Association Eurêka Plus
12 Avenue Jean Béranger
78160 Marly le Roi
Tél/fax : 01.39.58.87.92
www.eurekaplus.org

Association loi 1901, agréée « jeunesse&éducation populaire »

Fusée expérimentale SPIDER

Antoine de Maleprade - Dan Baczynski

Sommaire

L’association Eurêka +

L'association EUREKA +, historique et situation actuelle

Eurêka Plus propose aux jeunes de 12 à 25 ans de s'initier aux sciences et techniques par le biais de la réalisation de projets aérospatiaux. Tous les projets sont réalisés en équipe à l’initiative des jeunes.

Des mini-fusées, fusées expérimentales et des ballons stratosphériques sont ainsi réalisés chaque année dans nos locaux Yvelinois. Ce sont des projets novateurs, à la pointe de la technologie qui permettent d'insuffler à la nouvelle génération, une passion pour l'espace, la recherche et la culture spatiale.

Depuis 24 ans, les animateurs bénévoles d'Eurêka Plus sont au service de la science. Chaque année, l’association organise la Fête de l'Espace dans les Yvelines et participe à de nombreuses manifestations telles que les campagnes de lancements régionales, le rassemblement Espace Étudiant…

Présentation de L'équipe

Antoine de Maleprade (chef de projet), 16 ans, qui a déjà construit plusieurs fusées par le passé mais c'est sa première au sein du club eurêka+. Il est en classe de première S au lycée.

Dan Baczynski, 15 ans, qui va ici construire sa première fusée expérimentale. Il est aussi en classe de première S au lycée.

Chacun des deux membres vont participer à toutes les étapes de fabrication de la fusée avec Antoine qui est un peu plus qualifié pour l'électronique et Dan pour les plans et la mécanique.

Toute la conception et la fabrication de cette Fusée vont s'effectuer dans le cadre du club Eurêka+, sous la responsabilité de Thibault Raboisson.

Présentation et mise en œuvre du projet

I.Définition et but de l'expérience

Nous avons choisi de construire une fusée expérimentale qui avait pour but, de décoller, pour atteindre environ 1000m d'altitude puis d'ouvrir un ralentisseur permettant une chute maitrisée du corps de la fusée et enfin de mettre en œuvre un guidage du corps de la fusée pendant la chute pour atteindre une cible définie avant le vol.

II.Moyens mis en œuvre pour réaliser notre expérience

1.Propulsion

• Comment propulser notre fusée à 1000m d'altitude ?
➡ Nous allons utiliser le propulseur Pro-54 fourni par planète-sciences qui est largement suffisant si la fusée ne dépasse pas une dizaine de kilos.

2.Ralentir et Diriger la fusée

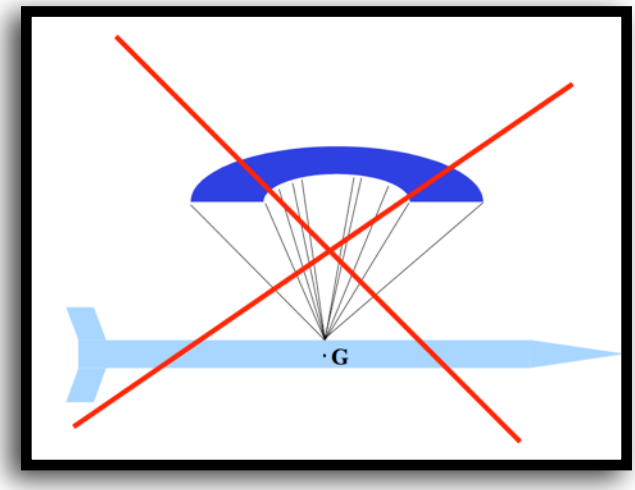
• Comment guider et ralentir la chute du corps de la Fusée ?
➡ Avec une aile de parapente et plusieurs capteurs ainsi que des actuateurs permettant de corriger la trajectoire de la fusée.

• Comment Déployer cette aile de parapente ?
➡ On a choisi de ralentir la fusée et de la stabiliser au moyen d'un autre ralentisseur et de n'ouvrir que plus tard l'aile de parapente.

a.Stabiliser la fusée au moyen d'un autre ralentisseur

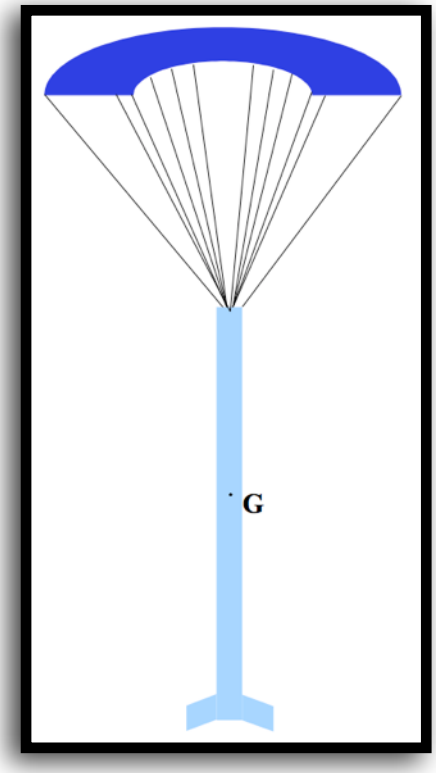
Pour que la fusée puisse se stabiliser de manière à ouvrir une autre aile par la suite, il nous faut définir l'emplacement du point d'attache du parapente.

• Horizontalement, au dessus du centre de gravité :



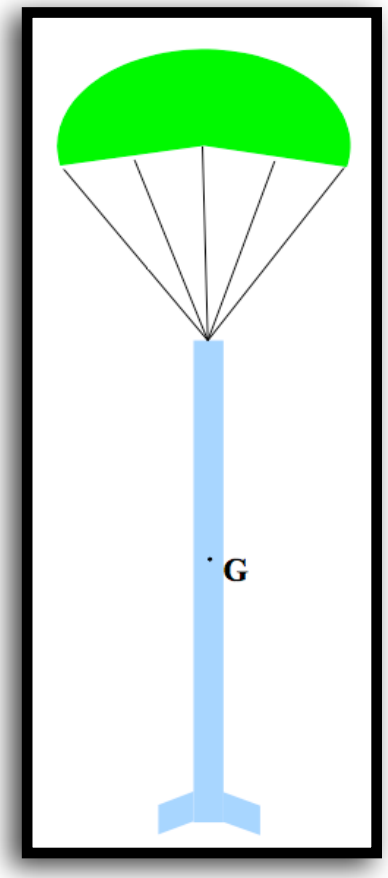
Il est difficile de trouver le centre de gravité exact qui peut varier très facilement avec de petites modifications. De plus la stabilité est limitée car il faut se placer plus haut que le CDG pour réduire le balancement. (Sinon la marge statique est trop faible)

• **A l'extrémité supérieur de la fusée :**



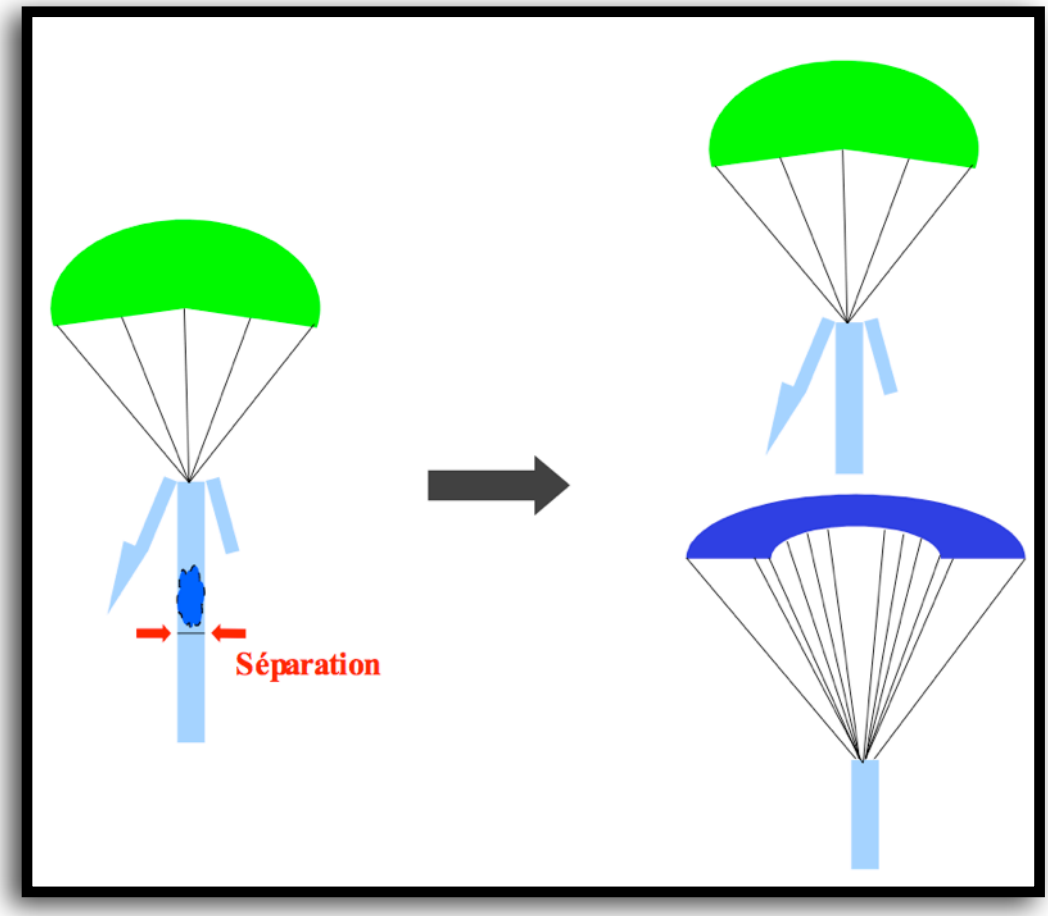
Ici, le balancement en sera diminué car le point d'attache est bien au dessus du centre de gravité mais plus éloigné. (ici la marge statique plus élevée) L'idéal serait de pouvoir écarter le point d'attache droit et le point d'attache gauche, pour que la fusée ait moins tendance à tourner sur elle même.

Pour la stabilisation avant l'ouverture de l'aile, on va mettre un parachute fixé dans cette même configuration :



On a donc choisi le type d'ouverture « peau de banane », c'est-à-dire que l'ogive va s'ouvrir en deux pour libérer le parachute.

b.Ouvrir l'aile de parapente



Description et objectif du projet « Spider »

Le vol de la fusée se décompose en 4 phases :
Phase 1 : Culmination
Phase 2 : Stabilisation de la fusée à l'aide d'un parachute (1000m)
Phase 3 : Déploiement d'une aile (800m)
Phase 4 : Pilotage automatique de l'aile afin de rejoindre une cible grâce aux capteurs embarqués

c.Pilotage et Algorithme de vol

Pour piloter la fusée, nous allons actionner les freins du parapente (suspentes arrières) ce qui va nous permettre de faire pivoter l'ensemble fusée-aile pour modifier sa trajectoire. Pour cela nous allons utiliser un servo-treuil (servomoteur multi-tours avec un couple élevé).

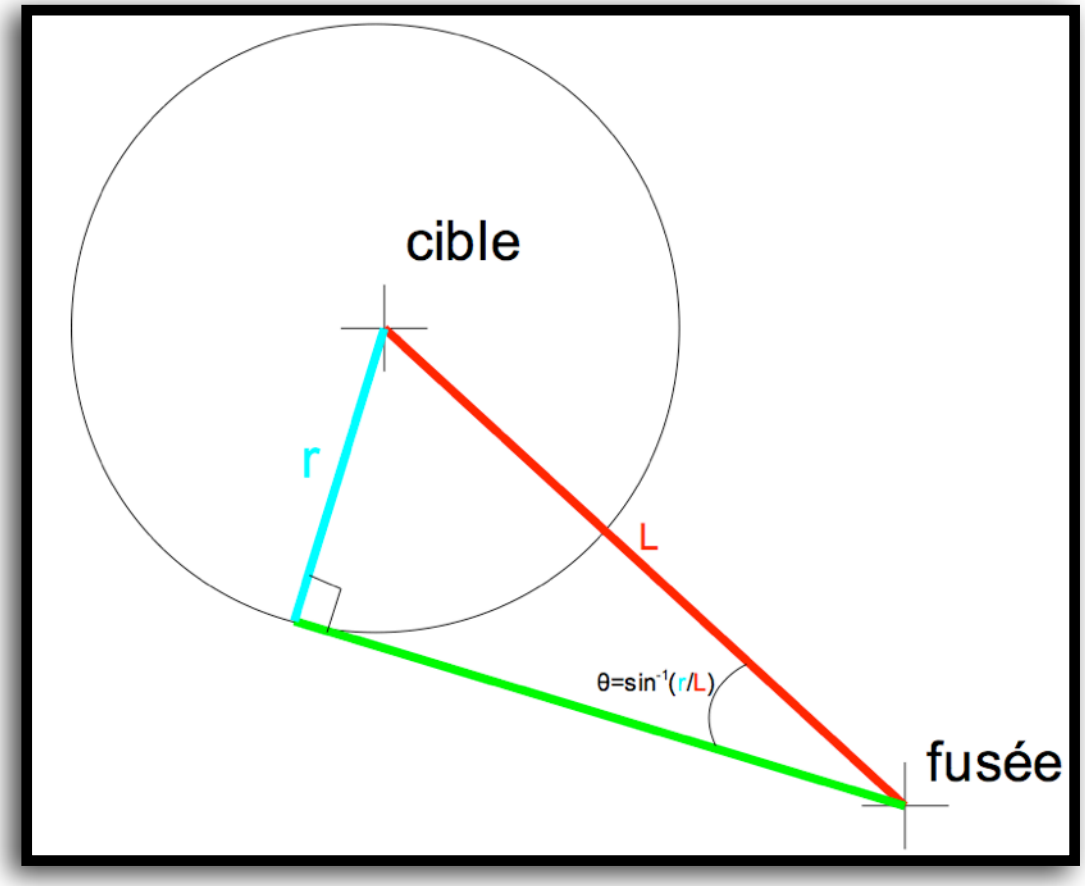
Ensuite, pour connaître les corrections à faire sur la trajectoire, nous allons utiliser un GPS et une boussole électronique. On pourra, à partir de ces deux modules :
- calculer le cap à suivre avec les coordonnées de la cible (GPS)
- connaître notre propre route (boussole)

Cependant nous pouvons rencontrer d'autres problèmes pour notre pilotage :

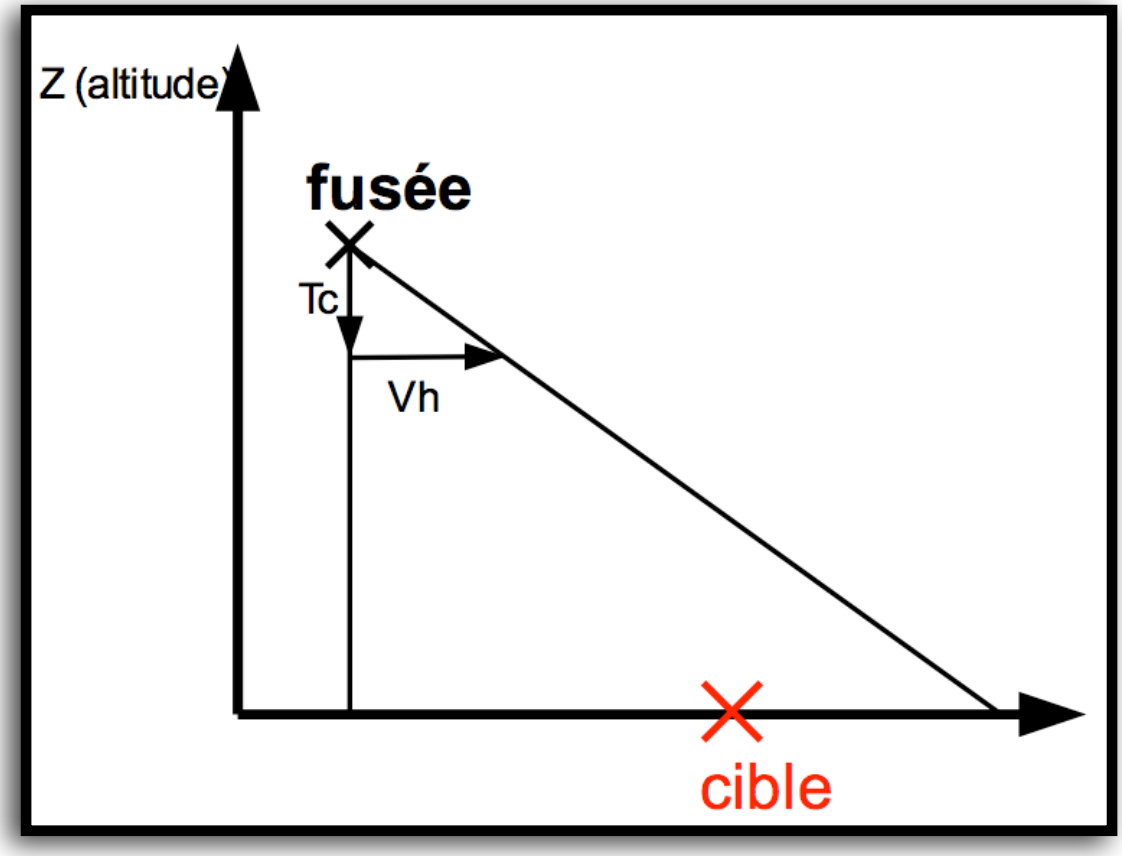
- **Une fois que la fusée a atteint sa cible (théoriquement) mais se trouve plus haut que celle-ci (en altitude), elle risque d'effectuer une trajectoire anormale et risquerait de ne pas atterrir sur la cible à la fin du vol**
- **Le vent risque de dévier la fusée et de lui faire suivre une route incorrecte**
- **Si on définit une position centrale, une petite différence dans la taille des suspentes ferait pivoter la fusée ou modifierait la trajectoire.**

Notre premier problème va se résoudre avec un capteur de pression (altitude) et en calculant une trajectoire selon le taux de chute et la vitesse de déplacement horizontal pour arriver sur la cible, face au vent si possible.

Pour cela, on ajoute un angle à la route à suivre pour que la fusée se dirige vers un cercle qui a pour centre la cible et lorsque l'altitude devient suffisamment basse, on modifie notre route pour se diriger vers la cible.

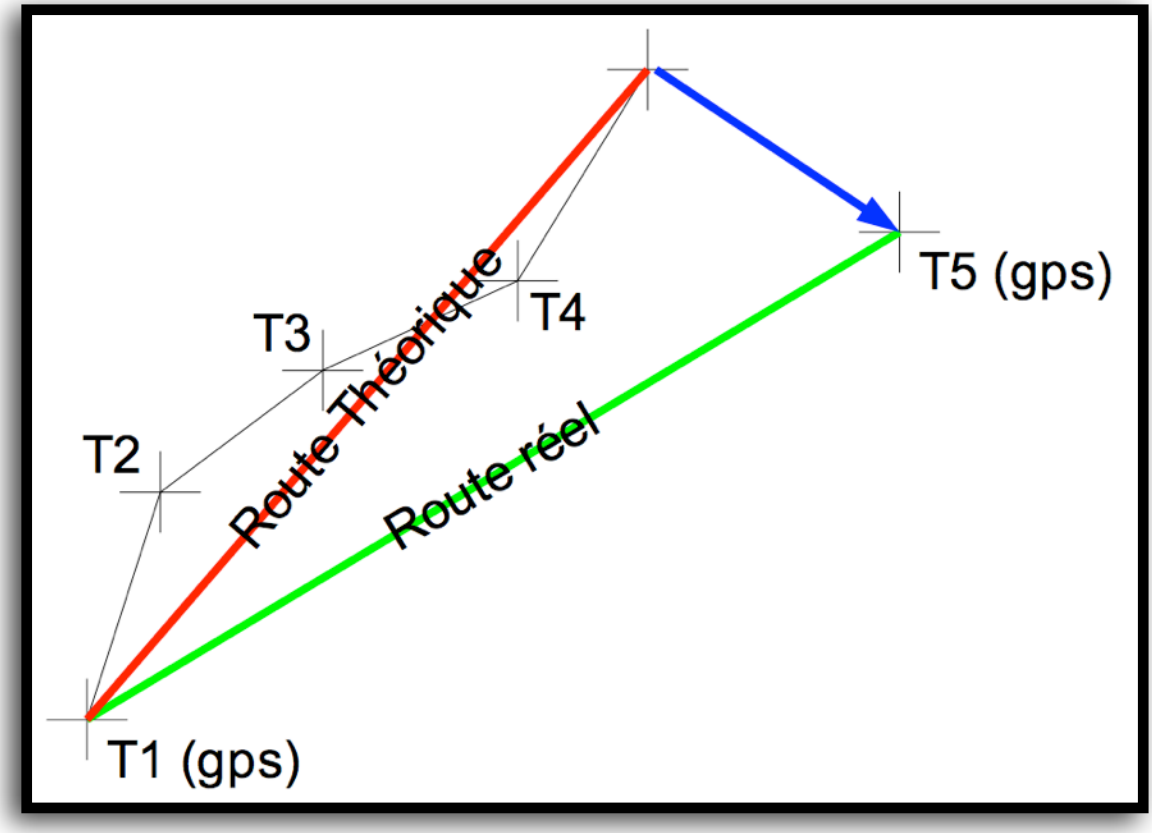


L est la distance entre la fusée et la cible et r est la rayon du cercle. On calcul θ qui est la correction à ajouter à la route à suivre. On pourra faire varier r pour se trouver face au vent lorsqu'on se dirigera vers la cible. (il existera un r minimal selon le parapente, servo ...). On pourra aussi vérifier par les calculs que la vitesse du vent est suffisamment basse pour que l'on puisse avancer face au vent.



On a Tc le taux de chute et Vh la vitesse de déplacement horizontal. Soit x et z les différences de distances au sol et en altitude entre la fusée et la cible, l'altitude sera suffisamment basse à partir du moment ou : Tc/z < Vh/x (théorème de Thalès). On pourra aussi anticiper le moment ou la fusée va se diriger vers la cible selon le temps mis par celle-ci pour effectuer un changement de cap.

Les deux problèmes suivants vont se résoudre en calculant la déviation entre une trajectoire théorique et une trajectoire réelle. La trajectoire réelle se définit par le chemin suivi entre deux positions GPS à un temps différent. La trajectoire théorique va se calculer en fonction du cap suivi (boussole) et de la vitesse de déplacement théorique (constante définie avant le vol).



Ici, on a la route théorique en rouge calculée en fonction des différents caps suivis : on définit un intervalle de temps Δt qui peut être le temps de rafraichissement de la boussole (minimum) ensuite on calcul en fonction de la vitesse de déplacement les coordonnées du point suivant (d = Vd/Δt). On fait la différence avec la route réelle pour trouver le vecteur de correction (en bleue) à utiliser dans nos calculs.

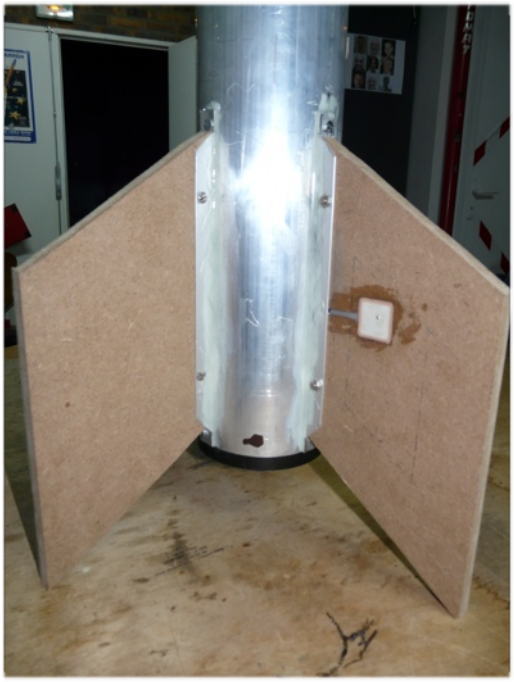
(Voir le Programme du microcontrôleur en Annexes pour plus de précisions)

1.L'électronique

a.Le GPS

Nous avons utilisé un GPS pour guider la fusée car cela nous permet de connaître la position de la fusée par rapport à la cible. Les coordonnées GPS de la cible seront déjà connues par l'électronique de la fusée, on va ensuite calculer les coordonnées polaires de la fusée par rapport à la cible.
Les problèmes rencontrés avec le GPS sont :
Le décrochage du signal (surtout au décollage)
Le placement de l'antenne qui doit être à ciel découvert.

Pour résoudre ces deux problèmes, nous avons utilisé un GPS de haute qualité permettant de garder le signal jusqu'à 6G d'accélération et de le récupérer (si perte) en 0,5sec.



Notre fusée a été construite avec un tube en aluminium ce qui exclut la possibilité de mettre l'antenne dans le tube. Pour pallier à ce problème, nous avons intégré l'antenne dans des aillerons en bois.

b.Le capteur de pression

Nous avons utilisé un capteur de pression pour connaître l'altitude de la fusée par rapport au sol (à la cible) l'altitude du GPS n'étant pas assez fiable.
Cela nous permettra d'établir une différence dans l'algorithme lorsque la fusée est au même niveau que la cible ou bien à quelques dizaines de mètres au dessus.
L'étalonnage du capteur est présente dans l'annexe.

c.Le compas électronique

Nous avons aussi inclus un compas électronique pour connaître la vitesse angulaire lorsque la fusée tourne ainsi que la direction dans laquelle l'aile est dirigée car le GPS peut fournir de mauvaises valeurs à cause d'un vent trop fort.
Le compas utilisé (CMP03) nous fournit un signal PWM suivant la direction du Nord magnétique terrestre en degrés absolus.

d.Un actuateurs : Le servo-treuil HS704-HB

Le servomoteur HS704-HB peut effectuer 5,5 tours ce qui nous permet un plus grand champ d'action sur le pilotage de l'aile et permet aussi d'entreprendre un virage engagé lorsque la fusée risque de sortir du gabarit.

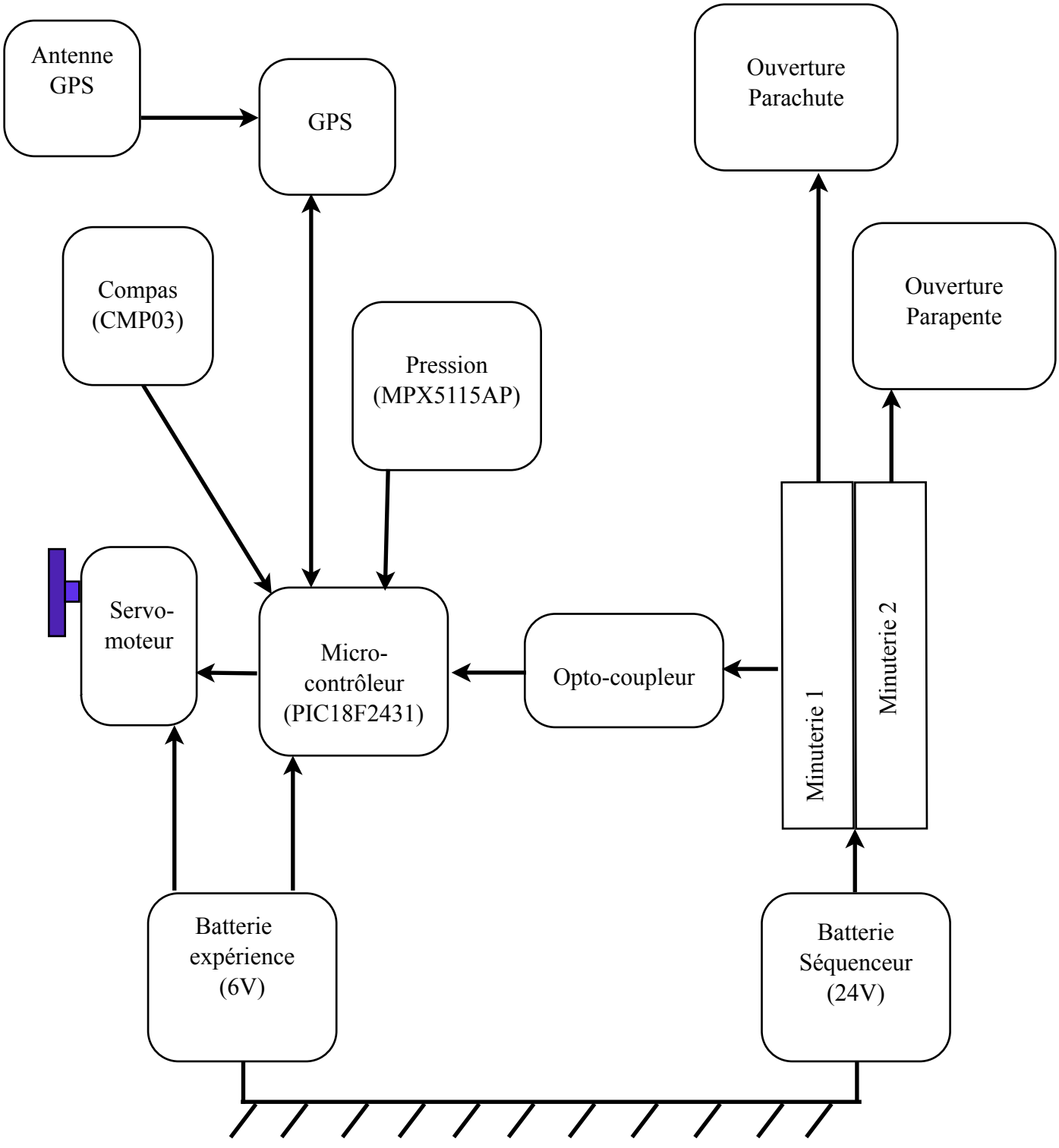
e.Le microcontrôleur : un PIC18F2431

Le pic18F2431 permet la communication entre tous ces périphériques et d'effectuer les calculs nécessaires au pilotage de la fusée.

f.Le séquenceur : NE556

Le NE556 nous permet d'avoir deux minuteries activant séparément le parachute puis l'aile de parapente et est en même temps assez fiable et facile à mettre en œuvre car le but de notre fusée n'est pas de tester un système de récupération.

g.Organisation



Alimentation et signalisation

La fusée va comporter deux blocs d'alimentation séparés pour le séquenceur et l'expérience.

- **Le séquenceur doit actionner deux électro-aimants d'une tension de 24V. Le NE556 (double temporisateur), vu précédemment, doit s'alimenter entre 4,5 et 16V. Nous avons donc intégré une batterie de 24V avec un régulateur 12V pour la carte du séquenceur.**

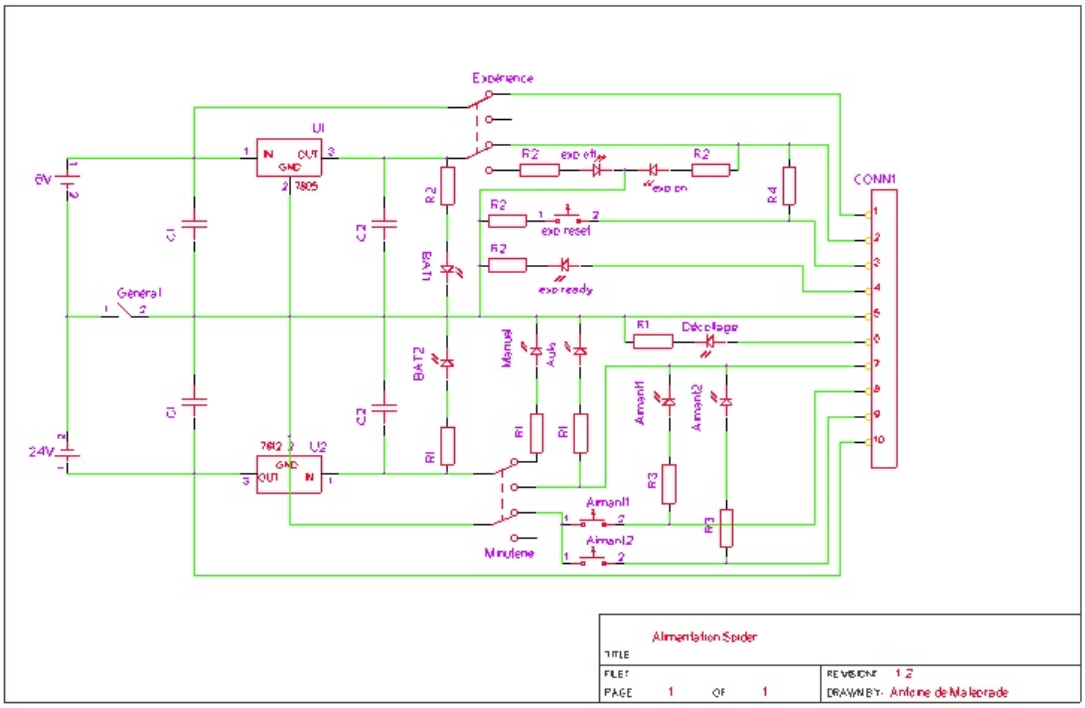
- **Les circuits de l'expérience vont comporter un servomoteur de 6V et une carte comprenant un microcontrôleur PIC alimenté en 5V : Nous allons utiliser une seconde batterie de 6V avec un régulateur 5V pour la carte du microcontrôleur.**

Interrupteurs :

- **Alimentation générale**
- **Expérience On/Off**
- **Séquenceur Manuel(off)/Automatique(on)**
- **Deux boutons poussoirs pour actions manuelle sur les électro-aimants (désactivés quand le séquenceur est en automatique).**

Signalisation :

- **deux del de contrôles batteries 1 et 2 (vertes)**
- **deux del séquenceur : manuel (rouge) et Automatique/Initialisé (verte)**
- **deux del expérience : off (rouge) on (verte)**
- **une del expérience « prêt » (verte)**
- **une del GPS « prêt » (bleue)**
- **une del détection du décollage (orange)**
- **deux del indiquant le déclenchement des électro-aimants (en automatique et en manuel) (rouge)**

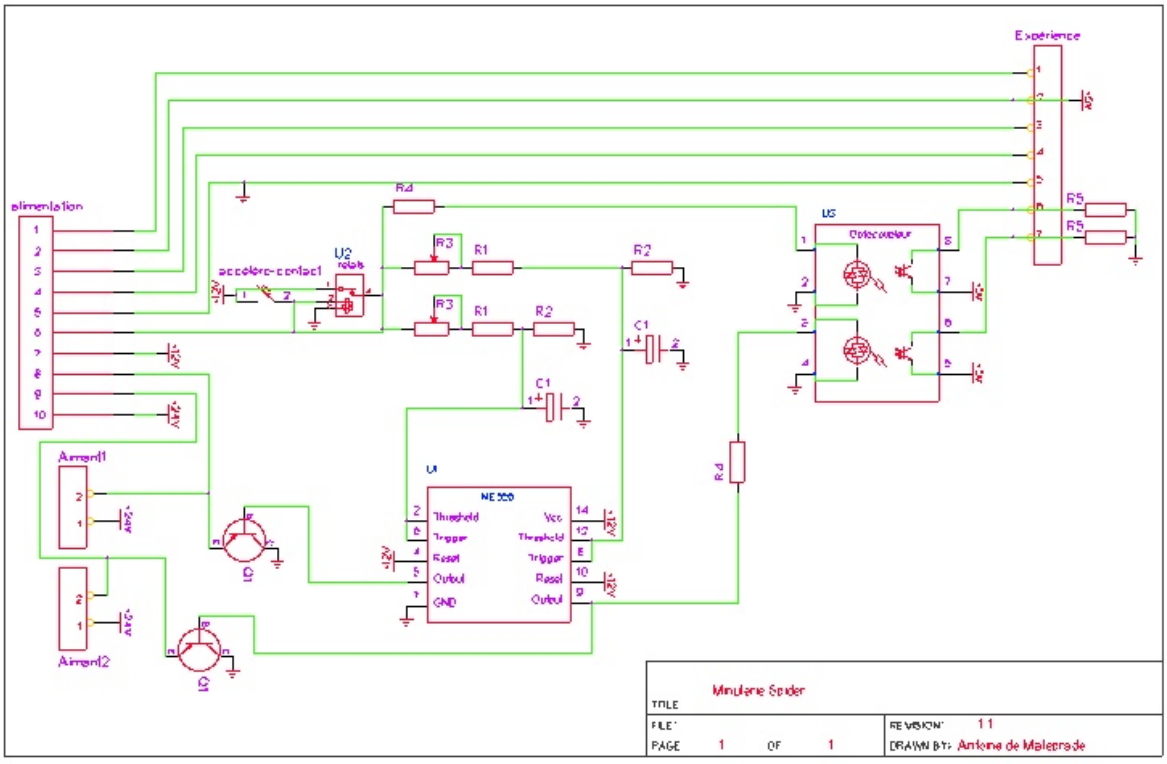


Sur la carte d'alimentation, on a donc les deux batteries en entrée avec les deux régulateurs pour avoir des tensions de 5-6-12 et 24V. Le connecteur 10 broches contient toutes les alimentations et informations vers les autres cartes :

1 : Alimentation 6V (servomoteur)
2 : Alimentation 5V (carte microcontrôleur)
3 : led « prêt » GPS
4 : led « prêt » expérience
5 : masse
6 : led décollage
7 : Alimentation 12V séquenceur
8 : Électro-aimant 1 (-)
9 : Électro-aimant 2 (-)
10 : Alimentation 24V

Séquenceur

Le séquenceur va donc être fabriqué autour d'un NE556 que l'on va paramétrer en fonction de deux condensateurs ainsi que de deux résistances plus deux résistances variables. On utilisera en sortie deux transistors pour connecter les Électro-aimants à la masse. Les signaux de décollage et d'activation du second Électro-aimant vont aussi être transmis au microcontrôleur par l'intermédiaire d'un optocoupleur.



Pour détecter le décollage, on utilise un accéléro-contact ainsi qu'un relais qui restera collé après le décollage jusqu'à une réinitialisation de la minuterie.

Les résistances R2 permettent le déchargement du condensateur pour ne pas avoir une durée aléatoire à cause d'un condensateur partiellement chargé.

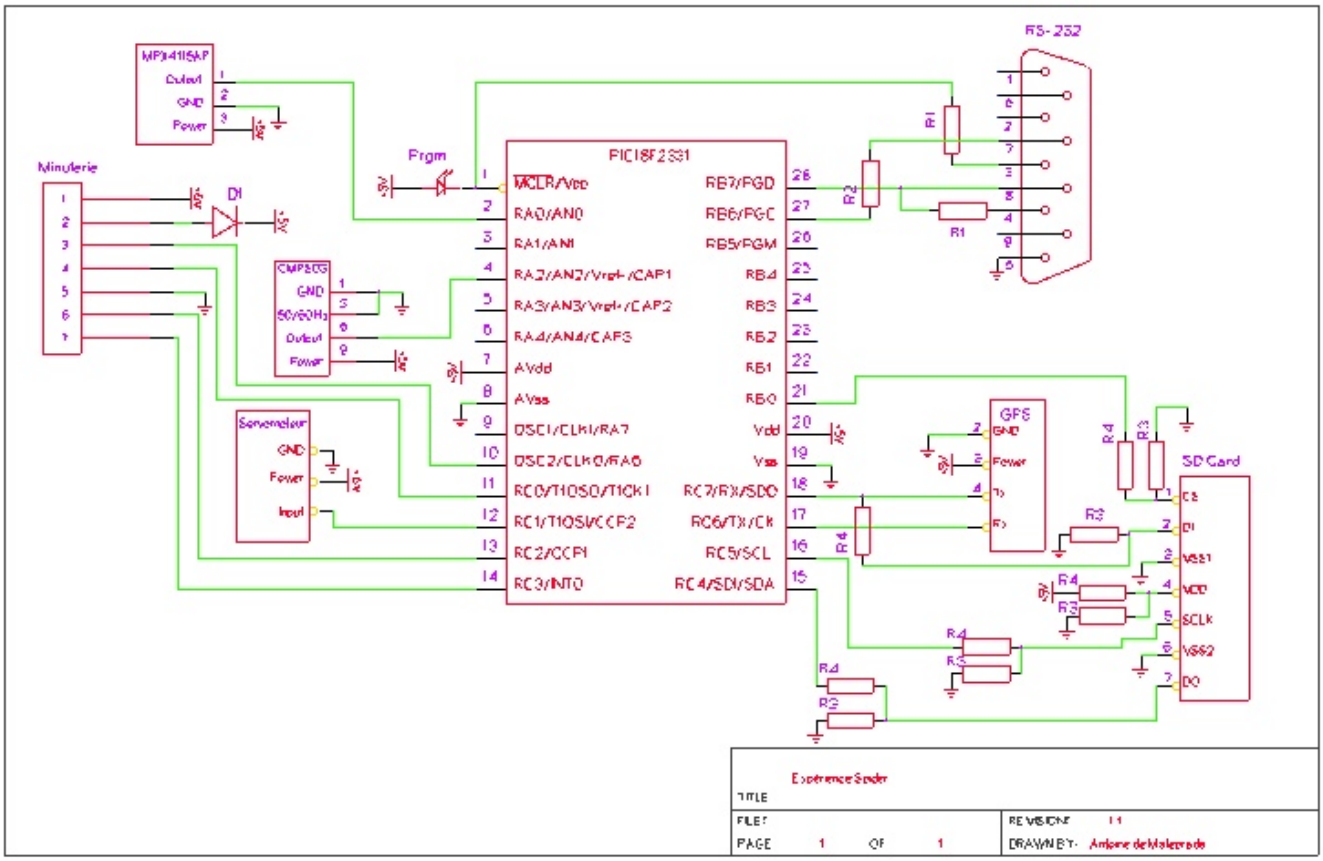
Le connecteur de gauche est le connecteur 10 broches venant de la carte alimentation. Les deux connecteurs 2 broches en bas à gauche sont les sorties vers les deux Électro-aimants. Le connecteur 7 broches de droite se dirige vers la carte expérience, il contient :

1 : Alimentation 6V (servomoteur)
2 : Alimentation 5V (carte microcontrôleur)
3 : led « prêt » GPS
4 : led « prêt » expérience
5 : masse
6 : information : décollage
7 : information : parapente ouvert (2eme Électro-aimant)

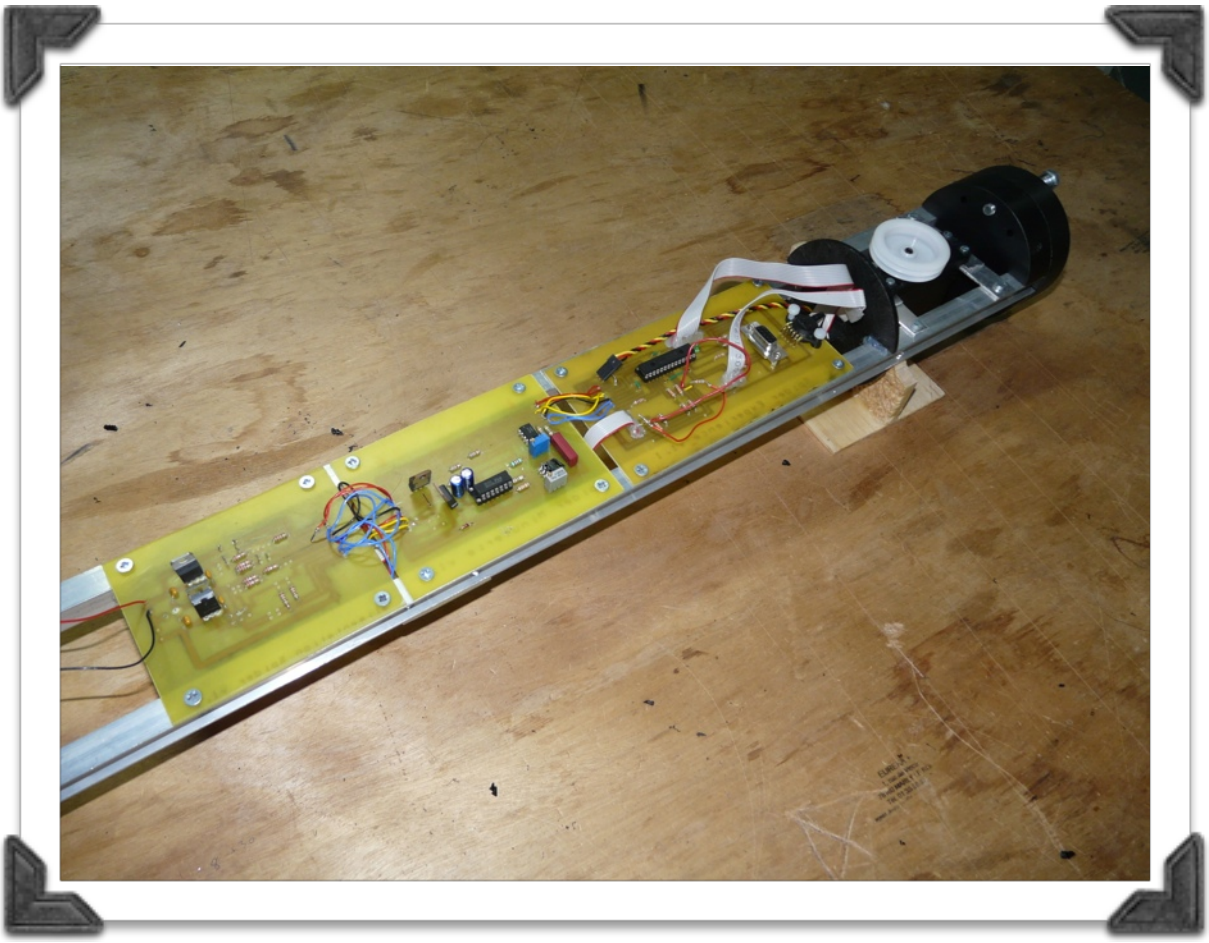
Carte du microcontrôleur

Le microcontrôleur nous a servi pour effectuer l'acquisition des capteurs ainsi que les calculs de trajectoire pour le vol. La carte contient donc un microcontrôleur PIC 18F2431 de chez Microchip utilisant son oscillateur interne. On aura sur la carte des connexions vers :

- **le GPS (UART)**
- **la boussole (PWM)**
- **le capteur de pression (analogique)**
- **le servomoteur (PWM)**
- **un connecteur RS232 pour une re-programmation du composant sans le retirer de la carte (avec un témoin lumineux)**



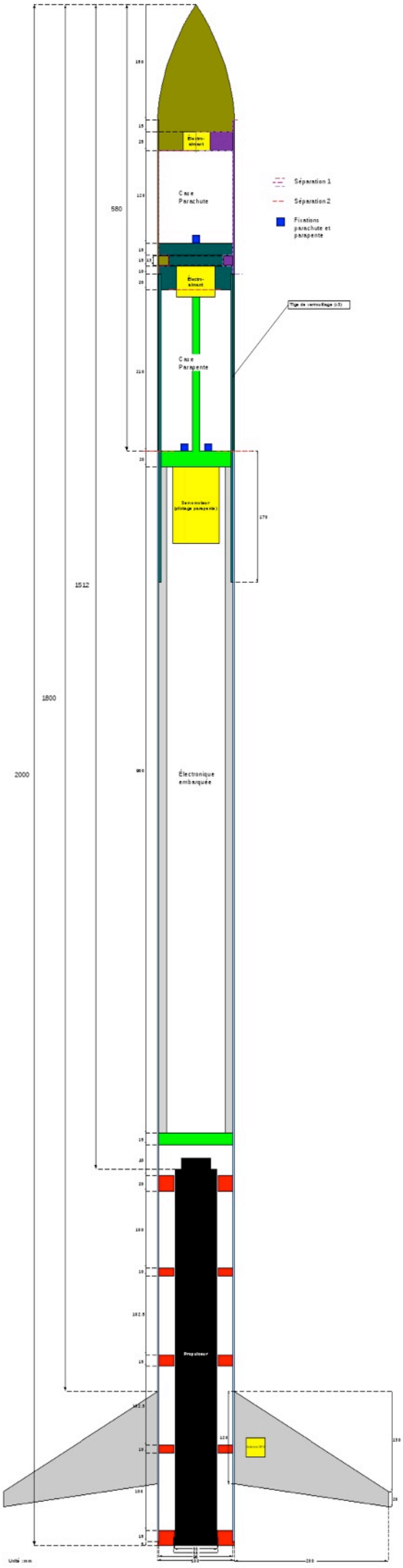
Le connecteur 7 broches à gauche correspond à celui de la carte du séquenceur. Une diode de protection à aussi été mise en place pour l'alimentation 5V.



4.La mécanique

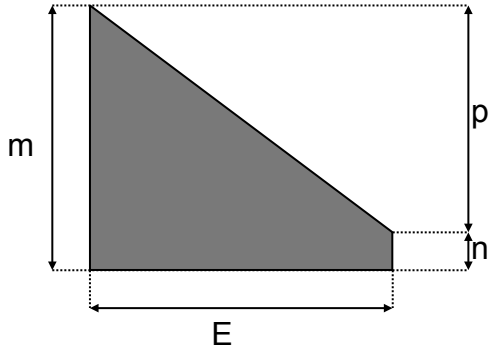
Voici les plans complets de la fusée (page suivante) contenant les indications de mesures et les différents compartiments de celles-ci. Récapitulatif :

- **Masse (sans moteur) : 8,75Kg**
- **Taille total : 2110mm**
- **Position du CDG (sans moteur) : 1100mm**
- **Implantation du propulseur : 1620mm**
- **diamètre de la fusée : 100mm**
- **taille de l'ogive : 150mm**
- **Maitre couple : 13854mm²**
- **Cx estimé : 0,7**

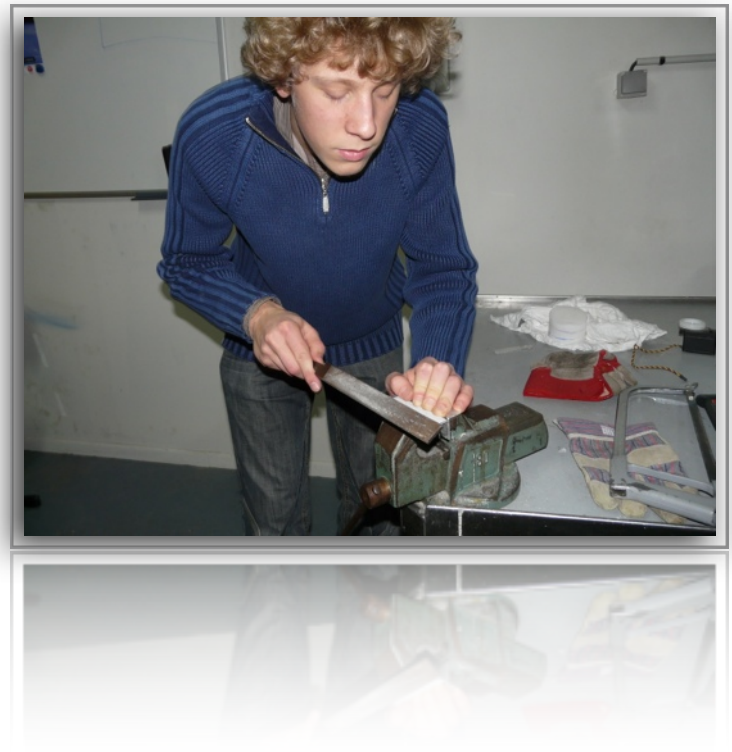


Aillerons :

- **m=200mm**
- **n=200mm**
- **p=80mm**
- **E=125mm**
- **épaisseur : 12mm (à cause de l'antenne du GPS)**
- **nombre : 4**
- **implantation : 1900mm (à partir de la pointe de l'ogive)**
- **diamètre : 100mm**



La fusée a été entièrement réalisée par Antoine et Dan avec le matériel du club.



La peinture…

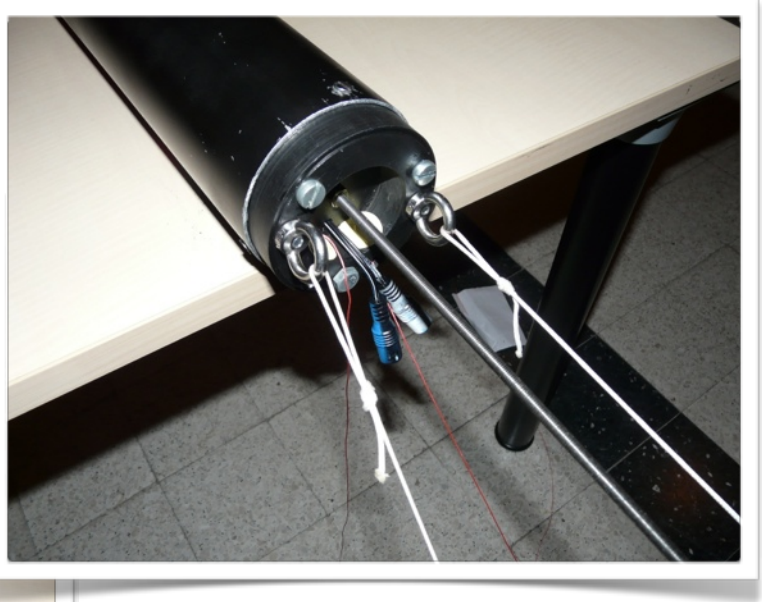
5.Sécurité

Nous avons prévu un bon nombre de sécurités pour ne pas sortir du cahier des charges imposé par le CNES et planète sciences :

- **Lors de la phase ascendante, le parachute et le parapente ne peuvent s'ouvrir à cause de la force dûe à la résistance de l'air pour des raisons mécaniques.**
- **Si le séquenceur ne fonctionne plus, rien ne se passe, la fusée fait un vol balistique.**
- **Si le séquenceur n'ouvre que le premier parachute, la fusée va descendre sous son seul parachute à 14m/s.**
- **Le parapente s'est ouvert mais le pilotage ne fonctionne pas : le servo moteur est toujours initialement placé en position complètement à gauche pour entrainer une vrille du parapente ce qui le fait chuter plus vite et verticalement.**
- **Le signal GPS n'est pas reçu ou bien perdu : On descend en vrille jusqu'à une nouvelle réception du signal GPS. (On ne peut savoir si la fusée risque de sortir du gabarit ou non)**
- **Une erreur dans l'algorithme ou bien le vent pousse le parapente et la fusée vers les limites du gabarit : le microcontrôleur vérifie périodiquement qu'il est assez loin des limites autorisées sinon il entraîne la vrille du parapente.**
- **Une erreur dans le programme le place dans une boucle infini : le chien de garde fait redémarrer le microcontrôleur et celui-ci place le servomoteur en position gauche quelques secondes avant de reprendre le pilotage (pour le cas où celui-ci entre de nouveau en boucle infinie).**

III.Les qualifications

(voir trajec et stabilito en annexe)



Le projet est qualifié !!!

Résultats du projet

La cible...

H-10min...

10 - 9 - 8 - 7 - 6 - 5 - 4 - 3 - 2 - 1 - 0 ... Mise à feu ... décollage ...

La fusée a fait un vol nominal lors du lancement au CELM de Biscarrosse Plage le 28 août 2009.

Propulsée par un Pro54, la fusée a effectué un vol stable, le mécanisme de séparation a parfaitement fonctionné libérant le parachute qui s'est ouvert à culmination, suivi par un succès de la deuxième séparation libérant l'aile de parapente débutant un vol stable en ligne droite.

La fusée a été retrouvée à 324 m de la cible ce qui est un résultat encourageant compte-tenu du vent le jour du lancement.

1.Ouverture du parachute

Le système d'ouverture du parachute comprenait deux petites portes en haut de la fusée en « peau de banane » et risquaient lors de l'ouverture de s'arracher avec la force du vent.

Celles-ci n'ont en réalité posé aucun problème et ont été récupérées intactes.
Le parachute s'est déployé correctement, sans torche et sans aucun problème.

Le système d'ouverture avec stabilisation de la fusée verticalement est donc un succès et a fonctionné selon nos attentes.

2.Ouverture et déploiement du parapente

Pour Déployer l'aile de parapente, nous avons utilisé une séparation transversale. Donc la fusée se séparait en deux, larguant la partie supérieure avec le parachute. Les risques que les suspentes du parapente s'emmêlent dans la tiges centrale était assez grand. (cf. Nom de la fusée)

Nous avons pu constater lors du vol, que le parapente s'est correctement déployé et nous avons pu aussi constater lors de la récupération de la fusée, que les suspentes ne s'étaient pas emmêlées autour de la tige centrale. Une fois encore ce système à fonctionné selon nos attentes.

3.Pilotage de l'aile

Peu de temps après l'ouverture de l'aile, celle-ci a commencé un virage engagé alors qu'elle s'était correctement déployée. Nous avons tenté de savoir pourquoi le logiciel de vol aurait programmé un virage engagé. Il y a plusieurs possibilités :

- **Le parapente n'ayant pu être testé dans des conditions de vol réel, le programme modifiait surement un angle trop important de l'aile pour corriger la trajectoire, et l'aile tournait trop vite pour que la fusée puisse suivre. Une fois le virage commencé, le programme n'a pas pu réagir.**
- **Une erreur dans le programme ou de mauvaise données des capteurs auraient pu tout aussi bien créer un dysfonctionnement aboutissant à la vrille de l'aile.**

1.GPS

Le GPS a parfaitement fonctionné car la position n'a pas été perdue lors du décollage ou bien il a raccroché très vite puisque dans le cas contraire, la fusée aurait avancé en ligne droite pendant 10 secondes (ce qui n'a pas été le cas) ou moins (jusqu'à récupérer le signal).
Pour obtenir de bonne performances nous avons utilisé un GPS évolué permettant de garder le signal avec de plus grandes accélérations qu'un GPS ordinaire.

2.Boussole

On a pu remarquer que lors de la récupération de la fusée, l'aile avait effectué des rotations sans que le corps de la fusée n'ait pu suivre. Le but de la boussole était de connaître la vitesse de rotation et la direction de l'aile. Il aurait été préférable d'intégrer la boussole à l'intérieur de l'aile pour ne pas perdre la précision de mesure due à la vrille des suspentes.

3.Algorithme

Celui-ci n'ayant pu être testé en vol avant le lancement, il est grandement possible qu'il soit incorrect ou qu'il comporte des bogues. La constante définissant l'angle de correction de base est à re-vérifier, il se peut qu'elle soit beaucoup trop importante.

La fusée "Spider" est sur la rampe, prête au décollage.

3, 2, 1, 0... décollage !

Eurêka Plus a participé à la campagne Nationale C'Space qui s'est déroulée au Centre d'Essais de Lancement de Missiles du 22 au 30 août à Biscarrosse. Bilan : vol nominal pour les fusées "Lunatique", "RC 9166" et "Spider", et vol balistique pour la fusée "Ze Squale Killer". La fusée expérimentale "Spider" devait atteindre une cible au moyen d'une aile et d'actuateurs permettant la modification de la trajectoire sous l'aile. Le vol de la fusée s'est décomposé en 4 phases : culmination, stabilisation de la fusée à l'aide d'un parachute (1 100 m) ; déploiement d'une aile (900 m) ; pilotage automatique de l'aile afin de rejoindre une cible et enregistrement des données de vol grâce aux capteurs embarqués (GPS, boussole et pression). "Le vol de Spider s'est parfaitement bien déroulé. La fusée a été retrouvée à 324 m de la cible ce qui est un résultat très encourageant" souligne Thibault Raboisson, animateur et président d'Eurêka Plus.

Infos : 01 39 58 87 92 - www.eurekaplus.org

Une première étape du guidage d'une fusée lors de sa chute a été réalisée avec sucès dans ce vol. Notre système d'ouverture fonctionne et permet une ouverture de l'aile de parapente lorsque la fusée est stabilisée. Nous avons pu remarquer aussi que certains GPS permettent de garder le signal même avec l'accélération de la fusée.

Bien que l'algorithme ne soit pas bien au point, cela nous en a appris plus sur le comportement de l'aile de parapente et sur les tests a effectuer.

Donc le projet a été un demi succès bien qu'il a été très difficile de le finir entièrement et à temps, vu que nous l'avons réalisé à deux, en un an.

Suivi d'une présentation au concours du GIFAS 2009 par Antoine de Maleprade et Dan Baczynski, ce projet a finalement remporté le prix EADS Space Transportation.

UN PRIX ESPACE INDUSTRIE POUR EURÊKA PLUS

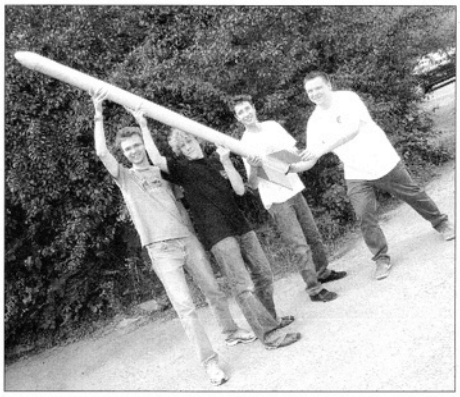
Depuis 1969, tous les deux ans, un jury de personnalités du milieu aéronautique et spatial remet à des clubs aérospatiaux de jeunes les Prix Espace Industrie.
Le 12 décembre dernier, au Musée de l'Air et de l'Espace du Bourget, Dan Baczynski et Antoine deMaleprade, du club Eurêka Plus, accompagnés du président de l'association, Thibault Raboisson, ont reçu le prix EADS Astrium des mains de Jean-Pierre Ledey, président de Planète Sciences, pour la présentation de leur fusée expérimentale "Spider". Il s'agissait de la 21e édition du prix Espace Industrie organisé par le GIFAS, le CNES et Planète Sciences, et des industriels qui manifestent ainsi leur intérêt pour les travaux et initiatives menés par les jeunes au sein de ces clubs.

Marly-le-Roi

Projet Eurêka Plus

À la conquête de l'espace

LA DERNIÈRE SEMAINE d'août verra une fusée marly-choise décoller du centre d'essais des Landes, à Biscarosse, lors du rassemblement "Espace Étudiant". L'engin mesurant deux mètres de long est en cours de finition dans les locaux de l'association Eurêka Plus. Depuis 24 ans, la structure et ses animateurs bénévoles permettent aux jeunes de s'initier aux sciences et aux techniques par le biais de projets aérospatiaux. Les mini-fusées sont l'apanage des plus jeunes, alors que les 14-25 ans s'attellent à la construction de fusées expérimentales, capables d'atteindre l'altitude de 1 500 mètres.



Antoine et Dan, à gauche, avec la fusée expérimentale qui décollera fin août, à Biscarosse.

UN AN DE TRAVAIL

Avec son propulseur à poudre installé par le Centre National d'Études Spatiales (CNES) le modèle qui décollera cet été n'a rien à envier à sa grande sœur Ariane. Antoine de Maleprade, 17 ans et Dan Baczynski, 16 ans, élèves en première au lycée Louis-de-Broglie, sont les deux constructeurs du modèle qui trône sur sa table d'assemblage. **«Le but d'un tel projet est de récupérer la fusée intacte avec sa carte mémoire qui enregistre tous les paramètres du vol»**, explique Thibault Raboisson, président de l'association. D'un poids de dix kilos, la fusée 2009 innove. Un système de parachute couplé à une aile delta ont été installés dans le cylindre d'aluminium. Le parachute laissera atterrir en douceur la partie supérieure de l'engin, alors que l'aile delta permettra au corps central un atterrissage dans une zone précise, définie à l'avance. La précision de quelques mètres constitue un challenge sur lequel les deux ingénieurs travaillent depuis la rentrée de septembre.

CALCULATEURS

Calculateurs électroniques et GPS vont piloter le retour sur terre. **«Nous utilisons des logiciels qui permettent de travailler sur la stabilité de la fusée. Les systèmes électroniques embarqués et le GPS sont conçus pour résister à de fortes accélérations»**, expliquent Antoine et Dan. D'un coût avoisinant les 2 000 euros, le projet bénéficie du soutien de sponsors et de l'aide de la ville de Marly. Grâce à un atelier, l'engin est entièrement construit sur place. Une vingtaine de jeunes marlychois seront cet été sur le pas de tir pour quelques secondes de bonheur qui viendront couronner plusieurs centaines d'heures de travail. Bon vol.

Emmanuel Fèvre

• www.eurekaplus.org

Tél: 01 39 58 87 92.

Sponsors

Annexes

I.Affiche de présentation de campagne
II.Stabilito
III.Trajec
IV.Courbes de pression et étalonnage
V.Référence de l'aile
VI.Chronologie
VII.Programme

Fusex **Spider**

Association Eurêka Plus

Objectifs du projet

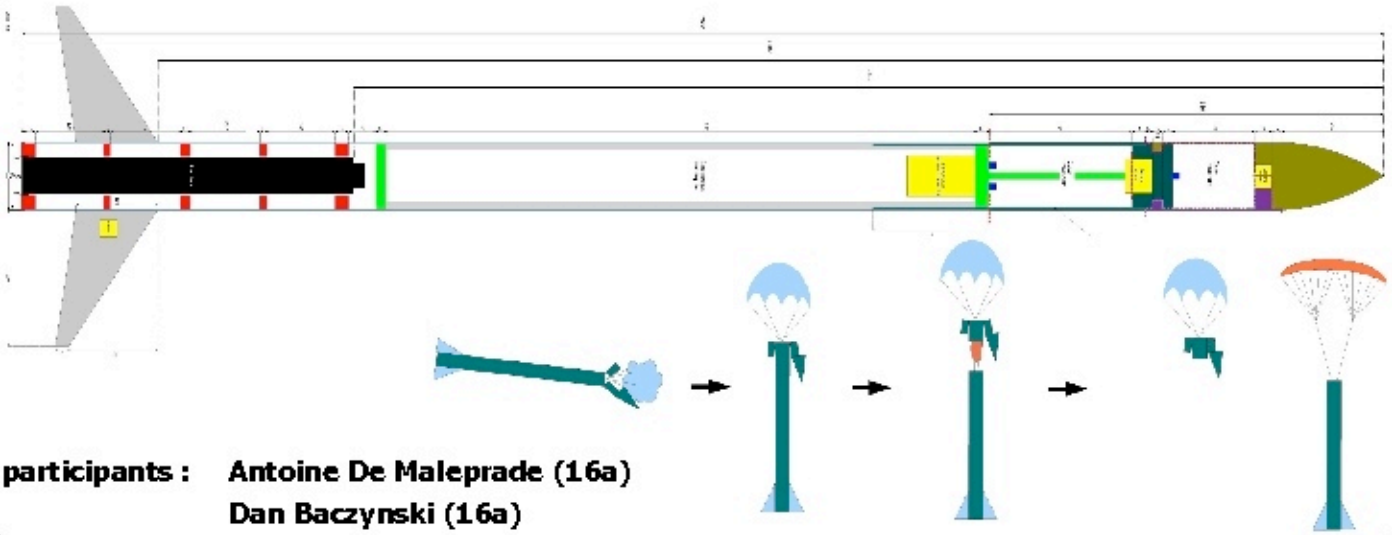
La fusée expérimentale Spider a pour but d'atteindre une cible au moyen d'une aile et d'actuateurs permettant la modification de la trajectoire sous l'aile. Le vol de la fusée se décompose en 4 phases :

Phase 1 : A culmination, stabilisation de la fusée à l'aide d'un parachute

Phase 2 : Déploiement d'une aile

Phase 3 : Pilotage automatique de l'aile afin de rejoindre la cible.

Phase 4 : Enregistrement de toutes les données de vol pour étude après vol (GPS, boussole, pression et commande du servomoteur)



2 participants : **Antoine De Maleprade (16a)**
Dan Baczynski (16a)

L'association Eurêka Plus

Eurêka Plus propose aux jeunes de 12 à 27 ans de s'initier aux sciences et techniques par le biais de la réalisation de projets aérospatiaux.

Des mini-fusées, fusées expérimentales et des ballons stratosphériques sont ainsi réalisés chaque année dans nos locaux Yvelinois. Ce sont des projets novateurs, à la pointe de la technologie qui permettent d'insuffler à la nouvelle génération, une passion pour l'espace, la recherche et la culture spatiale.

Depuis 24 ans, les animateurs bénévoles d'Eurêka Plus sont au service de la science.

www.eurekaplus.org **Tél.: 01.39.58.87.92** **EUREKA +, 1 rue de Viseu, 78160 Marly-le-Roi**

STABILITO

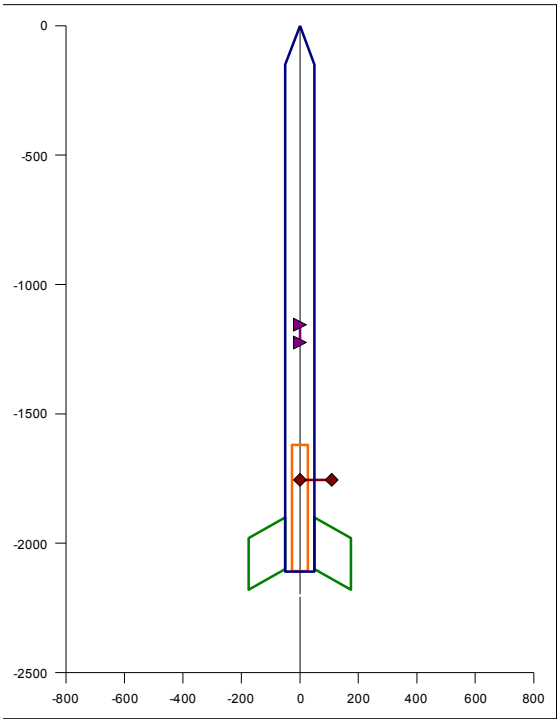
Stabilité des fusées à ailerons
Version 2.2 mono-empennage
Remplir les cases jaunes

Language:	Français

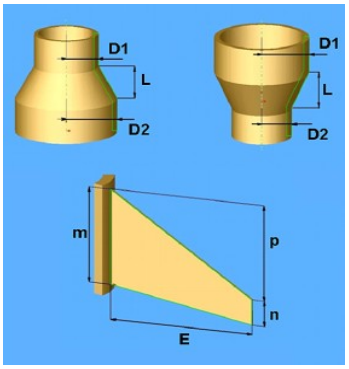
	Fusée	
Nom	Spider	
Club	Eureka+	
Type	Fusée expérimentale	
Masse	8,75 kg	sans propu
Centre de Masse	1100 mm	sans propu
Longueur totale	2110 mm	

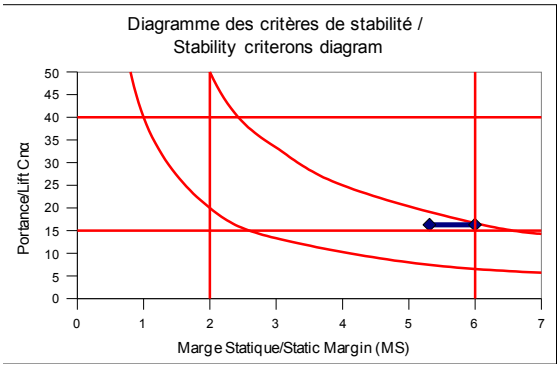
	Ogive
Hauteur	150 mm
Forme	Parabolique (arrondie)
Diamètre	100 mm
Diamètre Réf.	100 mm

	JupeRétreint1	JupeRétreint2
L		
D1		
D2		
Implantation		



	Ailerons
m	200 mm
n	200 mm
p	80 mm
E	125 mm
Epaisseur	12 mm
Nombre	4
Implantation	1900 mm
Diamètre	100 mm





	Propulseur
Type	8 : Pro54-5G (ex
Implantation	1620 mm
Poussée	800 N
Impulsion totale	2060 N.s

Rampe	
Longueur	4 m
	67 m/s²
	0,3 s

	Propu plein	Propu vide	Sans propu
Masse propu	1,68 kg	0,69 kg	-
CdM propu	250 mm	240 mm	-
Masse fusée	10,43 kg	9,44 kg	8,75 kg
CdM fusée	1224 mm	1156 mm	1100 mm

	XCp	Cnα
Ailerons	1990 mm	14,3
Ogive	75 mm	2,0
Jupe|Rétreint 1	0 mm	0,0
Jupe|Rétreint 2	0 mm	0,0

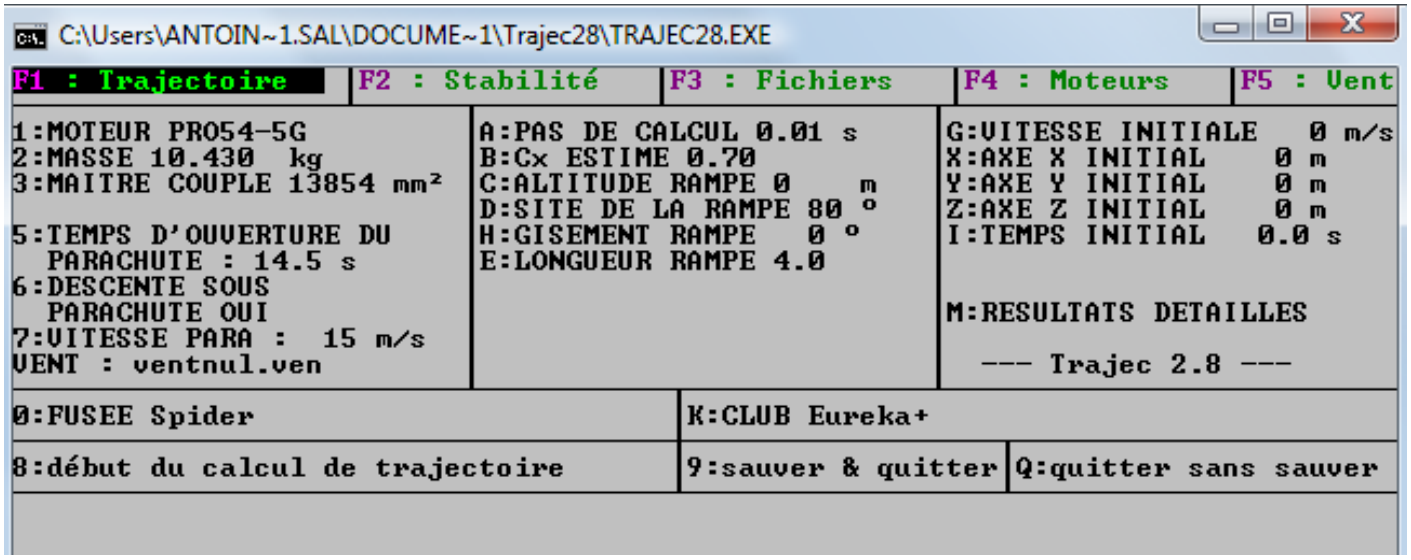
Critères	Fusée expérimentale	
Décollage	20 m/s	-
Finesse	10	35
Cnα	15	40
MS	2 D	6 D
MS*Cnα	40	100

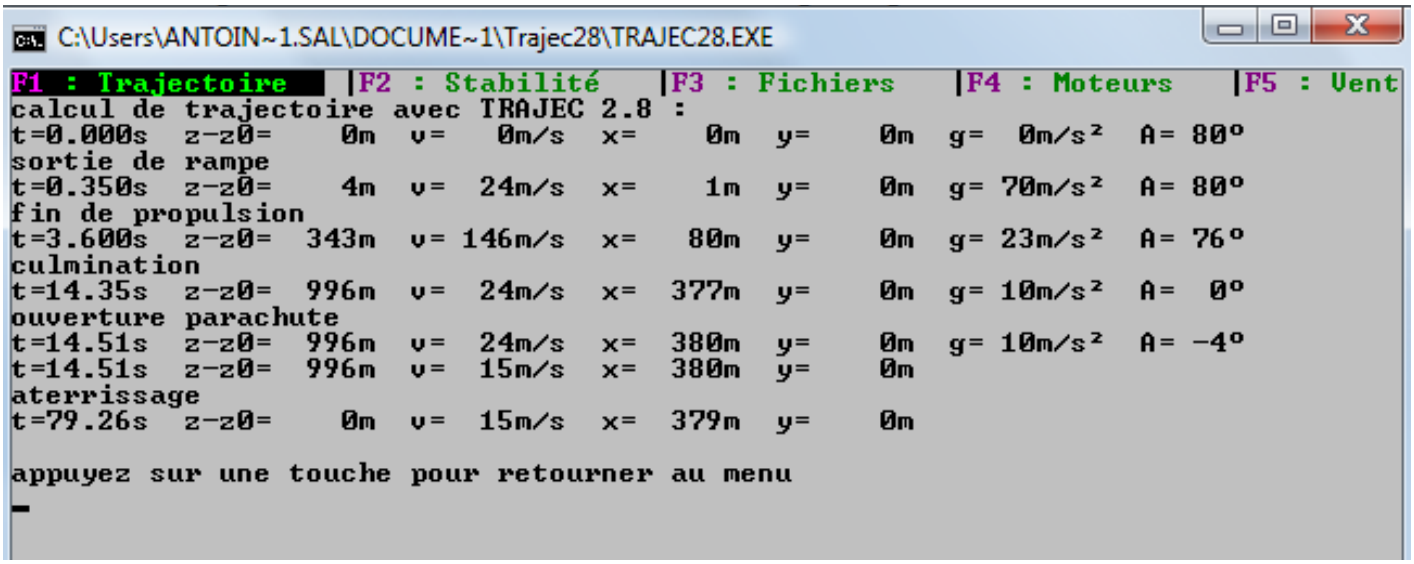
Résultats		10/10/2009	
Décollage	~23,1 m/s		OK
Apogée	~1099 m		
Culminat°	~15 s		
Finesse	21,1		OK
Cnα	16,3		OK
XCp	1755 mm		
MS	5,31 D	5,997 D	OK
MS*Cnα	86,7	97,8	OK

Conclusion	STABLE

Commentaire libre : La fusée est limite surstable (MS = 5,997 et MS <= 6)

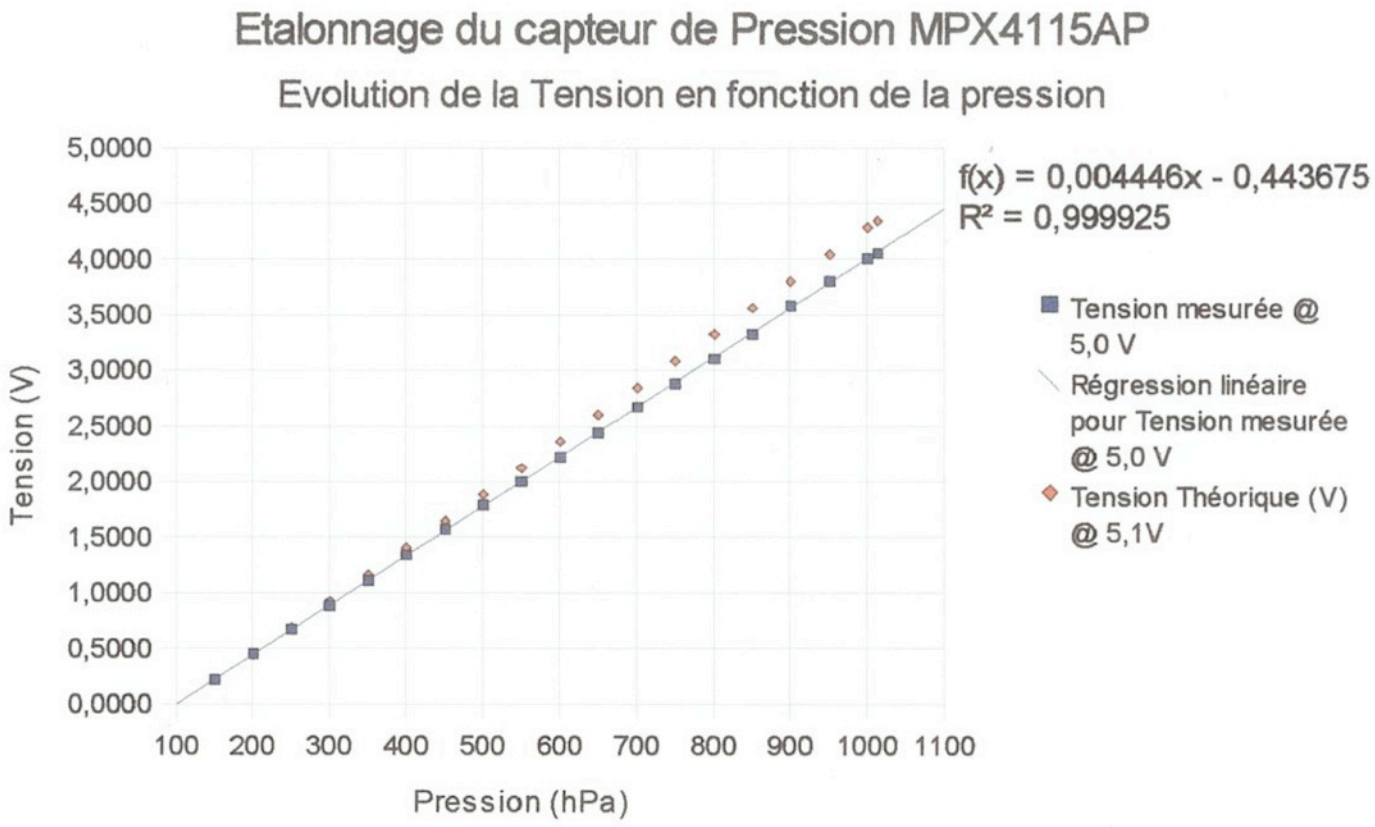
III.Trajec


C:\Users\ANTOIN~1.SAL\DOCUME~1\Trajec28\TRAJEC28.EXE
F1 : Trajectoire
F2 : Stabilité
F3 : Fichiers
F4 : Moteurs
F5 : Vent
1:MOTEUR PRO54-5G
2:MASSE 10.430 kg
3:MAITRE COUPLE 13854 mm²
5:TEMPS D'OUVERTURE DU PARACHUTE : 14.5 s
6:DESCENTE SOUS PARACHUTE OUI
7:VITESSE PARA : 15 m/s
VENT : ventnul.ven
A:PAS DE CALCUL 0.01 s
B:Cx ESTIME 0.70
C:ALTITUDE RAMPE 0 m
D:SITE DE LA RAMPE 80 °
H:GISEMENT RAMPE 0 °
E:LONGUEUR RAMPE 4.0
G:VITESSE INITIALE 0 m/s
X:AXE X INITIAL 0 m
Y:AXE Y INITIAL 0 m
Z:AXE Z INITIAL 0 m
I:TEMPS INITIAL 0.0 s
M:RESULTATS DETAILLES
--- Trajec 2.8 ---
0:FUSEE Spider
K:CLUB Eureka+
8:début du calcul de trajectoire
9:sauver & quitter
Q:quitter sans sauver


C:\Users\ANTOIN~1.SAL\DOCUME~1\Trajec28\TRAJEC28.EXE
F1 : Trajectoire
F2 : Stabilité
F3 : Fichiers
F4 : Moteurs
F5 : Vent
calcul de trajectoire avec TRAJEC 2.8 :
t=0.000s z-z0= 0m v= 0m/s x= 0m y= 0m g= 0m/s² A= 80°
sortie de rampe
t=0.350s z-z0= 4m v= 24m/s x= 1m y= 0m g= 70m/s² A= 80°
fin de propulsion
t=3.600s z-z0= 343m v= 146m/s x= 80m y= 0m g= 23m/s² A= 76°
culmination
t=14.35s z-z0= 996m v= 24m/s x= 377m y= 0m g= 10m/s² A= 0°
ouverture parachute
t=14.51s z-z0= 996m v= 24m/s x= 380m y= 0m g= 10m/s² A= -4°
t=14.51s z-z0= 996m v= 15m/s x= 380m y= 0m
aterrissage
t=79.26s z-z0= 0m v= 15m/s x= 379m y= 0m
appuyez sur une touche pour retourner au menu

Étalonnage du capteur de pression MPX4115AP

Pression (hPa)	Tension mesurée @ 5,0 V	Tension Théorique (V) @ 5,1V
0	0,0616	0,204
50	0,0617	0,204
100	0,0623	0,204
150	0,2180	0,204
200	0,4500	0,444
250	0,6700	0,683
300	0,8800	0,923
350	1,1100	1,163
400	1,3400	1,403
450	1,5700	1,642
500	1,7900	1,882
550	2,0000	2,122
600	2,2200	2,361
650	2,4400	2,601
700	2,6700	2,841
750	2,8800	3,080
800	3,1000	3,320
850	3,3200	3,560
900	3,5800	3,800
950	3,8000	4,039
1000	4,0000	4,279
1013	4,0500	4,341



S P I D E R

Chronologie

Heure	Durée	Lieu	Action	X
La veille	2 min	R3	*Démonter la structure*	
	1 min		*Mettre les interrupteurs en position centrale (off)*	
	10 min		*Changer toutes les piles à fixer avec du scotch*	
	5 min		*Bloquer les réglages avec le vernis (2 trimmers, tige filetée elec et masse sur accéléro)*	
	10 min		*Vérifier toutes les connections et pisto-coller les composants « sensibles »*	
	2 min		*Vérifier la fixation des interrupteurs et bloquer avec vernis*	
	4 min		*Brancher antenne GPS et la bloquer avec du pisto-colle et scotch*	
	1 min		*Mettre les interrupteurs en position Gauche (On) pour centrer le servomoteur*	
	10 min		*Mettre en place les freins et faire cinq autour de l'axe dans chaque sens*	
	5 min		*Maintenir en position « tendu » les deux freins au centre (pâte à fixe/double face)*	
	2 min		*Intégrer la structure dans la fusée et visser la structure*	
Jour J	5 min	R3	*Vérifier les charnières, la fixation de l'anneau parachute (point de vernis de blocage)*	
	5 min		*Vérifier la vis du petit électro-aimant et mettre les deux vis de l'ogive*	
	2 min		*Vérifier le domino du petit électro-aimant*	
	15 min		*Démonter la pièce du haut et vérifier le domino du gros électro et Remonter la pièce*	
	15 min		*Vérifier la fixation du parachute (refaire les nœuds) et la longueur des suspentes*	
	5 min		*Plier le parachute*	
	2 min		*Fermer la porte parachute*	
	10 min		*Vérifier la longueur et la fixation des suspentes et des freins du parapente*	
	10 min		*Plier le parapente*	
	10 min		*Brancher les jacks (Couleurs), appliquer la graisse, refermer le conteneur parapente*	
	1 min		*Vérifier les interrupteurs en position centrale (Off)*	
	20 min		*Partir pour le Plan d'Op*	
Jour J	1 min	Tente club	*Mettre interrupteurs en position gauche (On)*	
	1 min		*Vérifier le fonctionnement des capteurs (3 clignotements led verte puis led allumée)*	
	5 min		*Attendre la réception du signal GPS (clignotement led bleue)*	
	1 min		*Mettre les interrupteurs en position centrale (Off)*	
H – 60min	5 min	Rampe	*Monter en rampe*	
	1 min		*Mettre les interrupteurs en position gauche (On)*	
	5 min		*Attendre la réception du signal GPS (clignotement led bleue)*	
	1 min		*Mettre les interrupteurs en position centrale (Off)*	
	10 min		*Faire de compatibilité rampe*	
	15 min		*Mise en place du Pro54*	
	1 min		*Mettre les interrupteurs en position gauche (On)*	
	1 min		*Vérifier les deux led verte (minuterie, expérience) ainsi que la led verte capteurs*	
	5 min		*Attendre la réception du signal GPS (clignotement led bleue)*	
	1 min		*Vérifier les deux led verte (minuterie, expérience) ainsi que la led verte capteurs*	
	10 min		*Evacuer la zone rampe*	
H – 15min	10 min		*Mettre en place l'inflammateur*	
H – 10sec	10 sec		*Décompte final*	
H – 0	0 min		*Mise à feu*	

V.aile de parapente : PW 1,9 (tribord)

```
#include "p18f2431.h"  //Inclusion préprocesseur
#include "delays.h"                 //Inclusion header pour delay
#include "math.h"                   //Inclusion du module math


#pragma config OSC = IRCIO, WDTEN = ON, WDPS = 1024, PWRTEN = ON, BORV = 20         //Oscillateur interne de 8Mhz, WathDogTimer on
(1/1024), Temporisation au démarrage on, tension de Brown-out = 2.0V

//Definitione des Ports utilisés

#define led PORTCbits.RC0                   //led pour le fonctionnement du gps
#define accContact PORTCbits.RC3//Accélérocontact signalisant le décollage de la fusée
#define sd PORTBbits.RB0                    //Sélection de la carte SD
#define led2 PORTBbits.RB1                  //led pour le fonctionnement des autres périphériques (boussole, pression, sd, EEPROM)
#define parapente PORTBbits.RB2 //Déclenchement de la minuterie de l'ouverture parapente
#define servomoteur CCPR2                   //Registres contenant l'angle du servomoteur pour le pilotage parapente


//Définition des constantes

float cibleLat = 44.32747;                          //Latitude de la cible en dégrées
float cibleLon = -1.26967;                          //Longitude de la cible en dégrées
const char vht=4;                           //Vitesse horizontale théorique
const float lmLon=0.7154489;//cos(cibleLat*pi/180)
const float zonerougeLatHaut=44.33297;//Limite nord de la zone "navigable" en dégrées
const float zonerougeLatBas=44.31964; //Limite sud de la zone "navigable" en dégrées
const float zonerougeLonGauche=-1.276787;//Limite ouest de la zone "navigable" en dégrées
const float zonerougeLonDroite=-1.25955;//Limite est de la zone "navigable" en dégrées
const float pi = 3.141592653589793;         //valeur de pi pour les calculs trigonométriques

//Définition des variables pour Décompte de temps

unsigned char ms100 = 0;            //Comptage du temps en dixième de seconde
unsigned long temps = 0;            //Comptage du temps en seconde
unsigned char sec = 0; //Comptage du temps en seconde
unsigned char para=0; //Comptage du temps depuis ouverture parapente en seconde


//Définition des variables pour R/W EEPROM
unsigned char i=0;                  //Variable temporaire pour effacer le buffer
unsigned char EEbuf[4]={0,0,0,0};//buffer mémoire EEPROM
unsigned char EEwrite=4;            //position du prochaine emplacement EEPROM à écrire

//Définition des variables pour la capteur de pression/altitude
unsigned int pression = 0;//Pression absolu en hPa
int altitude = 0;                   //Altitude en mètres
int altitude2 = 0;                  //Altitude précédentes en mètres (pour calcul du taux de chute)
int altCible=0;                             //Altitude de la cible
int h=0;                                    //hauteur entre la fusée et le cible

//Définition des variables pour la boussole, la vitesse angulaire et le cap
unsigned char debTMR1 = 0;          //Comptage du nombre de débordement du timer1 entre deux pulsation de la boussole
unsigned char boussolenok = 0;//Comptage du temps pendant lequel la boussole n'a pas emis de signal en seconde
int route = 0;                              //route de la fusée en dégrées
int route2 = 0;                             //route précédente de la fusée en dégrées pour le calcul de la vitesse angulaire
int vang = 0;                               //vitesse angulaire horizontale de la fusée

//Défifnition des variables pour le pilotage de la fusée
static int cap = 0;                                 //cap à suivre en dégrées
static int captmp = 0;                      //variable temporaire pour calcul du cap a suivre
static int corr = 0;                        //Correction à effectuer sur l'angle du servomoteur
static unsigned int servotemp = 3000;//variable temporaire pour l'angle du servomoteur

//Information sur la vitesse et position de la fusée
float posLat = 0;                   //Latitude de la fusée en dégrées
float posLon = 0;                   //Longitude de la fusée en dégrées
float speed = 0;                            //Vitesse de la fusée en noeud(gps)
float course = 0;                   //route de la fusée en dégrées(gps)

//Déclaration des variables pour calcul de la correction de trajectoire
float xth = 0;                                  //déplacement longitudinal théorique
float yth = 0;                                  //déplacement en latitude théorique
float tempvh = 0;                           //Variable temporaire pour le calcul de la vitesse moyenne de déplacement horizontal
sur 10secondes
float vh = 0;                                   //Vitesse moyenne de déplacement horizontal sur 10secondes
float vc = 0;                                   //taux de chute (pression)
float x = 0;                                            //position par rapport à la cible (m)
float y = 0;                                            //position par rapport à la cible (m)
int x2 = 0;                                     //position précédente par rapport à la cible (m)
int y2 = 0;                                     //position précédente par rapport à la cible (m)
float xvent = 0;                                //Déplacement en x dû au vent
float yvent = 0;                                //Déplacement en y dû au vent
unsigned float dist = 0;                    //distance par rapport à la cible (m)
```

```
unsigned float distvent = 0;                    //distance par rapport à la cible après correction de trajectoire (m)
unsigned float dopt = 0;                                //distance optimale à laquelle la fusée peut se diriger vers la cible

unsigned char gpsnok = 0;
long tempfloat = 0;

struct {
        unsigned pilotage:1;    //Activation/Désactivation du pilotage
        unsigned error:1;               //Erreur réception gps
        unsigned decollage:1;   //Déclenchement décollage
        unsigned gpsok:1;               //Réception du gps valide/invalide
        unsigned initsdok:1;    //initialisation carte sd ok
        unsigned errdem:1;      //Erreur au niveau de l'un des capteurs
        unsigned ouvert:1;      //Ouverture du parapente
        unsigned zonerouge:1;   //Entrée en zone rouge
        unsigned virage:1;      //mode virage activer/Désactiver
        unsigned gps3D:1;               //Gps en mode 3D
        unsigned bslok:1;
        unsigned rmc:1;
        unsigned gsa:1;

} bitram;

//Déclaration des Fonctions


static void init(void);                 //Déclaration de la fonctions d'initialisation

//inclusion des headers

#include "guidage.h"
#include "gps.h"                //Inclusion du module pour la réception des trames gps
#include "boussole.h"   //Inclusion du module pour le capteur boussole
#include "inter.h"              //Interruptions


//Programme principal

void main(void){

        init();//Initialisation

        if(RCONbits.POR==0){
                altCible=altitude;
                EEbuf[0]=altCible/0x100;
                EEbuf[1]=altCible-EEbuf[0]*0x100;
                EEwrite=14;
                PIR2bits.EEIF=1;
        }else{
                EEADR=4;
                EECON1bits.RD=1;
                altCible=EEDATA*0x100;
                EEADR=5;
                EECON1bits.RD=1;
                altCible+=EEDATA;
        }

        if(accContact==1){
                Delay10KTCYx(250);
                ClrWdt();
                if(parapente==1) {
                        bitram.pilotage=1;
                        bitram.ouvert=1;
                }
                if(bitram.gpsok==0) bitram.pilotage=0;
        }
        Delay10KTCYx(100);
        ClrWdt();

        led2=1;
        Delay10KTCYx(50);
        led2=0;
        Delay10KTCYx(50);
        ClrWdt();

        if(bitram.bslok==1){
                led2=1;
        }else bitram.errdem=1 ;
        Delay10KTCYx(50);
        led2=0;
        Delay10KTCYx(50);
        ClrWdt();
```

```
		if(route!=0){ led2=1;
		}else bitram.errdem=1 ;
		Delay10KTCYx(50);
		led2=0;
		ClrWdt();

		if(bitram.errdem==0) led2=1;

		RCONbits.POR=1;
		while(parapente==0) {
				if(gpsnok>5){
						gpsnok=10;
						bitram.gpsok=0;
						led=0;
				}else{
						bitram.gpsok=1;
				}
				if(bitram.rmc==1)GPRMC();
				if(bitram.gsa==1)GPGSA();
				ClrWdt();
		}
		bitram.ouvert = 1;
		while(1){
				ClrWdt();
				if(bitram.rmc==1)GPRMC();
				if(bitram.gsa==1)GPGSA();
				if(gpsnok>5){
						gpsnok=10;
						bitram.gpsok=0;
						led=0;
				}else{
						bitram.gpsok=1;
				}
				if(para<20 && bitram.gpsok==0){
						bitram.pilotage=0;
						servomoteur=3000;
				}else if(bitram.zonerouge==1||bitram.gpsok==0){
						bitram.pilotage=0;
						servomoteur=1800;
						bitram.virage=1;
						para=20;
				}else{
						bitram.pilotage=1;
						para=20;
				}
				ClrWdt();
		}
}

static void init(void){
		OSCCON = 0x7F;			//oscillateur intene de 8Mhz

		PORTA = 0;
		PORTB = 0b00000001;	//Désélection carte SD
		PORTC = 0b00000010;	//signal du servomoteur Ã  l'Ã©tat haut
		//led=0;				//initialisation des led.
		//led2=0;
		//TRISA = 0b11111111;	//Pression sur AN0/RA0
		TRISB = 0b11111100;	// Selection SD(RB0) / led2(RB1) / parapente(RB2) / PGM(RB5) / PGC(RB6) / PGD(RB7)
		TRISC = 0b10011100;	//led(RB0) / servomoteur (RB1/CCP2) / boussole (RB2/CCP1) / decollage (RB3/INT0) / SD out(RB4/SDI) / SD
clock (RB5/SCl) / gps RC (RB6/TX) / SD in et gps TX (RB7/SDO/RX)

		bitram.pilotage = 0;
		bitram.gpsok = 0;
		bitram.error = 0;
		bitram.decollage = 0;
		bitram.initsdok = 0;
		bitram.errdem = 0;
		bitram.ouvert = 0;
		bitram.zonerouge = 0;
		bitram.virage = 0;
		bitram.gps3D = 0;
		bitram.bslok = 0;
		bitram.rmc = 0;
		bitram.gsa = 0;

		RCONbits.IPEN=1;
		INTCON = 0b11010000;
		INTCON2 = 0b11000000;
		//INTCON3 = 0b00000000;
		//PIR1 = 0;
		//PIR2 = 0;
		//PIR3 = 0;
```

```
	PIE1 = 0b01100101;
	//PIE2 = 0b00000000;
	PIE3 = 0b00000001;
	IPR1 = 0b00000101;
	IPR2 = 0b00000000;
	//IPR3 = 0b00000001;
	TMR1L = 0;
	TMR1H = 0;
	T1CON = 0b10000001;

	PR2 = 0xFF;
	TMR2 = 0;
	T2CON = 0b00000011;

	TMR5L = 0;
	TMR5H = 0;
	PR5H = 0x61;
	PR5L = 0xA7;
	T5CON = 0b11011001;

	CCP1CON = 0b00000101;
	CCPR2H=0xB;
	CCPR2L=0xB8;
	CCP2CON = 0b00001001;

	ANSEL0 = 0b00000001;
	ADCHS = 0b11111100;
	ADCON1=0b00000000;
	ADCON2=0b10000000;
	//ADCON3=0b00000000;
	ADCON0=0b00000011;

	SPBRG = 25;//9600bauds=12 4800bauds=25
	//SPBRGH = 0x00;
	//BAUDCTL = 0b00000000;
	//TXSTA = 0b00000000;
	RCSTA = 0b00010000;

	//EECON1=0;

	ClrWdt();
	RCSTAbits.SPEN = 1;
	PIE1bits.RCIE=1;
	ClrWdt();
}
```

Boussole.h

```
unsigned int boussole1 = 0;
unsigned int boussole2 = 0;
unsigned short long boussoletmp = 0; //temp d'une pulsation de la boussole -1 en microsecondes

static void lectBoussole(void){
	boussole2 = boussole1;
	boussole1 = CCPR1;
	boussoletmp = boussole1 + debTMR1 * 0x10000;
	boussoletmp -= boussole2;//C350
	route = (boussoletmp/200 - 660);
	route=route-mvar;
	if((route>=360)||(route<0)) route=0;
}
```

gps.h

```
unsigned char gpsRC[74];
unsigned char gpsRCRMC[74];
unsigned char RC=0;
unsigned char tr=0;
unsigned char RCtemp=0;
unsigned short long heure=0;
unsigned char RClen = 0;
float mvar=0;

void AquGPS(void){
	heure = (gpsRCRMC[7]-0x30)*36000+(gpsRCRMC[8]-0x30)*3600+(gpsRCRMC[9]-0x30)*600+(gpsRCRMC[10]-0x30)*60+
(gpsRCRMC[11]-0x30)*10+(gpsRCRMC[12]-0x30);
	posLat = (gpsRCRMC[16]-0x30)*10000000+(gpsRCRMC[17]-0x30)*1000000+((gpsRCRMC[18]-0x30)*1000000+(gpsRCRMC[19]-0x30)*100000+
(gpsRCRMC[21]-0x30)*10000+(gpsRCRMC[22]-0x30)*1000+(gpsRCRMC[23]-0x30)*100+(gpsRCRMC[24]-0x30)*10)/6;
	posLat = posLat/1000000;
	if(gpsRCRMC[26]==0x53) posLat= -posLat;
	posLon = (gpsRCRMC[28]-0x30)*100000000+(gpsRCRMC[29]-0x30)*10000000+(gpsRCRMC[30]-0x30)*1000000+
((gpsRCRMC[31]-0x30)*1000000+(gpsRCRMC[32]-0x30)*100000+(gpsRCRMC[34]-0x30)*10000+(gpsRCRMC[35]-0x30)*1000+
(gpsRCRMC[36]-0x30)*100+(gpsRCRMC[37]-0x30)*10)/6;
	posLon = posLon/1000000;
	if(gpsRCRMC[39]!=0x45) posLon= -posLon;
	speed=(gpsRCRMC[41]-0x30)*1000+(gpsRCRMC[42]-0x30)*100+(gpsRCRMC[43]-0x30)*10+(gpsRCRMC[45]-0x30);
	speed = speed/10;
	course=(gpsRCRMC[47]-0x30)*1000+(gpsRCRMC[48]-0x30)*100+(gpsRCRMC[49]-0x30)*10+(gpsRCRMC[51]-0x30);
	course = course/10;
	mvar=(gpsRCRMC[60]-0x30)*1000+(gpsRCRMC[61]-0x30)*100+(gpsRCRMC[62]-0x30)*10+(gpsRCRMC[64]-0x30);
	mvar=mvar/10;
	if(gpsRCRMC[66]==0x45)mvar= -mvar;
}
void GPRMC(void){
	bitram.rmc=0;
	if(bitram.gpsok==1)led=1;
	AquGPS();
	if(bitram.ouvert==1){
		CalculCoorCarth();
		CalculVitesse();
		if(posLat>zonerougeLatHaut||posLat<zonerougeLatBas||posLon<zonerougeLonGauche||posLon>zonerougeLonDroite){
			bitram.zonerouge=1;
		}else bitram.zonerouge=0;
		CalculPosTheorique();
		if(sec>=10){
			CalculCorrTrajec();
			sec=0;
		}
		sec++;
		CalculCap();
	}
	if(bitram.gps3D==1)led=0;
}

void GPGSA(void){
	if(gpsRC[9]==0x33){
		bitram.gps3D=1;
		gpsnok=0;
	}else if(gpsRC[9]==0x32){
		bitram.gps3D=0;
		gpsnok=0;
	}
	bitram.gsa=0;
}
```

```
void CalculCoorCarth(void){
	x = (cibleLon - posLon)*lmLon*111120;
	y = (cibleLat-posLat)*111120;
	dist = sqrt(x*x + y*y);
}
void CalculCorrTrajec(void){
	xvent=x2-x-xth;
	yvent=y2-y-yth;
	if(bitram.virage==0){
			vc=(altitude-altitude2)/10;
			vh=tempvh/10;
	}
	tempvh=0;
	xth=0;
	yth=0;
	x2=x;
	y2=y;
	altitude2=altitude;
}
void CalculCap(void){
	dopt=h*vh/vc+15;
	if(dist<100&&dist>dopt){
			bitram.pilotage=0;
			bitram.virage=1;
			servomoteur=3800;
	}else{
			bitram.pilotage=1;
			bitram.virage=0;
			xvent=x-xvent;
			yvent=y-yvent;
			distvent=sqrt(xvent*xvent+yvent*yvent);
			if(distvent!=0){
				captmp=asin(xvent/distvent)*180/pi;
				if(yvent<0) captmp=180-captmp;
				if(captmp<0) captmp+=360;
				cap=captmp;
			}
	}
}

void CalculVitesse(void){
	if(bitram.virage==0){
			tempvh += speed*1.8520/3.6;
	}else{
			tempvh += vh;
	}
	if(vc<0)vc=5;
}

void CalculPosTheorique(void){
	xth+=sin(route*pi/180)*vht/10;
	yth+=cos(route*pi/180)*vht/10;
}
```

```
void interruption (void);                       //Déclaration de la fonction pour les interruptions
void interruption_faible(void);     //Déclaration de la fonction pour les interruptions faible


#pragma code high_vector=0x08
void interrupt_at_high_vector(void)//Appel de la fonction interruption
{
        interruption();
}

#pragma code low_vector=0x18
void interrupt_at_low_vector(void)//Appel de la fonction interruption faible
{
        interruption_faible();
}



#pragma code /* return to the default code section */

#pragma interrupt interruption
void interruption (void){
        if(PIR1bits.TMR1IF){       //Renouveller la pulsation servomoteur à chaque débordement du timer1
                CCP2CON = 0;
                CCP2CON = 0b00001001;//CCP2 mode compare initialiser RC1 niveau haut et forcer au niveu bas sur une égalité
        }

        if(PIR1bits.CCP1IF){       //Interruption pour la lecture de l'angle boussole
                if(bitram.ouvert==1)led2 = !led2;
                bitram.bslok = 1;
                boussolenok = 0;        //RàZ compteur boussole non ok
                //lectBoussole();                       //lecture de l'angle

                boussole2 = boussole1;
                boussole1 = CCPR1;
                boussoletmp = boussole1 + debTMR1 * 0x10000;
                boussoletmp -= boussole2;//C350
                route = (boussoletmp/200 - 660);
                route=route-mvar;
                if((route>=360)|(route<0)) route=0;

                debTMR1 = 0;                            //RàZ compteur débordement timer1 entre deux pulsations
                PIR1bits.CCP1IF = 0;//Effacer CCP1IF
        }

        if(PIR1bits.TMR1IF){
                debTMR1++;                                              //Incrementation du compteur timer 1 pour mesure de langle boussole
                PIR1bits.TMR1IF = 0;//Effacer TMR1IF
        }

        if(INTCONbits.INT0IF){    //Interruption décollage
                bitram.decollage=1;    //Décollage pris en compte
                INTCONbits.INT0IF=0;//Effacer INT0IF
        }

        if(PIR3bits.TMR5IF){ //Interruption pour l'horloge après le décollage
                ADCON0bits.GO=1;
                if(ms100>=10){
                        vang=route-route2;
                        route2=route;
                        if(bitram.ouvert==1) para++;
                        while((vang<= -180)|(vang>180)){
                                if(vang<= -180) vang+=360;
                                if(vang>180) vang-=360;
                        }
                        corr=cap-route-2*vang;
                        while((corr<= -180)|(corr>180)){
                                if(corr<= -180) corr+=360;
                                if(corr>180) corr-=360;
                        }
                        if(bitram.pilotage==1){
                                servotemp+=(corr-vang)*1;//modification pour changer le sens du "+=" en "-="
                                if(servotemp<1800)servotemp=1800;
                                if(servotemp>4200)servotemp=4200;
                                servomoteur=servotemp;
                        }else servotemp=3000;
                        if(bitram.decollage==1) temps++;
                        boussolenok++;
                        gpsnok++;
                        ms100=0;
                }
```

```
                ms100++;
                PIR3bits.TMR5IF = 0;
        }
        if(PIR2bits.EEIF){
                EECON1bits.WREN=0;
                if(EEwrite<4){
                        EEADR=EEwrite;
                        EEDATA=EEbuf[EEwrite];
                        EECON1=0b00000100;
                        EECON2=0x55;
                        EECON2=0xAA;
                        EECON1bits.WR=1;
                        EEwrite++;
                }else if(EEwrite>10&&EEwrite<16){
                        EEADR=EEwrite-10;
                        EEDATA=EEbuf[EEwrite-14];
                        EECON1=0b00000100;
                        EECON2=0x55;
                        EECON2=0xAA;
                        EECON1bits.WR=1;
                        EEwrite++;
                }
                PIR2bits.EEIF=0;
        }
}

#pragma interruptlow interruption_faible
void interruption_faible(void){

        if(PIR1bits.RCIF){
                if(RCSTAbits.FERR) bitram.error=1;
                if(RCSTAbits.OERR) bitram.error=1;

                RCtemp=RCREG;
                if(bitram.rmc==0 & bitram.gsa==0){
                        if(RCtemp==0x24 & bitram.error==0){
                                gpsRC[0]=RCtemp;
                                RC=1;
                        }else if(RCtemp==0x0A & RC!=0 & bitram.error==0){
                                gpsRC[RC]=RCtemp;
                                RClen=RC+1;
                                if(gpsRC[1]==0x47 & gpsRC[2]==0x50 & gpsRC[3]==0x52 & gpsRC[4]==0x4d & gpsRC[5]==0x43){
                                        for(tr = 0;tr < 73;tr++)gpsRCRMC[tr]=gpsRC[tr];
                                        bitram.rmc=1;
                                }
                                if(gpsRC[1]==0x47 & gpsRC[2]==0x50 & gpsRC[3]==0x47 & gpsRC[4]==0x53 & gpsRC[5]==0x41){
                                        bitram.gsa=1;
                                }
                                RC=0;
                        }else if(RC!=0 & bitram.error==0){
                                gpsRC[RC]=RCtemp;
                                RC++;
                        }

                        if(bitram.error == 1) RC=0;
                        bitram.error=0;
                }

                PIR1bits.RCIF = 0;
        }

        if(PIR1bits.ADIF){
                pression=(ADRESL+ADRESH*0x100)/0.9216+950/9;
                altitude=9289.2-(9.17*pression);
                PIR1bits.ADIF = 0;
        }
}
```