînée

$$\theta(x) = \arccos\left(1 - \frac{2x}{L}\right)$$
$$y(\theta, C) = \frac{R}{\sqrt{\pi}} \sqrt{\theta - \frac{\sin(2\theta)}{2} + C \sin^3(\theta)}$$

En utilisant ces formules on a pu générer une coiffe ayant le profil extérieur suivant.

Forme de l'ogive de von Karman (L=210mm/R=50mm)



Ailerons

Les ailerons sont les composants qui assureront la stabilité du vol:

Contraintes :

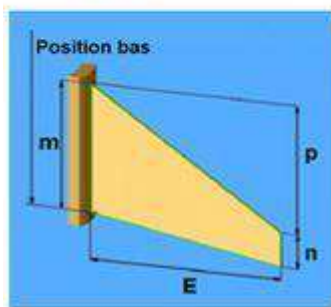
- stabilité
- masse
- traînée
- résistance mécanique
- Complexité de fabrication

Pour assurer une bonne stabilité le centre de poussée (CP) doit être entre 1.5D(diamètre) et 3.0D derrière le centre de gravité (CG), cette distance étant la marge statique,

- Afin de répondre à ses différentes contraintes nous avons opté pour des ailerons trapézoïdaux qui offrent le meilleur compromis entre traînée et résistance mécanique les dimensions sont indiquées dans le schéma ci-dessous :

Emplanture 'm'	230 mm
Saumon 'n'	80 mm
Flèche 'p'	75 mm
Envergure 'E'	150 mm
Epaisseur 'ep'	5 mm
Nombre	4
Position du bas	1460 mm
Diamètre	100 mm

Commentaire libre :



- Les ailerons seront découpés d'une plaque d'aluminium de 5 mm d'épaisseur, afin de résister aux contraintes spécifiées dans le cahier des charges (page 34).
- L'aluminium a été préféré à la fibre de verre car plus facile à travailler avec les outils dont nous disposons, le coût est également moins élevé.
- Les bords seront limés pour améliorer les propriétés aérodynamiques.
- Les ailerons seront placés à 40 mm du bas de la fusée.

Parachute

Ce pôle se charge de la conception et de la fabrication du parachute qui servira à assurer la descente et la récupération de la fusée.

1- Toile

Nous avons opté pour une toile circulaire reliée à un émerillon par huit suspentes. Une corde plus large assurera la liaison entre l'émerillon et le haut de la fusée.

Le matériau choisi pour le parachute est une toile de spi. Une cheminée de 4 cm de diamètre sera découpée au milieu de la toile pour éviter que la fusée ne se balance sous son parachute.

Dimensions de la toile :

A l'aide de la documentation, nous avons calculé la surface nécessaire du parachute. Nous avons utilisé la formule suivante :

Calcul de la surface du parachute $S_p = 2 M g / (r_0 \cdot C_x \cdot V^2)$ en m^2 .

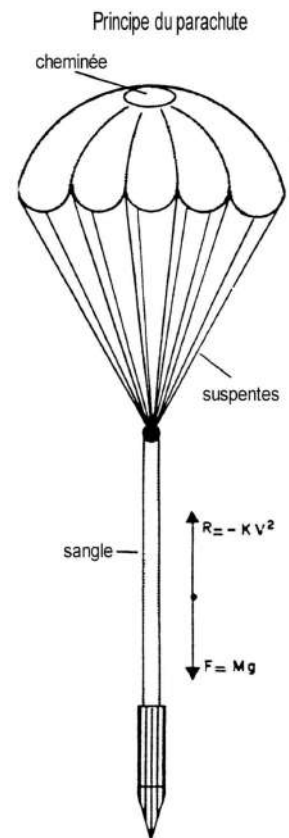
M : masse de l'ensemble de la fusée (en kg)

g : accélération de la pesanteur ($g = 9,81 \text{ m/s}^2$)

r_0 : Masse volumique de l'air ($r_0 = 1,3 \text{ kg/m}^3$)

C_x : Coefficient de pénétration dans l'air ($C_x = 1$)

V : Vitesse de descente (en m/s)



La masse de la fusée prise en compte est de 20 kg et nous voulons atteindre une vitesse de descente de 10 m/s. Cela nous donne une surface de 3 m^2 . Néanmoins, afin que le parachute puisse rentrer dans l'ogive, nous avons décidé de prendre une surface de 2.5 m^2 . Cela affecte peut la vitesse de descente qui est désormais de 11 m/s. La toile étant circulaire, nous avons décidé de la plier radialement cinq fois comme indiqué dans la documentation.

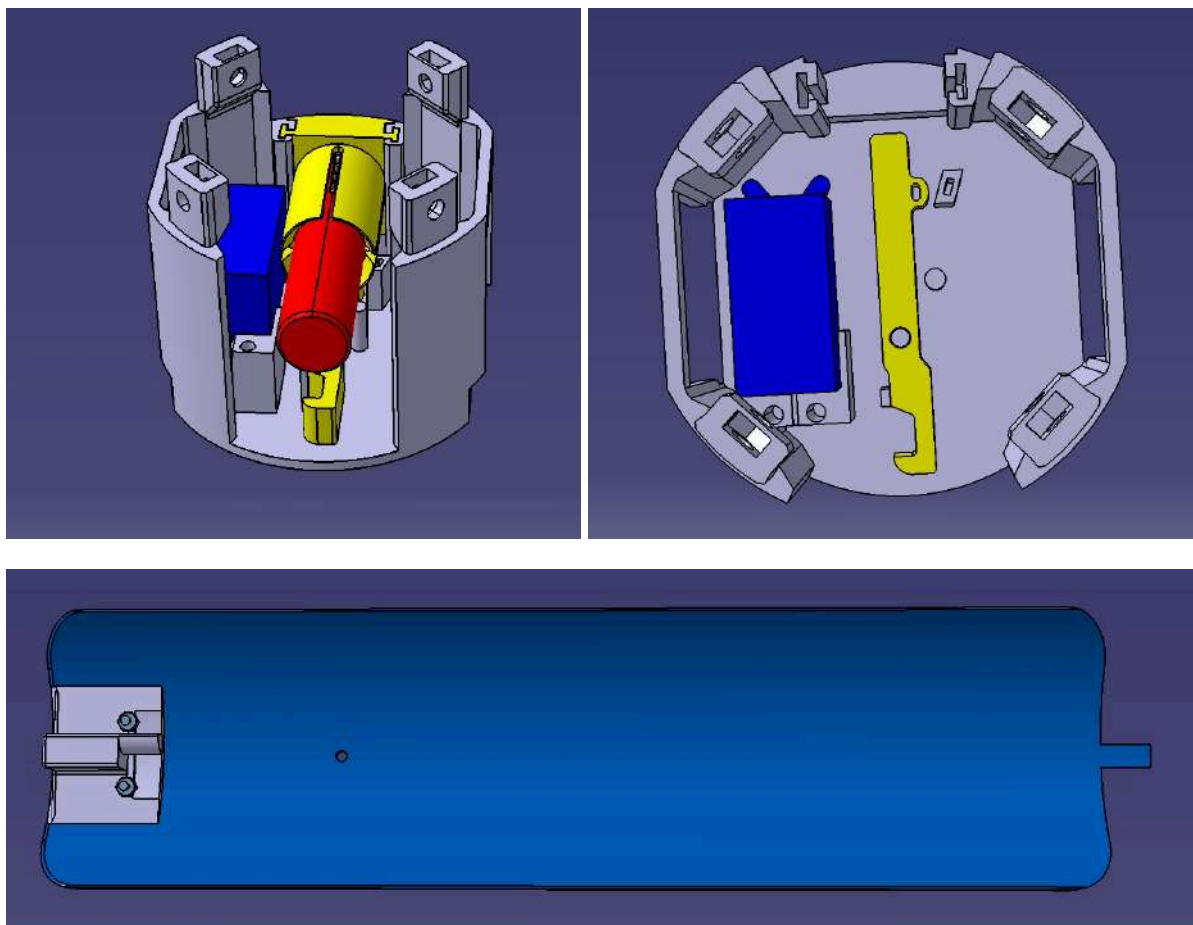
2- Suspentes

Concernant les suspentes, elles doivent résister à une force d'ouverture égale à 5000 N soit 625 N pour chaque suspente (d'après les courbes fournies dans la documentation, cela correspond à la force d'ouverture pour une masse de 20 kg et une vitesse d'ouverture de 50 m/s). Nous avons alors choisi une corde Dyneema de diamètre $\theta = 1 \text{ mm}$ qui a une résistance de 80 daN.

Les suspentes seront nouées à la toile à l'aide d'œillets fixés dans la toile.
La sangle entre l'émerillon et la fusée doit supporter une force de 250 kg. Nous avons choisi une corde de 2 mm de diamètre et résistante à 340 daN.

3- Stockage et éjection du parachute

Le parachute sera stocké dans un compartiment latéral de la fusée, fermé par une trappe. Cette trappe sera maintenue en position par un crochet, lui-même actionné par un servo-moteur commandé via une carte Arduino. Lors du moment opportun, le servo-moteur libérera le crochet, permettant ainsi à un ressort comprimé d'éjecter la trappe vers l'extérieur. Cette ouverture permettra au parachute, fixé à l'intérieur du compartiment, d'être expulsé latéralement hors de la fusée. Le parachute sera relié à la fusée par une corde, identique à celle utilisée pour les suspentes, assurant ainsi une liaison sécurisée une fois déployé.



4)Electronique & Programmation

a) Séquenceur

Le séquenceur joue un rôle crucial dans le lancement de la fusée, notamment en déclenchant le mécanisme de déploiement du parachute. Son fonctionnement repose sur une synchronisation précise pour activer le système au moment optimal. Cette tâche est accomplie en deux phases distinctes, chacune intégrant des aspects matériels et logiciels.

Les contraintes :

- avoir une autonomie d'au moins 1 heure et la mise en marche doit se faire en rampe.
- avoir la puissance nécessaire pour déclencher les mécanismes de récupération.
- séquenceur actif (la fusée a décollé) ou inactif (la fusée attend le décollage)

Fonctionnement du Séquenceur :

Le séquenceur agit en deux étapes. Dans un premier temps, il doit détecter le décollage de la fusée pour initier le processus de séquençage. Le pôle aérodynamisme, à l'aide d'un outil de calcul de trajectographie fourni par planète science, fournit le temps de vol nécessaire avant le déploiement du parachute. Cette information est ensuite utilisée pour activer la seconde étape au moment optimal (l'apogée).

La seconde étape consiste à déclencher le déploiement du parachute. Cela implique une coordination précise entre les composants matériels et logiciels du séquenceur.

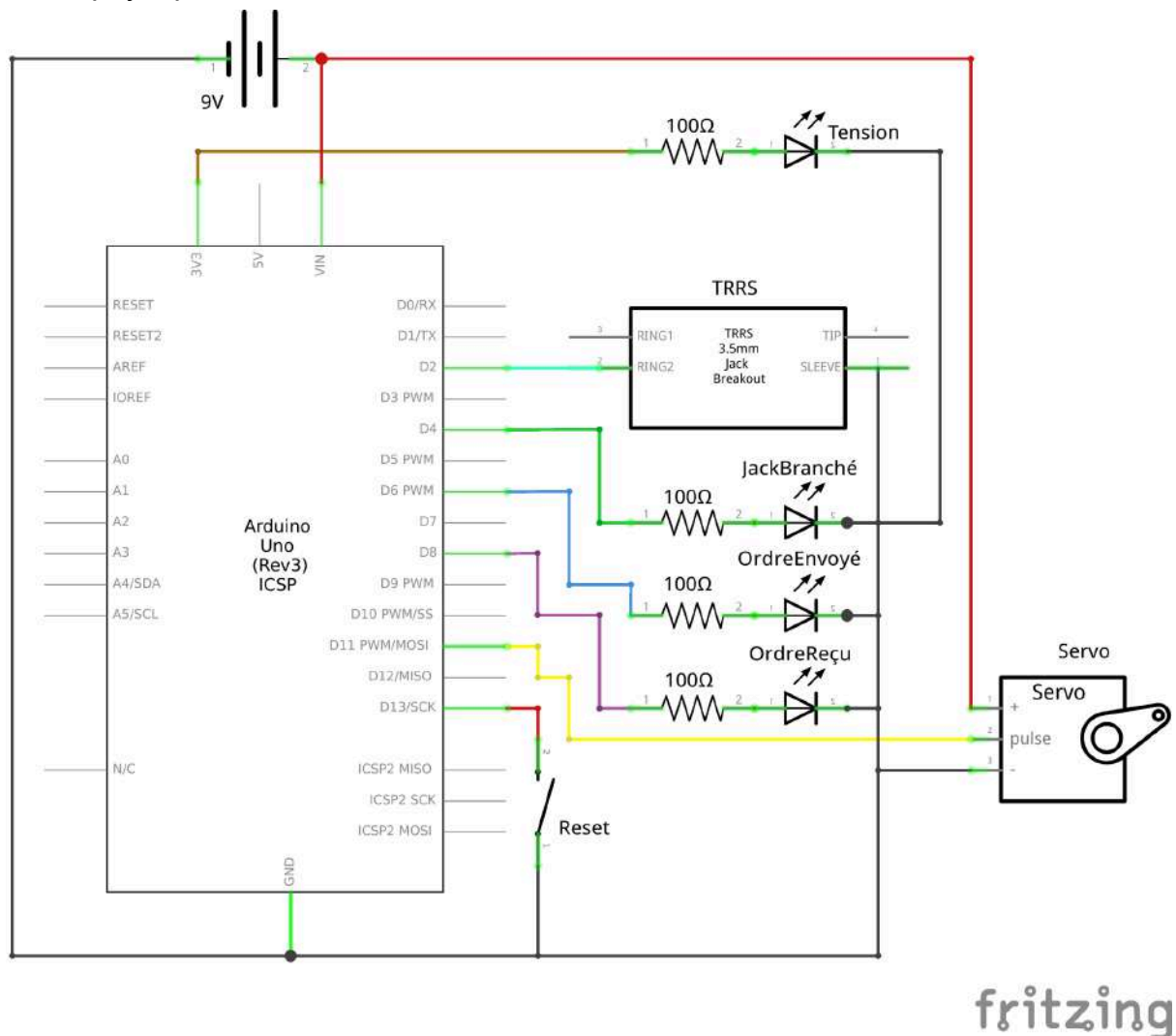
Plus de détails sur son fonctionnement précis sont disponibles dans l'annexe.

Synchronisation avec le Chronomètre :

Le séquenceur utilise un chronomètre comme composant essentiel pour la synchronisation. Ce dispositif permet d'assurer une précision temporelle maximale, garantissant ainsi le déclenchement du parachute à l'instant de l'apogée. Instant calculé par le logiciel de trajectographie en supposant une trajectoire balistique et en prenant en compte les frottement avec l'air pour nous ce temps est de 17.3 secondes.

Schéma Électrique :

Le schéma a été réalisé lors de la phase de conception du séquenceur afin de prévoir les branchements des différents composants ainsi que la confection du circuit physique.



Ces branchements ont été reproduits sur le circuit final comme indiqué dans l'annexe.

Liste des composants :

Composant	Nombre	Prix unitaire	Prix Total
Résistance (100Ω)	4	0.0507	0.0507
Servomoteur	1	10.99	10.99
TRRS (port Jack)	1	1.25	1.25
Arduino Uno	1	24	24
Pile 9V	1	3	3

Connectique pile 9V	1	0.95	0.95
Lot de câbles 3 x 40	1	6.98	6.98

Un critère essentiel dans la sélection des composants a été la capacité à fonctionner sous une faible puissance électrique. Cette exigence découle directement des contraintes de dimensionnement de la fusée, qui nécessite un système d'alimentation compact et portable. Ainsi, nous avons opté pour une alimentation à base de piles, assurant une autonomie suffisante tout en maintenant la légèreté et la portabilité du dispositif.

L'ensemble de ces choix technologiques vise à garantir une intégration optimale des composants tout en respectant les contraintes spécifiques liées au domaine spatial. Cette approche permettra d'obtenir des mesures précises tout en assurant une fiabilité et une efficacité maximales lors de la mission de la fusée.

Pour le choix de la carte, nous avons opté pour une Arduino Uno qui est la carte la plus basique que l'on peut trouver. Elle a pour avantage d'être à la fois assez compacte (74 x 53 x 15 mm) afin de pouvoir entrer dans la fusée et d'être facile à utiliser pour les débutants.

En ce qui concerne les autres composants, les détails de leurs utilisations et application dans le code du séquenceur sont expliqués dans l'annexe.

Enfin, nous avons prévu de faire imprimer le circuit final sur un shield arduino, et d'y souder des connecteurs JST pour faciliter le branchement des composants, mais également le changement de composants. Mais pour de multiples raisons, notamment matérielles, nous avons choisi d'utiliser un proto shield. L'intérêt d'un shield pour la arduino Uno est qu'il peut se brancher directement au-dessus de la carte, plutôt que de brancher les composants à la carte via des câbles et rend alors le système plus robuste face aux secousses. Aussi, le proto shield présente l'avantage de ne pas nécessiter d'impression de circuit et permet un gain de temps considérable ainsi que quelques économies compte tenu de la différence de coût.

b) chaîne de mesure

Cette partie du pôle électronique est chargée de réaliser la chaîne de mesure de la fusée.

La chaîne de mesure est dédiée à l'**obtention des données** de missions, c'est-à-dire la mise en place des différents capteurs et des connectiques permettant l'acquisition des données.

Les objectifs :

Données de mission:

- Mesures de l'accélération et de la rotation
- Mesures de la pression locale de la fusée

Récupération :

- Stockage des données dans la fusée

Les contraintes :

Plusieurs contraintes sont imposées dans un cahier des charges défini par le CNES et Planète Science. Il est donc impératif pour le pôle chaîne de mesures d'adapter le choix des composants à ce cahier des charges. Ces contraintes sont les suivantes :

- Autonomie de l'électronique de la fusée moins trois heures
- Autonomie d'au moins quatre jours pour le stockage. Celles-ci sont par exemple la résistance à l'humidité ou encore aux variations de température
- Mise en place de cavalier et points de tests aux abords des différents composants
- Câblage multibrins, torsadé et blindé
- possibilité de mettre un terme à la prise de données une fois le vol terminé, notamment pour éviter la surcharge de la carte SD du stockage de données.

Le second type de contrainte est celui imposé par le dimensionnement de la fusée. En effet il est important de prévoir des composants qui, une fois mis en place et reliés correctement se limitent à l'espace qui leur aura été attribué par le pôle structure.

Les différentes étapes de travail du pôle chaîne de mesures :

I. Composants

La première étape fut la recherche de composants adaptés, consommant peu. Nous en sommes arrivés au résultat suivant :

Composants	Fréquence max
Gyroscope accéléromètre MPU6050	1KHz
Baromètre - thermomètre MPL3115A2	166Hz (6ms)
Lecteur carte SD GT126	>100Hz
GPS NEO6MV2	5 Hz
Antenne RYLR406	370/s
Raspberry PICO	133MHz (CPU)
Seeed Shield	

La **carte de commande** est une Raspberry PI PICO qui permet de communiquer en I2C, UART et SPI. Elle possède une mémoire et son CPU est d'une fréquence de 133MHz. Elle permet la parallélisation grâce à ces deux cœurs.

Le **baromètre-thermomètre** est un Sparkfun MPL 311512 communiquant via I2C, et pouvant atteindre une fréquence d'échantillonnage d'environ 150 Hz.

Le **gyroscope/accéléromètre** est un MPU-6050, communiquant par I2C, . L'accéléromètre supporte jusqu'à 16g à une fréquence d'échantillonnage de 1Khz.

Le module de **carte SD** est un SD GT126. Il est accompagné de la carte Micro-SD Intenso. Il communique par SPI. Le module a un buffer interne de 512o. Lorsque le buffer est plein, l'écriture sur la carte prend en pratique environ moins de 10 ms. La capacité de stockage maximale est de 8GB. À raison d'un échantillon de 600 toutes les 6ms, la carte sera remplie en 9j 6h13m19s. Nous avons considéré que cela était suffisant pour ne pas avoir à inclure dans le code l'arrêt de l'enregistrement à la fin du vol.

Le [code](#) a été écrit en C avec l'IDE arduino. La raspberry PICO possède deux cœurs, nous avons donc pu paralléliser l'obtention des données et leur écriture.

Par manque de temps, l'antenne RYLR406 n'a pas été incluse dans le circuit. L'antenne du GPS présentait un défaut, nous n'avons jamais réussi à la faire fonctionner. Nous ne l'avons pas incluse dans le circuit.

L'autonomie de la batterie est d'environ 6h, nous l'avons testé en laissant le circuit allumé toute une nuit

II. réalisation du schéma de câblage

Nous avons réalisé le schéma du montage avec fritzing.

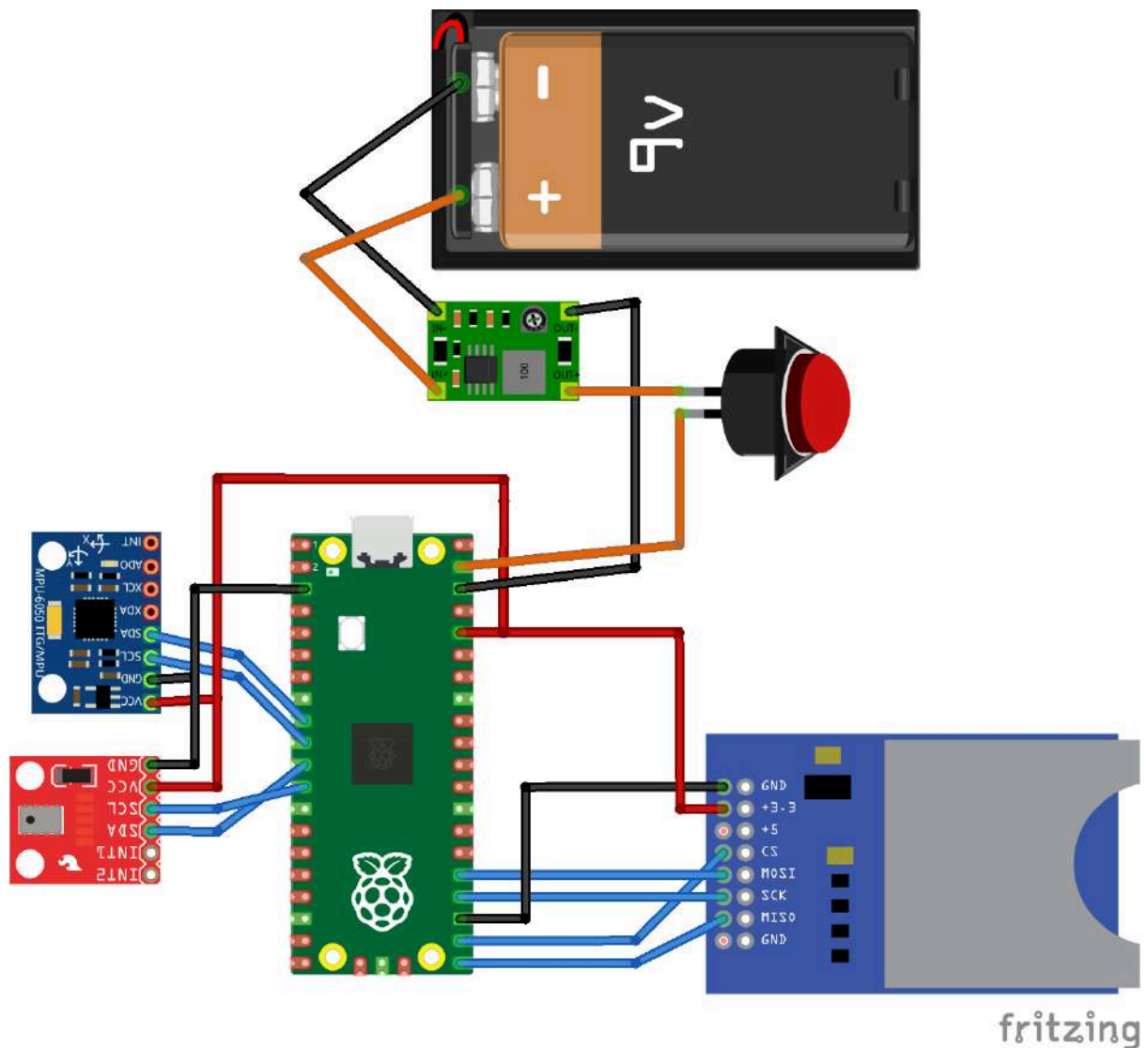


Schéma circuit de mesure

code couleur :

- En rouge l'alimentation de chaque composant
- En orange l'alimentation de la carte par la pile
- En noir la masse
- En bleu les connections SPI et I2C

III. Traitement des données collectées

Les données obtenues sont celles de l'accéléromètre sur trois axes, celles du gyromètre sur trois axes et la pression du baromètre. Pour convertir la pression en altitude on utilise la formule suivante:

$$z = - \ln(\text{press} / \text{press}_0) * R * T_0 / M * g$$

avec R, M, G constantes

T0 = 23°C température ambiante le jour du lancement

p0 = 969hP pression au sol le jour du lancement (tel que mesurée par le baromètre ce jour là)

Le gyromètre ne donne que la vitesse de rotation. Pour obtenir la rotation initiale, on utilise l'accélération de la pesanteur mesurée à la rampe de lancement avant lancement.

Le repère dans lequel les données seront analysés sera le repère avec comme origine la rampe de lancement, comme hauteur celui de la pesanteur ($e_z = -g/|g|$). Le vecteur accélération est multiplié par la matrice de rotation mise à jour avec l'intégration de la vitesse angulaire du gyromètre pour revenir du repère capteur au repère initial.

Pour pouvoir obtenir la position de la fusée, on intègre avec un filtre de Kalman les données d'accélération corrigées avec l'altitude relative obtenue avec le baromètre, tel que:

$$x_k = F_k x_{k-1} + B_k u_k + w$$

et

$$y_k = H_k x_k + v$$

avec

x_k = [px, py, pz, vx, vy, vz] vecteur contenant position et vitesse

F_k matrice fonction de Δt intégrant x_k-1

u_k = [ax, ay, az] vecteur de l'accélération calculée avec l'accéléromètre

B_k matrice intégrant u_k

w bruit d'évolution

et avec

y_k = [pz] altitude locale calculée avec le baromètre

H = [0,0,1,0,0,0] matrice conservant l'altitude relative de x_k

v bruit de mesure

Ainsi, nous pouvons obtenir la position de la fusée et tracer la trajectoire avec les données de l'accéléromètre-gyromètre et du baromètre.

5)Coûts

a) prévisionnel

Nous avons estimé les coûts à plusieurs reprises en réalisant des modifications au fur et à mesure que nous avançons dans la conception. Nous avons finalement déterminé que nous aurions besoin d'environ 1500 EUR pour réaliser le projet, le détail des coûts est présent dans le fichier excel joint à ce document.

Pour déterminer les coûts nous avons cherché principalement sur internet le prix de différents composants dont nous aurions besoin. Pour que les pièces doivent être usinées, nous nous sommes directement renseignées auprès des techniciens (interne à l'insa) auprès desquels nous allons réaliser nos pièces.

b) Financement

Les fonds nécessaires à la réalisation du projet ont été rassemblés grâce à deux sources différentes.

un financement d'une hauteur de 600 EUR de la part du Bureau des étudiants (association à laquelle le club est rattaché)

Un financement de 930 EUR de la part du FSDIE (Fonds de solidarité et de développement des initiatives étudiantes).

Le financement total s'élève à 1530 EUR et suffirait à financer l'entièreté du projet.

c) Réel

Au final le coût total de la construction de la fusée s'est élevé à **716,97 €** le prix est bien plus faible que ce que nous avons prédit, notamment parce que l'usinage nous a été offert par le centre d'usinage de notre école.

la répartition des coûts était comme suit:

- mécanique : 237,42 €
- électronique : 328.05 €
- parachute : 51.50 €
- administratif : 100 €

6) CSpace

Trois membres de notre club ont pu participer au C'Space 2025. Arrivés sur le site le samedi en fin de journée, nous avons immédiatement commencé l'assemblage de la fusée et procédé aux derniers ajustements sur la chaîne de mesure, notamment pour corriger quelques dysfonctionnements liés aux capteurs.

Le dimanche, en début d'après-midi, nous avons présenté la fusée aux différents contrôles techniques. Si plusieurs étapes ont été validées sans problème, un blocage est survenu concernant la bague de poussée en ABS : les contrôleurs ont exprimé des doutes quant à sa capacité à résister à la puissance du moteur. Il a donc été décidé de la remplacer par une bague en métal.

Grâce à la solidarité entre équipes, nous avons pu récupérer une feuille d'aluminium auprès d'un groupe et emprunter une Dremel à un autre. Nous avons passé le reste de la journée de dimanche, ainsi qu'une bonne partie du lundi matin, à découper, limer et ajuster cette nouvelle bague. Celle-ci devait à la fois permettre le passage du moteur et assurer la reprise de poussée par le bas.

Nous sommes repassés en contrôle le lundi après-midi. Toutes les vérifications restantes ont été validées, ce qui nous a permis de réaliser le vol simulé et d'obtenir la validation finale, juste à minuit.

Le lancement initialement prévu pour le mardi après-midi a été avancé de manière inattendue. Alors que nous finalisons les derniers ajustements du code, remplaçons les piles et ajoutons quelques éléments de personnalisation à la fusée, nous avons été contactés par l'équipe de lancement : ils nous demandaient si nous pouvions être prêts pour un tir à 11h30. Nous avons accepté et nous nous sommes rendus rapidement sur zone.

Malheureusement, un contretemps technique a empêché le tir immédiat. Nous sommes finalement retournés sur le pas de tir dans l'après-midi, vers 15h30. Un premier problème s'est présenté lors de la mise en rampe : le diamètre de la nouvelle bague de poussée était trop grand, empêchant l'insertion correcte de la fusée. Après concertation, nous avons décidé de changer de rampe et d'essayer un nouveau diamètre (105 mm), ce qui a permis une mise en place correcte.

La démarche pré-vol a alors pu être réalisée : vérifications finales, quelques photos, puis passage de relais à l'équipe pyrotechnique pour l'installation du moteur. Lors de cette étape, une anomalie de non-continuité a été détectée sur la ligne d'allumage. Après une investigation approfondie, l'équipe a pu corriger le problème.

Le lancement a finalement eu lieu à **17h51**. Le vol s'est déroulé sans incident, avec un déploiement correct du parachute. La fusée est descendue lentement et, par

chance, a atterri à proximité d'un chemin, ce qui a grandement facilité sa récupération. Nous avons pu la retrouver et la ramener rapidement et sans difficulté.

7) Résultats

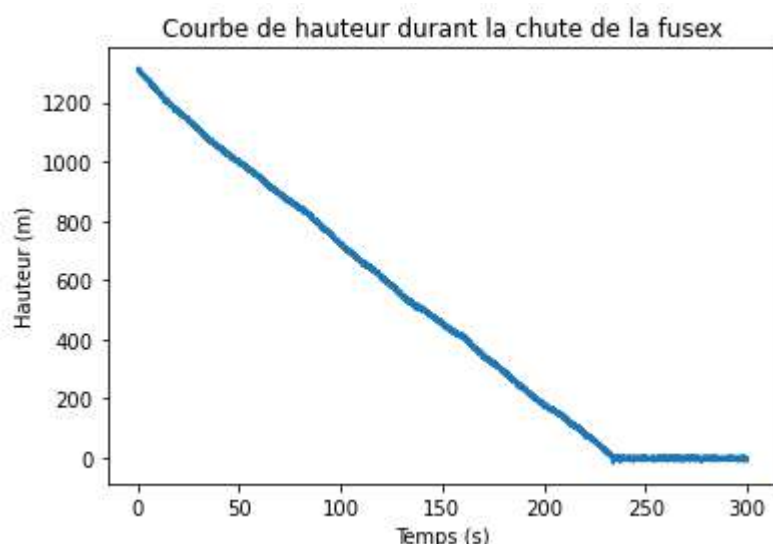
a) Vol

Le vol est nominal et s'est très bien passé, Le moteur un pro 54 green a poussée la fusée hors de la cage, celle-ci a réalisé une légère rotation en sortie de cage (probablement dû au fait que la cage était un peu trop large et a moins bien guidé la fusée en sortie), elle a néanmoins continué sa montée avec le parachute qui s'est déployé juste après l'apogée, celui-ci étant particulièrement sur-dimensionné la descente a été particulièrement lente, (5mn30) mais par chance le vent a joué en notre faveur et la fusée a été retrouvée sur le bord d'une route sans grande difficulté.

b) Expérience

Malheureusement, l'expérience ne s'est pas déroulée comme prévu. Nous ne disposons que d'une partie des données, correspondant à la phase de descente. Les données enregistrées avant cela ont été perdues, probablement à la suite d'un redémarrage de la carte provoqué par un court-circuit ou un mouvement de fil au moment de l'ouverture du parachute, ce qui a entraîné la réécriture des premières mesures.

Nous avons tenté de récupérer les données effacées, mais sans succès : elles semblent définitivement perdues.



Nous pouvons toutefois retracer l'évolution de l'altitude pendant la descente et savons que la fusée a atteint une altitude d'au moins 1650 mètres.

Recommandations

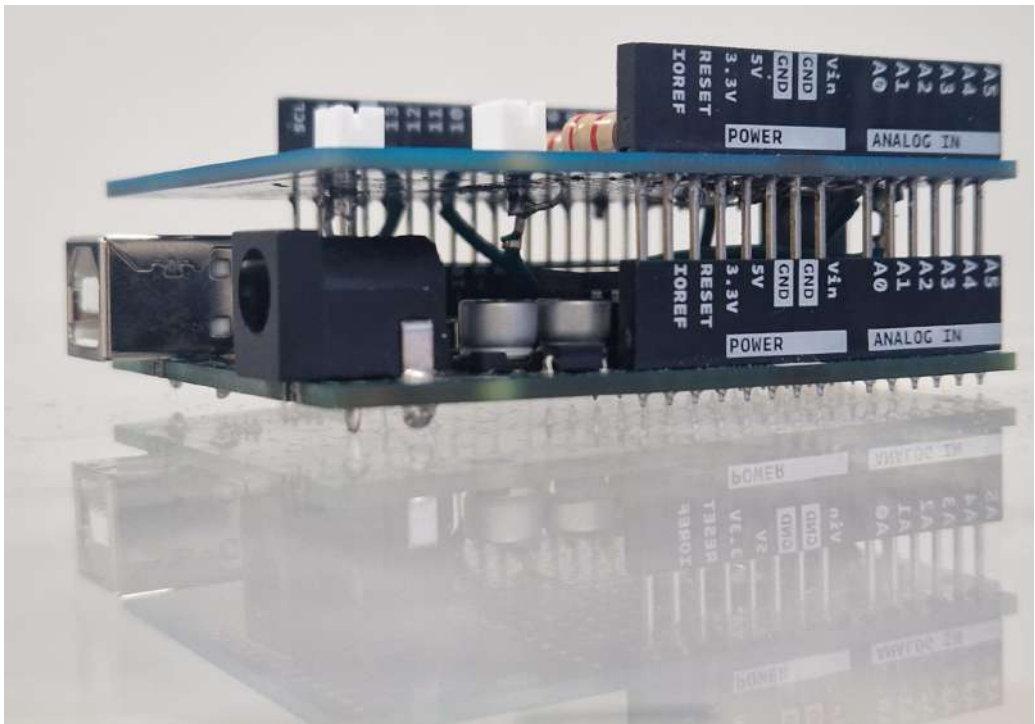
- Les acquisitions de accéléromètre-gyromètre et du baromètre ne permettent pas de réaliser une trajectographie 3D. Il faut ajouter un moyen de corriger la déviation horizontale de l'accéléromètre, comme la position obtenue par un récepteur GPS ou par des antennes au sol.

- Il ne faut pas inclure dans le code d'acquisition des mesures la suppression de tout ou partie des données déjà enregistrées sur la carte SD.

8) Conclusion

Le projet **Fusée expérimentale Bellatrix**, porté par le club **INS'ASTRO** de l'INSA Rouen Normandie, a abouti à un lancement réussi lors du **C'Space 2025**, concrétisant plusieurs mois de conception, fabrication et assemblage. La fusée a atteint une altitude d'au moins **1650 m**, avec un déploiement nominal du parachute et une récupération intacte à proximité du site. Ce succès a été obtenu malgré des imprévus majeurs, tels que le remplacement en urgence de la bague de poussée en ABS par une version en aluminium, des ajustements de dernière minute sur la structure et l'électronique, ainsi qu'une perte partielle des données de vol due à un probable redémarrage de la carte au moment de l'ouverture du parachute. Cette première itération valide la conception générale — structure aluminium, parachute surdimensionné, chaîne de mesure à base de Raspberry Pi Pico et capteurs inertiels — tout en révélant des points d'amélioration : fiabilisation des connexions électriques, sécurisation du stockage des données, intégration d'un GPS pour compléter la trajectographie et optimisation du système de récupération pour réduire la durée de descente. Réutilisable et modulable, **Bellatrix** constitue désormais une plateforme d'essai robuste pour les générations futures, offrant un socle technique solide pour affiner les performances et élargir le champ des expériences embarquées, et ceux en essayant dès l'année prochaine d'intégrer un tube de pitot dans la fusée, un gps et une télémessure.

Séquenceur



Le mode d'emploi

Abstract

Ce guide a pour objectif de permettre à des personnes ne connaissant pas cette partie du projet de pouvoir la reprendre. Il explique en détails le montage et le fonctionnement du séquenceur.

Contents

Matériel et montage	2
Présentation du matériel	2
Alimentation	2
Leds	2
Factory Reset	3
Servo	3
TRRS	3
Montage	4
Code	5
Constantes	5
Variables	5
Fonctions utilitaires	6
sms	6
armer	6
fermerServo	6
ouvrirServo	6
parachute	7
lancerProcess	7
stopperProcess	8
autoReset	8
actualiserProcess	8
Fonctions principales	9
setup	9
loop	10
Fonctionnement du séquenceur	11
Déroulement normal	11
Cas particuliers	11

Présentation du matériel

Dans cette partie, on ne va considérer que les éléments en dehors du shield et de la carte arduino.

Alimentation



Simplement le support de pile 9V avec un bouton ON/OFF et un connecteur JST¹ pour le branchement sur le shield.

Leds

Les leds sont les témoins de l'état du processus. Le code couleur pour les leds est le suivant :

- Rouge : Témoin d'alimentation
- Bleue : Témoin de la prise jack
- Verte : Timer en cours - processus lancé
- Jaune : Ouverture de la trape - fin du processus

En cas de changement de led, il faut faire attention au sens de branchement : l'anode (broche la plus longue) est le côté positif et la cathode, négatif.



¹Ce composant étant récurrent, il ne sera mentionné que cette fois

Factory Reset



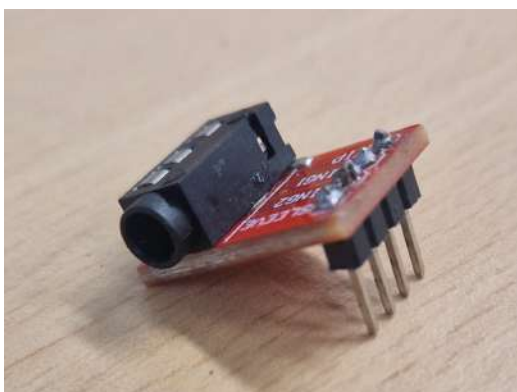
Ce bouton fait une remise à 0 logicielle.
C'est tout.

Servo

Le servo moteur sert au déclenchement l'ouverture de la trappe du parachute, qui on l'espère, permettra à la fusée d'atterrir sans dommages. Idéalement, il devrait être alimenté en parallèle de la carte et non via la carte pour une meilleure fiabilité. Également, il ne dispose pas de capteur directement exploitables, il est donc impossible de connaître sa position réelle, et donc, si sa position ne peut être atteinte, le séquenceur ne pourra en avoir connaissance. Il faut donc s'assurer qu'un espace suffisant est disponible pour ne pas gêner ses rotations.



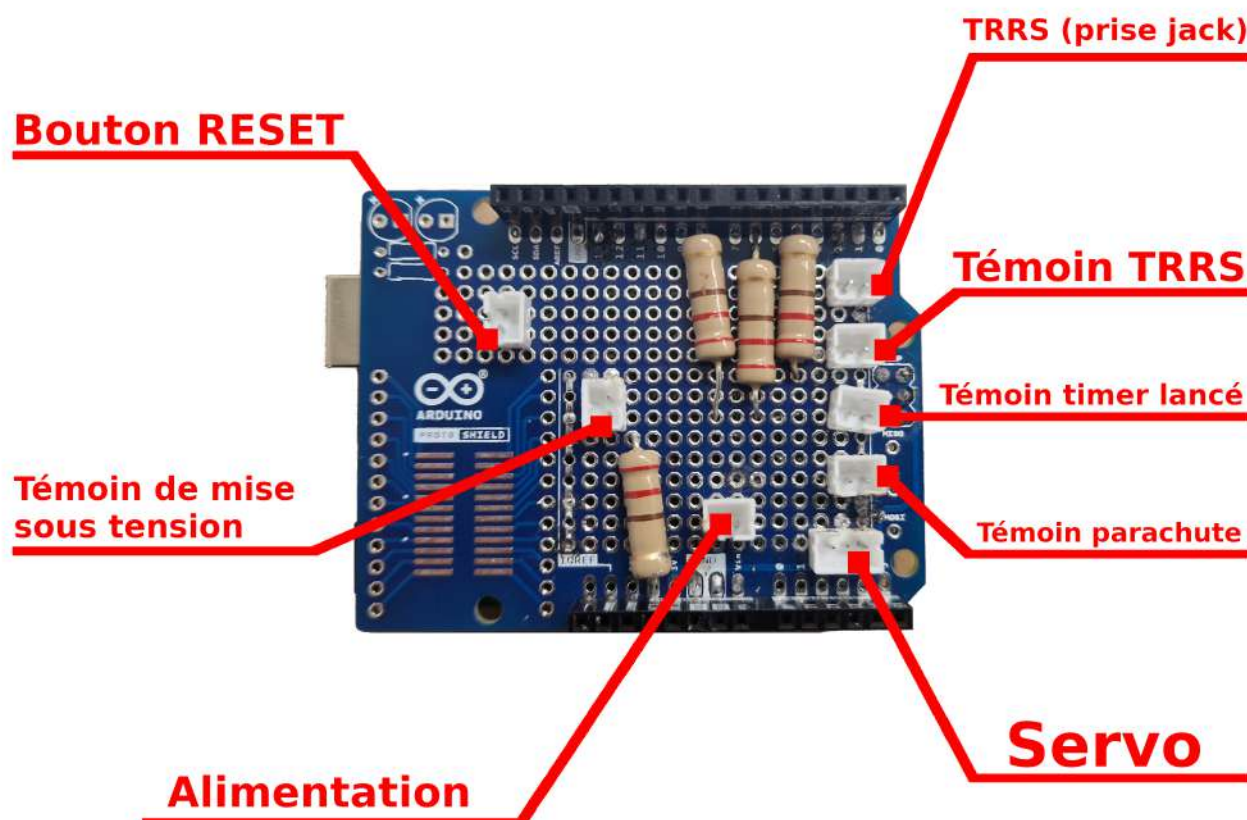
TRRS



Le TRRS est un port Jack utilisé pour détecter si un câble y est branché. C'est ce qui démarrera le séquenceur au décollage. La détection se fait via une mesure de tension à la masse, pour une meilleure fiabilité et pour éviter tout incident, il faut donc que la prise soit bien branchée dans le TRRS et idéalement que le câble soit soudé au bout pour prévenir d'un sectionnement.

Montage

Voici le schéma de montage :



Chaque composant est connecté à une prise JST. Sur le schéma, les éléments pointent sur les connecteurs JST correspondant sur le proto shield. Il y a 8 composants à brancher, 7 sont des connecteurs à deux pins, le restant en possède trois². Il suffit de brancher chaque composant à sa prise comme indiqué sur le schéma.

Tous les connecteurs JST des témoins sont soudés dans le même sens sur le shield. Si on les regarde avec la fente devant, le pin de gauche correspond au V+ et le pin de droite au ground. **Si il faut changer une led**, l'anode (côté positif) correspond à la broche la plus longue. Ne surtout pas la souder à l'envers.

Le TRRS suit la même logique que pour les témoins.

Le bouton fait simplement une connexion lorsqu'il est utilisé. Il n'a donc pas de sens de branchement.

Pour le connecteur du servo, il est soudé sur la ligne de ground avec les connecteurs des témoins. Il faut donc avoir le ground du même côté (vers la droite dans le sens du schéma). **MAIS**, comme les soudures ont été réalisées au mieux selon le proto shield, le pin V+ est celui à gauche (toujours par rapport au schéma). Le pin du signal est donc celui du milieu. Généralement, le pin du signal est jaune sur les servos.

Enfin, le connecteur de l'alimentation est soudé en accord avec les pins en face. Dans le sens du schéma, on a le ground à gauche et le V+ à droite.

Dernier détail : Le circuit n'inclus pas la régulation de la tension du servo. Le régulateur devra être intégré entre le connecteur JST du servo moteur sur le proto shield et le servo lui-même.

²Il s'agit du connecteur du servo moteur

Constantes

Constante	Type	Rôle	Valeur
JACK	int	Pin du TRRS	2
SERVO	int	Pin du servo (signal)	11
JACK_LED	int	Pin du témoin du TRRS	4
ORDRE_E_LED	int	Pin du témoin du processus en cours	6
ORDRE_R_LED	int	Pin du témoin de fin de processus	8
BOUTON	int	Pin du bouton de réinitialisation logicielle	13
TEMPS_VOL	unsigned int	Temps de vol en secondes	17
TEMPS_RESET	unsigned int	Délais pour le reset automatique (secondes)	8
SERV_FERM	int	Angle du servo fermant la trappe	90
SERV_OUVR	int	Angle du servo ouvrant la trappe	0

Variables

Variable	Type	Rôle
PRET	bool	Trape fermée, processus prêt à être lancé
INIT	bool	Système initialisé (prise jack branchée au moins une fois)
PROCESS	bool	État du processus (en cours ou non)
AUTORST	bool	Reset automatique du système lancé
t0	unsigned long	Point temporel de départ du timer
servo	Servo ³	Instance du servo moteur

³Ne pas oublier d'inclure `Servo.h`

Fonctions utilitaires

sms

```
unsigned long sms(unsigned int s) {  
    return s * 1000;  
}
```

Cette fonction sert uniquement à convertir un nombre de secondes en milli secondes.

armer

```
void armer() {  
    Serial.println("Armement");  
    INIT = false;  
    PRET = false;  
    PROCESS = false;  
    digitalWrite(ORDRE_R_LED, LOW);  
    digitalWrite(ORDRE_E_LED, LOW);  
    digitalWrite(JACK_LED, LOW);  
    fermerServo();  
}
```

Prépare le système en enclanchant la fermeture de la trappe.

fermerServo

```
void fermerServo() {  
    servo.write(SRV_FERM);  
    PRET = true;  
    Serial.println("Servo fermé");  
}
```

Fermeture effective de la trappe.

Le système est prêt, mais non initialisé.

ouvrirServo

```
void ouvrirServo() {  
    servo.write(SRV_OUVR);  
    PRET = false;  
    Serial.println("Servo ouvert");  
}
```

Ouverture de la trappe déployant le parachute.

Le système n'est plus prêt à être lancé.

parachute

```
void parachute() {  
    if (INIT && PRET) {  
        Serial.println("Daploiment du parachute");  
        digitalWrite(ORDRE_R_LED, HIGH);  
        ouvrirServo();  
        stopperProcess(); // le processus est terminé  
    }  
}
```

Enclenche le processus de déploiement du parachute.

lancerProcess

```
void lancerProcess() {  
    if (PRET && INIT && !PROCESS) {  
        Serial.println("Démarrage du timer");  
        PROCESS = true;  
        t0 = millis();  
        digitalWrite(ORDRE_E_LED, HIGH);  
    }  
    else  
        Serial.println("Démarrage du processus impossible");  
}
```

Lancement du processus (ie, le timer).

Il ne peut être lancé que si

- Le système est prêt
- Le système est initialisé
- Le processus n'est pas en cours

Le point de départ t_0 sera enregistré à ce moment.

stopperProcess

```
void stopperProcess() {
    if (PROCESS) {
        PROCESS = false;
        Serial.println("Processus stoppé");
        digitalWrite(ORDRE_E_LED, LOW);
        autoReset();
    }
    else
        Serial.println("Interruption du processus impossible");
}
```

Arrêt du processus et lancement de la réinitialisation automatique.

autoReset

```
void autoReset() {
    if (!PROCESS && !PRET) {
        AUTORST = true;
        Serial.println("Reset automatique dans" + String(TEMPS_RESET) + "secondes");
        t0 = millis();
    }
    else
        Serial.println("Impossible de lancer le reset automatique");
}
```

Lancement du timer de réinitialisation automatique.

Le point t_0 est réutilisé pour ce timer.

actualiserProcess

```
void actualiserProcess() {
    if (PROCESS && PRET && millis() - t0 > sms(TEMPS_VOL))
        parachute();
    else if (AUTORST && !PRET && millis() - t0 > sms(TEMPS_RESET))
        armer();
}
```

Actualisation de l'état. SI le processus est en cours, mesure $t - t_0$. Si la mesure obtenue est supérieure à la durée de vol TEMPS_VOL ⁴, alors le déploiement du parachute est commandé. Si le reset automatique est lancé, la mesure de temps est comparée à TEMPS_RESET et si le délais est atteint, l'armement du système est effectué.

⁴Voir la table des constantes

Fonctions principales

setup

```
void setup() {  
    Serial.begin(9600);  
    pinMode(SERVO, OUTPUT);  
    pinMode(JACK_LED, OUTPUT);  
    pinMode(ORDRE_R_LED, OUTPUT);  
    pinMode(ORDRE_E_LED, OUTPUT);  
    pinMode(JACK, INPUT_PULLUP);  
    pinMode(BOUTON, INPUT_PULLUP);  
  
    digitalWrite(JACK_LED, LOW);  
    digitalWrite(ORDRE_E_LED, LOW);  
    digitalWrite(ORDRE_R_LED, LOW);  
  
    servo.attach(SERVO);  
    armer();  
}
```

Mise en marche logicielle. Déclaration des pins (les pins indiqués par `INPUT_PULLUP` utilisent une résistance interne. Il s'agit des pins déclanchant des actions (boutons et TRRS)).

Le `Serial` est branché à 9600 bits par secondes pour les logs de debug.

Par défaut, tout les témoins d'états logiciel sont éteints au démarrage. Seul le témoin physique de mise sous tension est allumé (c'est normal, il n'est pas contrôlé par un pin).

Au démarrage, le système s'arme automatiquement (fermeture de la trappe). Il est possible de commenter cette ligne si pour une raison ou une autre, il ne faut pas fermer la trappe lors du démarrage du séquenceur. Dans ce cas, il faudra se référer à la dernière partie de ce tutoriel pour découvrir la manoeuvre à effectuer pour fermer la trappe du parachute, et préparer le système.

ATTENTION : Si un obstacle empêche le bon fonctionnement du servo moteur et sa rotation. Cela peut endommager le servo et augmenter inutilement la consommation du séquenceur. De plus, le système n'aura pas connaissance de ce problème et considérera l'armement comme réussi.

loop

```

void loop() {
    if (digitalRead(BOUTON) == LOW) {
        if (!PROCESS) {
            armer();
            delay(500);
        }
    }

    bool jack = (digitalRead(JACK) == LOW);
    digitalWrite(JACK_LED, (jack) ? HIGH : LOW);
    if (jack) {
        INIT = true; // jack branché au moins une fois => système initialisé
        if (PROCESS) // stopper un éventuel processus en cours
            stopperProcess();
        else if (AUTORST) // reset manuel
            armer();
        delay(750); // pour assurer une bonne initialisation
    }
    else {
        if (INIT && PRET && !PROCESS) // conditions nécessaires pour lancer le timer
            lancerProcess();
    }

    if (PROCESS || AUTORST) {
        actualiserProcess();
    }
}

```

La boucle principale du séquenceur.

Dans un premier temps, la première partie vérifie l'état du bouton pour enclancher l'armement de la trappe si activé. Le principe du bouton est qu'il connecte son pin digital au ground, lorsque l'on pèse dessus, la connexion est établie et il est possible de lire la valeur LOW sur le pin. Si il n'est pas actif, la valeur ne sera pas LOW en raison de la résistance de pull up.

La seconde partie exploite ce même principe pour le TRRS. Si la prise jack est branchée, le pin du TRRS sera connecté au ground. Dans ce cas, le système est initialisé.

La logique de cette boucle est décrite dans la partie suivante du guide.

Fonctionnement du séquenceur

Déroulement normal

Dans le cadre *prévu* du fonctionnement de ce séquenceur, les étapes de fonctionnement sont les suivantes :

- Mettre sous tension (bouton ON/OFF)
- Armement (automatique ou manuel avec le bouton reset) (système prêt)
- Initialisation (branchement de la prise jack)
- Lancement du processus (timer) (débranchement du jack)
- Fin du processus (déploiement du parachute)
- La réinitialisation automatique est lancée

Dans le contexte du fonctionnement de la fusée, le câble jack ne sera pas débranché manuellement.

Cas particuliers

- Appuyer sur le bouton entre la fin du processus et le reset automatique avance la réinitialisation et réarme le système.
- Appuyer sur le bouton alors que le processus est lancé n'a aucun effet
- Pour stopper le processus prématurément, il suffit simplement de rebrancher la prise jack, le système reviendra à son état prêt et initialisé, mais le timer ne sera plus en cours.