Institut Supérieur de l'Aéronautique et de l'Espace

Issu du rapprochement SUPAERO et ENSICA

PROJET DE FIN D'ETUDES

SUJET

DIVING BIRD

Arnaud LABADIE

2014

FORMATION ENSICA

REMERCIEMENTS

Je remercie mon tuteur Luc JAULIN pour son aide. Il m'a permis de prendre du recul sur la façon de faire de l'automatique. Ca manière d'aborder la conception d'un contrôleur vient compléter la formation que j'ai eu. A côté de l'aspect purement matriciel et applicatif de mes cours j'ai pu observer l'importance de l'intuition dans les choix puis de la mise en pratique. Ses désirs pédagogiques et son savoir-faire mon beaucoup apportés.

Je remercie mon second tuteur Fabrice Le-BARS pour le partage de ses connaissances sur les robots marins ainsi que de la patience dont il fait preuve lors de la gestion de l'intendance. Son aide est essentielle au fonctionnement du laboratoire.

Je remercie Patrick ROUSSEAUX de l'IFREMER pour son aide lors de la construction de l'avion. Ses compétences pratiques et ses choix pragmatiques associé à sa grands expérience sont une source d'envie et montre le chemin qui me reste à parcourir et les savoirs faire à acquérir. Etre avec lui m'a donné envie de faire et de continuer à faire.

Je remercie Olivier MENAGE de l'IFREMER pour ses recommandations pratiques, son aide ainsi que pour ses conseils et méthodes sur la création de gros projet.

Je remercie Michel INIZAN pour son suivi et son désir de poursuivre ce projet. Il est devenu le réfèrent et s'est attelé à l'apprentissage du pilotage d'aéromodèles pour donner à l'ENSTA les compétences en Interne.

Je remercie Jean-Pierre KERNIN pour son aide précieuse. Sans lui rien n'aurait été démontré. Ses compétences et son expérience dans le pilotage d'aéromodèles m'ont permis de fournir des résultats à la fin de ce stage. Il est le principal atout pour la poursuite de projet sur des drones volants à voilure fixe.

Je remercie enfin les doctorants et les post-docs avec qui j'ai pu discuter. Ils m'ont permis de prendre conscience de nombreuse techniques et problématiques autour de leur projet.

Sommaire

Introduction

Ma recherche d'un stage pour mon PFE m'a amené à retourner à l'ENSTA Bretagne. Cette école dans laquelle j'ai effectué ma première année de formation d'ingénieur est connue notamment pour sa robotique marine avec des productions comme les robots vaimos, sauc'isse ou sardine.

Le stage qui m'a été proposé et qui porte le nom de « diving-bird » est à la fois ambitieux et exigeant. En effet, l'objectif est de faire un drone volant et sous-marin à la fois. Il doit donc pouvoir parcourir une certaine distance dans les airs, amerrir, plonger, faire une mission sous-marine, décoller et atterrir. Ce projet m'a attiré par sa complexité et par le savoir-faire que je pourrai en retirer une fois exécuté.

Cependant, l'objectif premier de ce stage est la volonté de mes tuteurs d'acquérir un savoir-faire dans un domaine en expansion, celui des drones volants pour, après, produire ce drone hybride. Hormis deux quadri-rotos, l'ENSTA Bretagne ne possède pas d'activités drones volants, notamment ceux à voilure fixe. Leurs tentatives se sont concentrées sur un planeur qui n'a jamais réussi à voler de par l'absence de pilote et par le manque de maitrise. Ainsi, l'objectif du stage s'est révélé plus complexe que prévu car il a fallu partir de rien. Les objectifs du stage se sont donc centrés sur :

- la création d'un aéronef à voilure fixe,
- la création de l'électronique,
- l'élaboration de la loi de commande,
- la recherche d'un pilote,
- le changement des mentalités.

Les difficultés viennent de plusieurs challenges qui se sont révélés à moi au fur et à mesure. La création de la structure est une première difficulté puisque je ne disposais d'aucune connaissance initiale en modélisme. En effet, un drone à cette échelle s'apparente à un avion de modélisme. Malgré la quantité de robots sous-marins développés à l'ENSTA Bretagne, je n'ai pas pu réutiliser leur électronique qui se base sur des minis ordinateurs portables associés à des centrales inertielles onéreuses et lourdes. Cela vient s'ajouter au fait que les grandes techniques d'automatisation robustes des drones volants reposent sur la modélisation mathématique de l'aéronef qui est compliqué à avoir.

Ainsi ce stage m'a fait partir de zéro.

I-Construction de prototype

Il est certain que l'achat d'un avion de modélisme assure l'utilisation rapide d'une plateforme saine pour tester nos algorithmes. L'expérience montre qu'entre les mains d'élèves s'improvisant pilotes, la durée de vie de tels appareils est très courte et le temps de vol n'excède pas les 10 secondes. Ainsi, les crashs qui surviennent, imposent l'achat de nouvelles pièces qui mettent du temps à arriver. Une solution peut être l'achat de « trainer » qui sont plus tolérants aux chutes et l'abandon de cette pratique qui consiste à laisser des élèves inexpérimentés piloter des avions sur un terrain non adapté et dans des conditions météorologiques qui ne sont pas adéquates.

Cependant, mon projet initial demande la construction d'une structure qui soit capable de voler, flotter et couler. Un modèle du commerce est difficile à modifier. Pour un sous-marin on cherche un ratio $\frac{poids}{poids\ du\ volume\ en\ eau}$ proche de 1. De ce fait, les structures faites en polystyrène ou en balsa ne conviennent pas à notre utilisation. Au vu du temps disponible, le design de cette structure est hors de ma portée. Mon objectif est donc de donner certains éléments qui leur permettront de dimensionner les parties principales du prototype.

Enfin, ce besoin vient d'une demande explicite émanant de mes tuteurs qui souhaitent acquérir certaines compétences en interne sur la réalisation d'avion ainsi que la publicité qu'apporte la construction d'un appareil fait en totalité en interne.

Ainsi, les besoins de construire un avion viennent donc :

-Du besoin d'avoir un avion rapidement disponible même en cas de crash,

-Du besoin futur de designer un prototype pouvant remplir l'objectif de l'intitulé de mon stage,

-Du désir de mes tuteurs d'acquérir certaines connaissances.

Ce travail a abouti à la construction de deux prototypes.

1 - Eléments de base

Les lignes qui suivent vont donner un aperçu des méthodes pour choisir les différents éléments qui composent un avion à échelle réduite. L'objectif n'est pas l'optimisation de chaque partie de l'appareil. L'optimisation globale de la plateforme nécessite plusieurs itérations et prend beaucoup de temps. Mon but est de construire rapidement une structure qui puisse voler simplement et qui soit relativement simple à faire. L'ensemble des problématiques qui sont liées à l'autonomie, la performance et l'esthétique ne sont pas prises en compte. De plus la puissance actuelle des moteurs électriques permet de s'octroyer certaines libertés qui simplifient les choix. On peut tout faire voler à partir du moment où l'on respecte 3 règles :

- disposer d'une puissance suffisante,
- disposer d'une forme orthodoxe,
- disposer d'un bon centrage.

L'objectif lors de la conception de l'appareil est de répondre aux attentes de mes tuteurs mais aussi d'avoir un avion qui soit simplement automatisable. Cela peut être fait en faisant en sorte que la cellule soit déjà stable. Par exemple un empennage octroie une certaine stabilité en tangage et lacet. De ce postulat, je suis parti sur la conception d'un avion classique. La conception d'une plateforme requiert le réglage et le choix des éléments suivants :

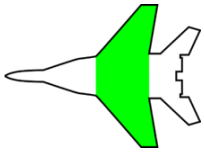
- Ailes
- Ensemble propulsif
- Centrage

1-1 Ailes

Le choix des ailes impose le choix de la forme, des dimensions et de la position. Ces choix vont permettre de compléter la stabilité et gérer la vitesse de l'avion. Pour que l'avion soit piloté simplement il faut qu'il aille lentement. Cela peut être réglé en choisissant une charge alaire faible. Cette condition peut être observée en regardant la formule du vol en pallier :

$$P = \frac{1}{2}\rho S V^2 Cl$$

Avec :



- S qui représente la surface Alaire (en vert)

- V la vitesse de l'avion

- P le poids

Les abaques existant en aéromodélisme indiquent que pour un avion Outdoor d'envergure comprise entre 1m et 1.6m la charge alaire doit se situer entre 20 et 50g/dm². Grace à cette formule on peut voir qu'à P fixé, si S augmente alors la vitesse V diminue. Pour calculer les dimensions de l'aile, envergure et variation de la corde, à partir de la surface il faut choisir sa géométrie. Bien que des formes comme les ailes en flèche ou elliptique possèdent des avantages notables comme la diminution de la trainée, la simplicité de construction a été mise en avant. Ce sont donc des ailes rectangulaires qui sont choisies. Les profils utilisés pour la création des ailes dépendent des modèles et seront donnés dans la partie « prototypes » de ce mémoire.

Finalement, la position des ailes, hautes et un dièdre ont été choisis pour renforcer la stabilité en roulis.

1-2 Ensemble propulsif

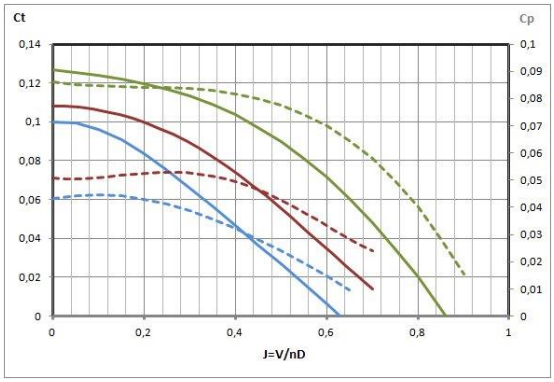
Par soucis de simplicité, une motorisation électrique a été choisie car elle ne demande aucun entretien et elle est simple à mettre en œuvre. En modélisme, les moteurs électriques brushless sont majoritairement utilisés et sont donc ceux sélectionnés. Le dimensionnement de l'ensemble propulsif demande de choisir une hélice, un moteur, des batteries et un contrôleur brushless.

Les deux caractéristiques principales de l'hélice sont : - Le diamètre

- Le pas

Le diamètre influence la traction maximale et le pas a un effet sur la vitesse maximale. Ces paramètres sont choisis pour offrir le meilleur rendement de l'hélice à une vitesse de croisière donnée. Il n'y a pas de formule exacte pour choisir l'hélice. En effet la force de traction, la puissance et le rendement dépendent de l'avance :

- Avance : $Av = \frac{V}{N*D}$ avec V Vitesse de l'avion
- Coefficient de traction : $Ct = \frac{Ft}{\rho*D^4*N^2}$ avec $Ct = f(Av)$
- Coefficient de puissance : $Cp = \frac{Pt}{\rho*D^5*N^3}$ avec $Cp = h(Av)$
- Rendement : $\eta = \frac{Ct*Av}{Cp}$



Les caractéristiques principales du moteur sont :

- Kv : nombre de tours/volt
- Pmax : puissance maximale
- la tension d'alimentation

La puissance du moteur est classiquement calculée en fonction des besoins lors du décollage. Par sécurité et simplicité le moteur a été surdimensionné pour offrir au pilote la possibilité de redémarrer vite en cas de problème. En effet lors du premier vol et pour une première construction, on ne sait pas comment l'avion va réagir et en cas de bourrasque l'avion peut décrocher. Ce surplus de puissance, bien qu'entrainant un surplus de masse est rassurant et très utile. Pour avoir un ordre d'idée de la puissance qu'il nous faut, l'utilisation des formules des modélistes est faite. Même si on dispose des équations du moteur et de l'hélice, l'évaluation des paramètres tout au long de la chaine de transmission de puissance est laborieuse et inutile puisque des estimations basées sur l'expérience des modélistes sont efficaces. Les équations du moteur sont données à titre d'indication :

Les équations du moteur sont :

-Vitesse de rotation du moteur : $N = Kv * (U - I * Ri)$

-Tension utile : $U_{utile} = U - I * Ri$

-Puissance consommée : $Pc = U * I$

-Puissance en sortie moteur : $Ps = (U - I * Ri) * (I - Io)$.

-Puissance en entrée du rotor : $Pr = N * C$ (ou C est le couple au niveau du rotor).

Les caractéristiques principales des batteries LiPo sont :

-La capacité (mAh ou Wh)

-Le nombre de cellules

-Le taux de décharge (x C)

La batterie fournit l'énergie aux moteurs et aux actionneurs. Sa capacité donne un ordre d'idée sur l'autonomie. Le nombre de cellules donne un ordre de grandeur de la tension maximale en entrée du moteur qui est liée à la vitesse de rotation maximum. Enfin le taux de décharge indique l'intensité à ne pas dépasser lorsque que l'on débite du courant pour éviter la détérioration de la batterie.

Par exemple pour une batterie :

-2400maH (capacité. Ici 2.4Amperes en 1heure)

On a :

-La tension chargée max est 4*4.2 = 16.8V

-4S (nombre de cellules qui fournissent la tension)

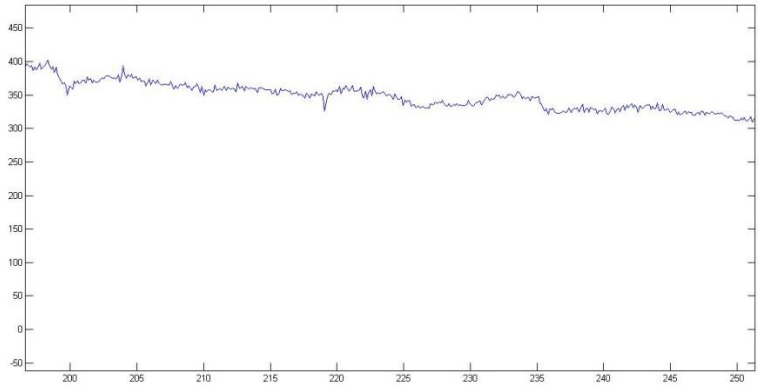
-20C (taux de décharge maximum. Ici 20*2.400 = 48A max)

-La tension minimum limite à ne pas dépasser 4*3.7 = 14.6V

-Capacité totale 2.4A * 14.6 = 35.5Wh

Si l'avion utilise en moyenne 200Wh alors on a une autonomie théorique de 60*35.5/200= 10minutes

Les batteries LiPo présentent l'avantage de contenir une grosse puissance pour une masse réduite. Cependant il faut veiller à ne pas les décharger en dessous de 3.7V par cellule. Il faut donc soit prévoir un dispositif de mesure de la tension, soit veiller à ce que le temps d'utilisation ne dépasse pas une limite que l'on s'est fixée. Il est à noter que l'utilisation des batteries neuves est recommandée. Des tests effectués avec un ampère mètre à effet Hall ont permis de mettre en évidence une variation d'intensité délivrée selon l'état de la batterie. On remarque aussi une chute de tension lorsque l'on tire de l'énergie.



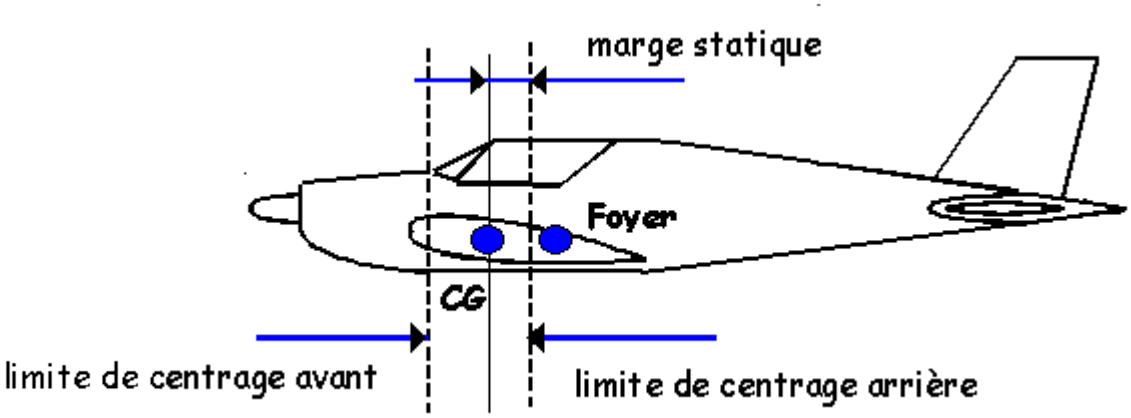
Chute de puissance progressive lors de l'utilisation de la batterie

Finalement, la stratégie adoptée est de choisir en premier la puissance du moteur avec $Pmax \approx 350W/kg * Poids$ et un $Kv \approx 1000$. Les batteries choisies sont des batteries LiPo 4S 2400mah car déjà disponibles dans le laboratoire. Avec ces 3 éléments que sont la puissance maximum, le Kv et la tension en entrée disponible grâce aux batteries il est possible d'acheter le moteur. Enfin le choix de l'hélice s'est fait selon les recommandations du fabricant. En effet, si le diamètre ou le pas de l'hélice sont trop importants par rapport aux recommandations, le couple supplémentaire induit diminu fortement la vitesse de rotation du moteur. Or comme on peut le voir dans les équations, et aux vues des vitesses de rotations auxquelles nous sommes, un diamètre ou un pas trop grands n'apportent pas forcement le gain de traction ou de vitesse espéré car cela réduit la vitesse de rotation.

Finalement, la puissance maximum du moteur et la tension de la batterie permettent de choisir le contrôleur brushless en fonction de l'intensité maximale. Ce contrôleur transforme un signal PWM qui est la commande en puissance du moteur en tension triphasée. De nombreux logiciels existent pour affiner l'ensemble des calculs. Les constructeurs de moteur mettent souvent en lignes des applications qui aident à choisir sa motorisation. Il est aussi possible de trouver les abaques des hélices.

1-3 Centrage

Lorsque l'on souhaite mettre en place la charge utile et joindre l'aile au fuselage il faut veiller au respect de la position du centre de masse par rapport au foyer.



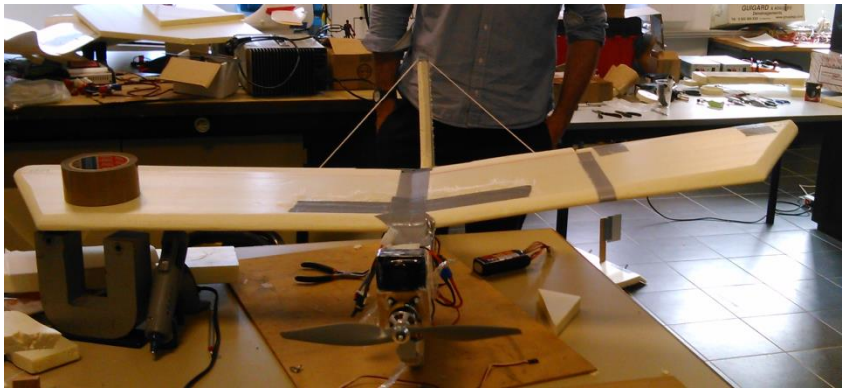
La position du foyer peut être calculée avec des logiciels comme PredimRc. Cependant, l'utilisation d'une aile rectangulaire permet d'utiliser une astuce de modéliste et une connaissance communément admise qui est que le foyer se trouve aux alentours des 25% de la corde de l'aile. Selon le type d'avion la position du foyer est comprise entre 20 et 30% de la corde. Par sécurité lors d'une première mise en œuvre de l'appareil, il vaut mieux avoir un centrage très en avant.

2 – Les prototypes

Les éléments décrits ci-dessus ont permis d'élaborer les deux prototypes décrits ci-dessous. Ces deux prototypes répondent à des problématiques différentes. L'optimisation n'est pas l'objectif. Le coté fait maison cher à mes tuteurs est un vecteur de publicité et d'éducation. En effet, l'une des doctrines mise en avant est l'apprentissage par l'expérience. Les travaux effectués permettent de poser les jalons des travaux futurs en créant un précèdent dans le laboratoire. Cependant, mon avis reste d'acheter un modèle du commerce pour partir sur une base saine.

2-1 Premier prototype

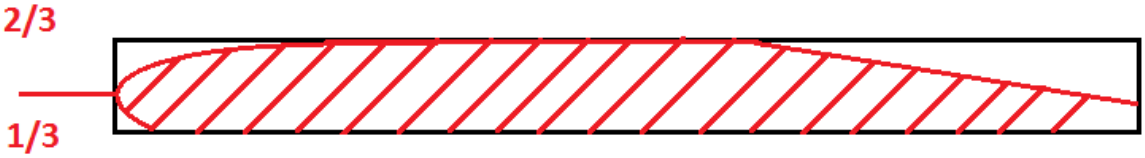
Le premier avion permet de voir qu'avec peu de moyen en en respectant les 3 points énoncés en introduction on peut faire voler un avion. Ce premier montage a pour but d'être très vite fait et peu onéreux. En effet, l'objectif est la simplicité de fabrication. Il suffit d'une planche de polystyrène extrudé de 1250*600*20, de colle, de scotch et de moins d'une demi-journée de travail pour obtenir ce prototype.



Premier démonstrateur

La fabrication de ces ailes est très simple mais doit respecter une règle, il faut créer un bord d'attaque. En effet, un bord d'attaque brut engendre une vitesse du flux d'air entre l'intrados et l'extrados très forte qui fait décoller le filet d'air et induit un décrochage rapide. Mais une aile n'a pas besoin d'avoir un beau profil pour pourvoir sustenter l'avion. Il faut donc apporter quelques modifications très simples à la plaque :

- Arrondir le bord d'attaque.
- tailler le bord de fuite



Cette aile est rapide à faire et convient pour ce prototype. De plus n'importe quel élève peut en faire une sans savoir-faire en bricolage. La corde permet de rigidifier l'aile puisque l'épaisseur de la plaque est faible.

Vol du premier prototype par un pilote très expérimenté

La fabrication des surfaces mobiles s'est fait de façon artisanale. Lors de ma recherche je n'ai pas été capable de trouver d'informations sur le dimensionnement des surfaces mobiles. Cependant, l'intuition permet de comprendre que plus la surface mobile est loin du centre de gravité plus elle a un

effet. De même, plus sa surface est grande plus son effet est important tant au niveau des rotations que des frottements.

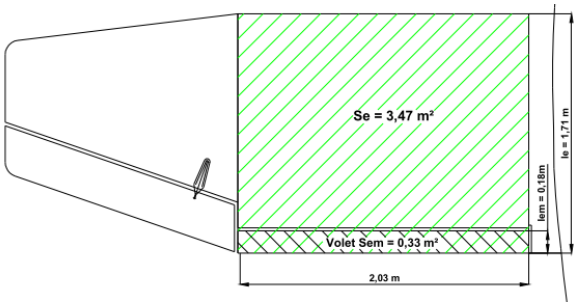
Après cela il faut choisir les servomoteurs. Les Servomoteurs qui actionnent les surfaces de contrôle doivent être dimensionnés en fonction de l'effort que subissent les gouvernes. Pour cela on regarde surtout le moment charnière.

La formule de Mr Toussaint donne :

$$Mc = \frac{r*Ve^2*Cmc*Sem*Iem}{2*g}$$

$$Cmc = \left(0.25 * Ae * \sqrt{S}\right) * ie + \left(0.25 * Ae - 0.04(1 - S)\right) * b$$

$$Ae = \frac{0.095*L}{L+1.73} \quad S = \frac{Sem}{Se}$$



Avec :

- Mc : moment charnière en m.kg
- ie : incidence de l'aile en °
- r : densité de l'aire (1.225kg/m3)
- g = 9.81m/s²
- Sem : Surface du volet en m²
- Iem : Corde moyenne du volet en m
- b: angle de braquage du volet en °
- L : allongement du volet
- Se : surface de l'aile avant le volet

En combinant cette formule et en prenant exemple sur les modèles d'avions qui sont dans le commerce on peut prendre des servomoteurs ayant un poids de l'ordre de 20g. Ces servomoteurs possèdent un couple autour des 2.5kg/cm et sont déjà présents au laboratoire.

2-2 Second Prototype

Le second prototype démontre l'évolution des objectifs. Il est créé pour pourvoir embarquer plus de charge utile, voler lentement et être plus résistant. Dans le cas d'un crash détruisant les ailes, une seconde paire a été produite. Les ailes ont été construites en premier, puis une charge utile de 300g a été définie, enfin en considérant le poids du fuselage des batteries et une estimation du poids du moteur, on a estimé le poids total de l'appareil. Ce poids a permis d'évaluer la puissance du moteur comme décrit précédemment.

Les ailes utilisent un profil Clark Y qui est l'un des plus communs dans le monde du modélisme. Avec une longueur de 160mm et une corde de 30cm ces ailes permettent de voler lentement grâce à la portance que donnent à la fois le profil et la surface. La technique de fabrication est très simple et se trouve sur internet. Elle nécessite peu de matériel et deux personnes et reste à la portée d'élève. Elles ont été découpées dans un bloc de polystyrène extrudé à l'aide d'un fil chaud. Le dièdre est fait à l'aide de tissu en fibre de verre et rigidifié avec de l'époxy. Elles sont détachables pour permettre d'avoir accès à l'intérieur du fuselage pour ranger la charge utile et le matériel nécessaire à la propulsion de l'appareil.

Avion n°2, prêt à voler

Le fuselage a été donné par un modéliste. Il a nécessité l'ajout de l'empennage, de la dérive et du support moteur. Le temps gagné par l'utilisation de ce fuselage a généré de légers problèmes vite surmontés. En effet, il était prévu pour un moteur thermique, le poids de ce dernier permettait d'avoir un centrage avant. Ainsi, même en positionnant le moteur électrique et la batterie le plus en avant, l'ajout de plomb est nécessaire et est venu alourdir l'avion.

Vol sous le ciel nuageux du Finistère

La construction manuelle et mon inexpérience sont à l'origine de nombreux défauts. Il a donc fallu plusieurs vols pour pouvoir les trouver et les compenser.

II-Modélisation d'un avion

Cette partie a pour objectif de donner les équations qui permettent d'approximer au mieux le comportement d'un aéronef. La modélisation qui va être présentée prend en compte :

-la dynamique d'un corps solide,

-sa cinématique,

-la modélisation des forces aérodynamiques.

La complexité des phénomènes aérodynamiques ne peut pas être entièrement mise en équation et l'on se contente souvent de rester autour d'un point d'équilibre de fonctionnement de l'avion. En effet, les équations qui sont fournies ne prennent pas en compte les différents types de décrochages. Il faut donc en tenir compte et s'assurer que l'avion reste dans une enveloppe de vol sure. De plus, il faut aussi prendre en compte les contraintes mécaniques de l'avion.

Une fois que le modèle mathématique est fixé, il est nécessaire de trouver les coefficients qui vont caractériser le support que l'on utilise. La complexité de la forme de l'avion (Aile, profil, fuselage, empennage) est en grande partie retranscrite par ces coefficients. Ainsi, on peut changer de modèle en changeant ces coefficients. L'objectif est donc de trouver ses coefficients par des méthodes numériques, empiriques ou par identification. Une fois les coefficients déterminés, on peut utiliser ce modèle pour générer les lois de commande qui viendront réguler notre avion et simuler son comportement.

Les besoins d'un simulateur viennent de :

-La demande de mon tuteur,

-Le besoin d'avoir un ordre de grandeur pour les coefficients du contrôleur.

A partir du modèle présenté, deux simulateurs ont été réalisés. L'un en Matlab pour mes propres tests, et un second en Qt pour mon tuteur.

1- Simulateurs

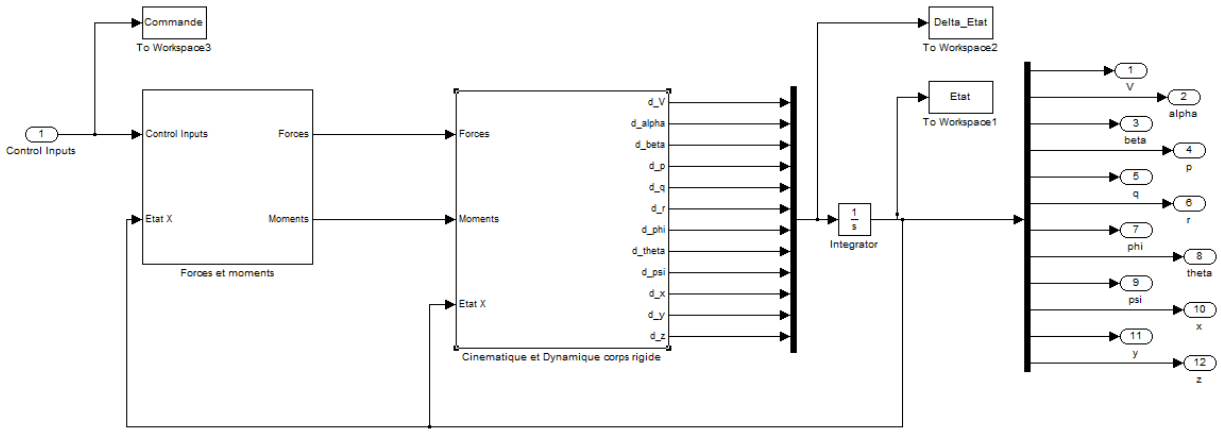
Les simulateurs utilisent principalement deux méthodes pour reproduire le comportement d'un système :

-L'utilisation de formules mathématiques.

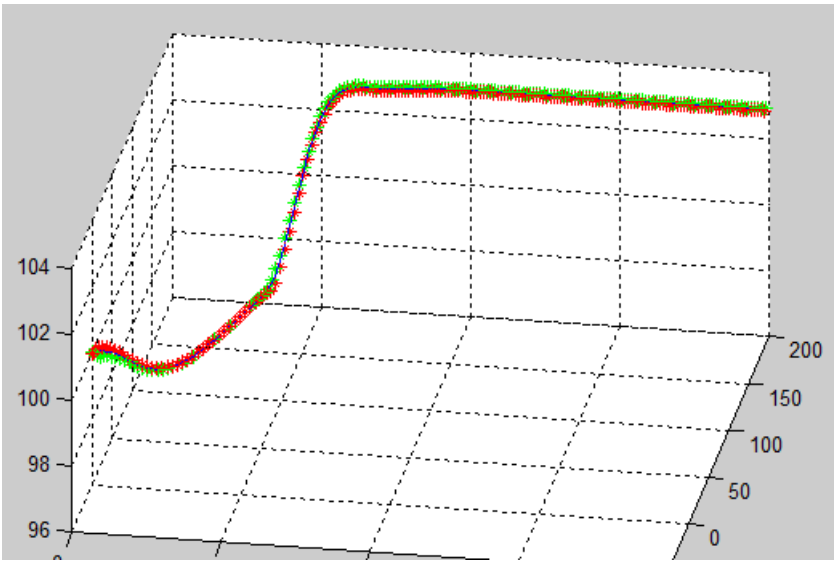
-L'utilisation d'un ensemble de données mesurées lors de test.

Mes simulateurs reposent sur une expression mathématique censée approximer au mieux le comportement du système à simuler. C'est la qualité de cette représentation mathématique qui peut être aussi utilisée lors de la conception de la loi de commande qui va définir la qualité du simulateur ainsi que la méthode d'intégration. Ces modélisations de système se basent sur des coefficients qui sont caractéristiques d'un système en particulier. Dans la suite de cette partie, je vais rappeler les équations qui permettent de décrire la physique du comportement d'un aéronef ainsi que les coefficients qui permettent de les différencier.

Pour caractériser l'évolution d'un aéronef dans l'espace, il nous faut les équations d'évolution des variables de position et des variables d'orientation. A partir de ces expressions, des méthodes comme l'intégration par la méthode d'Euler peuvent être utilisées. Cette méthode est simple à mettre en œuvre. Ce simulateur est là pour la compréhension du mouvement, la précision et la qualité ne sont pas les objectifs.

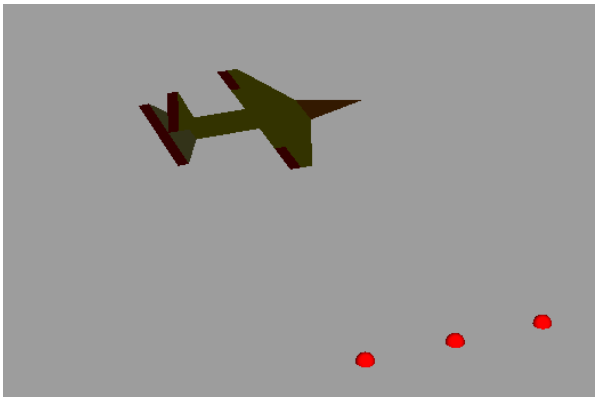


Calcul des forces et moments puis utilisations des équations mécaniques pour un corps rigide

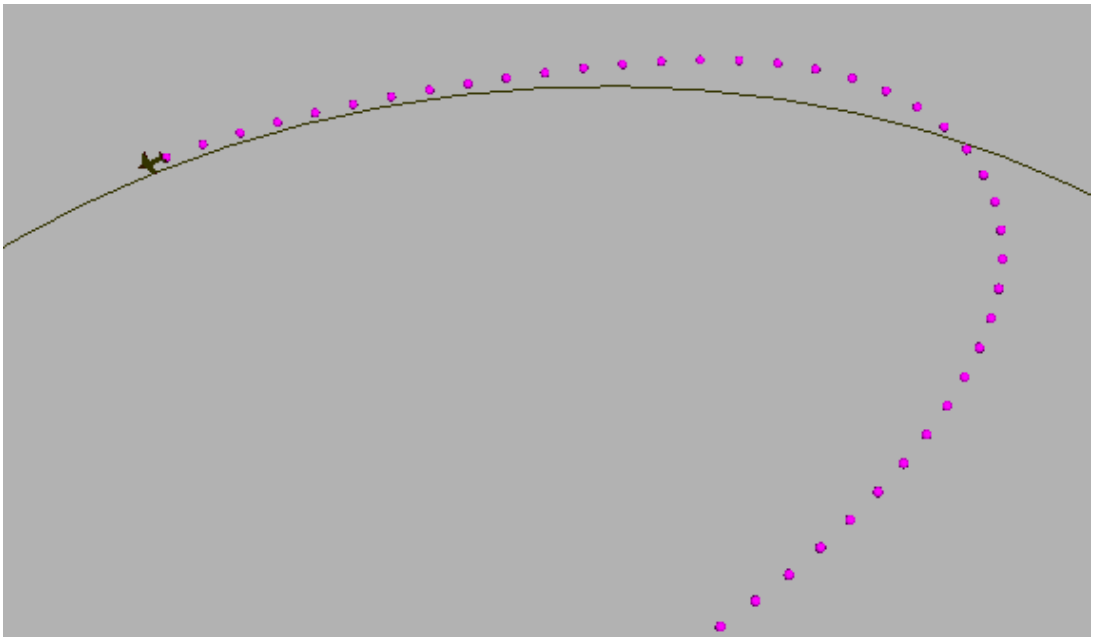


Affichage d'une trajectoire après la modification des consignes

Les coefficients qui caractérisent un aéronef sont les coefficients des forces et moments aérodynamiques. L'obtention de ces coefficients et leurs expressions sont développés dans la suite de ce mémoire. Ces coefficients sont répertoriés dans des tables de données aérodynamiques ou dans des abaques. Une représentation linéaire des coefficients n'est valable qu'autour d'une position d'équilibre. Ainsi à chaque configuration de vol que l'aéronef est susceptible de prendre, il est nécessaire d'obtenir la valeur de ces coefficients et d'en donner une plage de validité. Cependant, au vue de la difficulté pour obtenir ces coefficients, les simulateurs développés ne permettent de simuler que des situations de vol en pallier avec un angle d'attaque faible. Il est possible d'imposer à l'avion des comportements acrobatiques en braquant complètement les surfaces de contrôles. Cependant, le comportement ainsi obtenu s'éloigne du bon.



Représentation graphique simple d'un avion avec 4 surfaces de contrôle



Modélisation d'une trajectoire circulaire

Certains laboratoires donnent les coefficients aérodynamiques de l'avion qu'ils contrôlent. Les coefficients que je cherche sont ceux d'un avion de modélisme d'une envergure d'environ 1.5m et d'un poids autour de 2kg. L'université du Minnesota dispose de ces coefficients pour leur drone qui reposent sur une plateforme Faser ultra Stick relativement proche de la mienne et les met en ligne. Pour des projets plus ambitieux, les données aérodynamiques du F16 sont en grandes parties accessibles, par exemple.

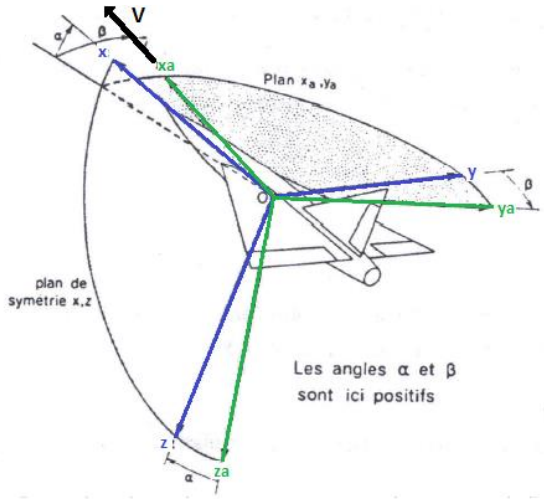
2- Equations dynamique et cinématique

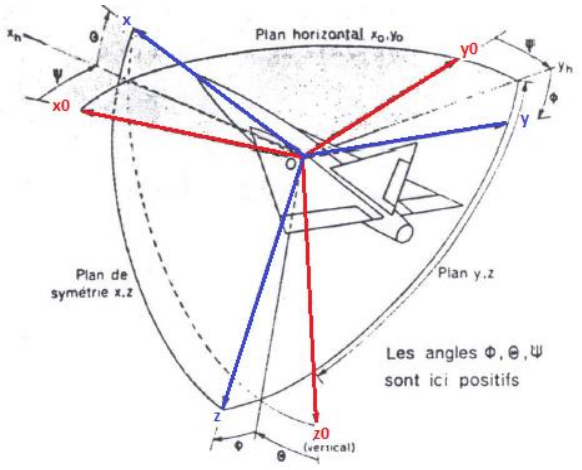
Cette partie est un résumé rapide de la modélisation dynamique et cinématique d'un avion. Elle a pour but de donner les conventions et les hypothèses sous lesquels les équations qui suivent sont établies et non de les redémontrer. Pour ceux désirant obtenir plus de précisions, se référer à l'ouvrage de THOR I. FOSSEN.

1-1 Les repères :

La mécanique du vol utilise 3 repères pour décrire le comportement de l'avion et les forces aérodynamiques :

- **Repère Normal Terrestre : $Ox_0y_0z_0$** repère NED, x_0 pointe vers le nord, y_0 vers l'Est, z_0 vers le bas
- **Repère Avion: Gxyz**
- **Repère Aérodynamique : $Gx_ay_az_a$**





1-2 Equations dynamiques :

La matrice d'inertie dans le repère Avion est : $I = \begin{bmatrix} Ix & 0 & -Ixz \\ 0 & Iy & 0 \\ -Ixz & 0 & Iz \end{bmatrix}$ On suppose que l'avion est symétrique.

Les taux de rotation dans le repère Avion : $\vec{\omega} = \begin{pmatrix} p \\ q \\ r \end{pmatrix}$

La vitesse inertielle de l'avion dans le repère avion : $\vec{V} = \begin{pmatrix} u \\ v \\ w \end{pmatrix}$

Les forces appliquées à l'avion dans le repère avion : $\vec{F} = \begin{pmatrix} X \\ Y \\ Z \end{pmatrix}$ sans le poids

Les moments appliqués au centre de gravité de l'avion dans le repère avion : $\vec{F} = \begin{pmatrix} L \\ M \\ N \end{pmatrix}$

On suppose que l'axe du moteur passe par le centre de gravité d'où $\vec{F}m = \left(\begin{array}{c|c} Xm & Lm \\ 0 & 0 \\ 0 & 0 \end{array}\right)$

Equations des forces

$$X = m(\dot{u} + qw - rv + gsin(\theta))$$

$$Y = m(\dot{v} + ur - wp - gcos(\theta)\sin(\phi))$$

$$Z = m(\dot{w} + vp - qu - \dot{g}cos(\theta)\cos(\phi))$$

Equations des moments

$$L = \dot{p}Ix - Ixz(\dot{r} + pq) + (Ix - Iy)qr$$

$$M = \dot{q}Iy - Ixz(p^2 + r^2) + (Ix - Iz)pr$$

$$N = \dot{r}Iz - \dot{p}Ixz + (Iy - Ix)pq + qrIxz$$

Cinématique angulaire

$$\dot{\phi} = p + qsin\phi tan\theta + rcos\phi tan\theta$$

$$\dot{\theta} = qcos\phi - rsin\phi$$

$$\dot{\psi} = q\frac{sin\phi}{cos\theta} + r\frac{cos\phi}{cos\theta}$$

Equation Cinématique

$$\begin{pmatrix} \dot{pn} \\ \dot{pe} \\ \dot{pd} \end{pmatrix} = \begin{pmatrix} c\theta c\psi & s\phi s\theta c\psi - c\phi s\psi & c\phi s\theta c\psi + s\phi s\psi \\ c\theta s\psi & s\phi s\theta s\psi + c\phi c\psi & c\phi s\theta s\psi - s\phi c\psi \\ -s\theta & s\phi c\theta & c\phi c\theta \end{pmatrix} \begin{pmatrix} u \\ v \\ w \end{pmatrix}$$

Les forces représentées par les lettres X Y Z sont les forces aérodynamiques dans le repère avion. De même, les moments L M N sont les moments aérodynamiques dans le repère avion. Les forces aérodynamiques dans le repère aérodynamique ont pour expressions $Fi = \frac{1}{2}\rho SV^2Ci$ ou ci est le coefficient aérodynamique. Pour obtenir les forces dans le repère avion il faut utiliser une rotation. Cependant cela dépend de la façon dont les Ci ont été évalués. La modélisation et l'obtention de ses coefficients sont décrites dans les parties suivantes.

3- Modélisation des coefficients aérodynamiques

Les forces et moments aérodynamiques font intervenir les coefficients aérodynamiques. Ce sont ces coefficients qui permettent de caractériser le comportement de l'aéronef. Ils traduisent mathématiquement et avec la précision que le modèle mathématique nous permet, les phénomènes aérodynamiques.

Ces phénomènes aérodynamiques sont des phénomènes non linéaires. Etablir une équation est compliqué et nous éloigne déjà de la réalité. Cependant certaines formes d'équations sont communément admises et sont utilisées pour des systèmes comme les avions ou les missiles. En considérant un avion avec les surfaces de contrôles suivantes :

- 2 Ailerons
- Gouverne de profondeur
- Gouverne de direction

Une expression courante des coefficients aérodynamiques est :

Pour les coefficients des forces aérodynamiques :

$$C_L = C_{L0} + \alpha\, C_{L\alpha} + \dot{\alpha}\, C_{L\alpha dot} + \frac{b}{2V}\, q\, C_{Lq} + \delta_{elevator} C_{Lelev} + \delta_{l\text{-}aileron} C_{Ll\text{-}ail} + \delta_{r\text{-}aileron} C_{Lr\text{-}ail}$$

$$C_D = C_{Dmin} + \frac{S}{\pi b^2 osw}(C_L - C_{Lmin})^2 + |\delta_{elevator}|\, C_{Delev} + |\delta_{l\text{-}aileron}|\, C_{Dl\text{-}ail} + |\delta_{r\text{-}aileron}|\, C_{Dr\text{-}ail} + |\delta_{rudder}|\, C_{Drud}$$

$$C_Y = \beta\, C_{Y\beta} + \frac{b}{2V}\,(p\, C_{Yp} + r\, C_{Yr}) + \delta_{rudder} C_{Yrudder}$$

Pour les coefficients des moments aérodynamiques :

$$C_l = \beta\, C_{l\beta} + \frac{b}{2V}\,(p\, C_{lp} + r\, C_{lr}) + \delta_{rudder} C_{lrudder} + \delta_{l\text{-}aileron} C_{ll\text{-}ail} + \delta_{r\text{-}aileron} C_{lr\text{-}ail}$$

$$C_m = C_{m0} + \alpha\, C_{m\alpha} + \frac{b}{2V}\, q\, C_{mq} + \delta_{elevator} C_{melev} + \delta_{l\text{-}aileron} C_{ml\text{-}ail} + \delta_{r\text{-}aileron} C_{mr\text{-}ail}$$

$$C_n = \beta\, C_{n\beta} + \frac{b}{2V}\,(p\, C_{np} + r\, C_{nr}) + \delta_{rudder} C_{lrudder} + \delta_{l\text{-}aileron} C_{ll\text{-}ail} + \delta_{r\text{-}aileron} C_{lr\text{-}ail}$$

4- Obtention des coefficients aérodynamiques

Les coefficients aérodynamiques permettent de saisir les comportements les plus classiques de l'avion comme :

-le roulis induit
-le lacet induit
-une stabilité en lacet
-une stabilité en tangage

Ainsi, chaque aéronef peut être étudié et simulé grâce aux équations ci-dessus. L'objectif est donc d'obtenir ces coefficients aérodynamiques pour l'aéronef en question. Cet aéronef pouvant être un missile ou un avion par exemple. Il est important de noter que ces coefficients ne sont valides qu'autour de la position d'équilibre autour de laquelle ils ont été calculés. Ainsi, il faut fixer des plages de validité des coefficients autour notamment de plages de variations de l'angle d'attaque. Dès que l'on sort d'une de ces plages de variations, les coefficients changent. Cependant, pour un aéronef classique qui ne fait pas « d'acrobatie » la variation de l'angle d'attaque est faible. En vol en palier on a donc un seul set de coefficients. Il existe plusieurs méthodes pour obtenir ces coefficients. Tout dépend du coefficient et de la marge d'erreur. En effet les quatre grandes méthodes que nous ayons à disposition sont :

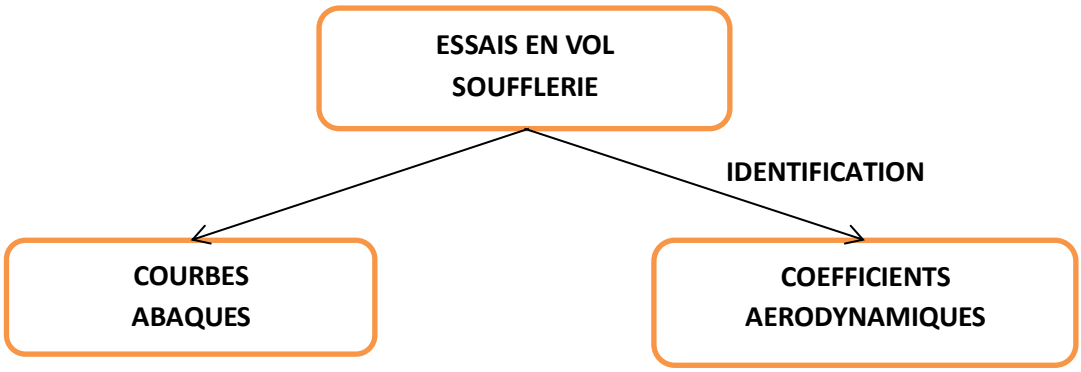
-L'estimation
-L'indentification
-Le calcul numérique
-L'utilisation d'abaque

L'estimation peut permettre un accès rapide à certains coefficients. Cette estimation passe par l'utilisation de formule empirique ou bien par une mise à l'échelle par multiplication par un facteur à déterminer. Des formules comme les formules de Scheider ou l'utilisation des données mises à disposition par la DATCOM permettent d'effectuer ce genre calcul.

L'utilisation des logiciels de calcul de mécanique des fluides permet aussi d'obtenir certains coefficients. Cependant la complexité du phénomène physique ne permet pas d'obtenir une évaluation précise des coefficients. De plus tous les coefficients ne peuvent pas être évalués. Cette méthode peut être plus précise que l'estimation dans certains cas.

L'utilisation d'abaques donne un accès direct au coefficient Cα. Plus particulièrement, les essais des profils NACA sont faits à échelle réduite car plus pratique. L'utilisation des abaques pour des avions à taille réelle nécessite donc une mise à l'échelle qui n'est pas nécessaire pour les mini drones.

L'élaboration d'une formule mathématique pour les coefficients aérodynamiques offre la possibilité de faire de l'identification. Pour plus de précision se référer à l'ouvrage de Klein et Morelli, Aircraft System Identification. Pour constituer les abaques ou faire de l'identification il est nécessaire de recueillir un ensemble de donner.



Dans le cas de l'identification il est souvent nécessaire d'avoir recours à plusieurs des solutions décrites précédemment. En effet, elles ne permettent pas d'avoir accès aux mêmes coefficients. Les essais statiques donnent accès aux coefficients statiques. Les essais dynamiques donnent accès aux coefficients dynamiques.

SOUFFLERIE

Obtention des coefficients Statiques dépendants de :

$$\alpha\ \beta\ \delta_{aileron}\ \delta_{rudder}\ \delta_{elevator}\ \delta_{flap}$$

ESSAIS EN VOL

Obtention des coefficients Dynamiques dépendants de :

$$\dot{\alpha}\ \dot{\beta}\ p\ q\ r$$

Les abaques offrent l'avantage d'être les plus proches de la réalité. L'identification offre l'avantage d'avoir accès à une formule mathématique pour l'établissement des équations d'état.

Ces types de moyens (soufflerie, essai en vol) ne sont pas à ma portée. En effet, l'accès à une soufflerie est onéreux et jusqu'à une période avancée de mon stage je ne disposais toujours pas de pilote. Pour pouvoir produire un simulateur j'ai donc utilisé l'expression linéarisée des coefficients aérodynamiques décrites ci-dessus. Les valeurs de ces coefficients sont issues des campagnes de test de l'université du Minnesota autour d'un modèle à voilure fixe, le faser ultra stick.

Faser ultra stick de l'université du Minnesota

Ordre de grandeur des coefficients Aérodynamiques Linéaires Dans le repère Aérodynamiques pour un modèle FASER

CL0 = 0.1086, CLalpha = 4.58, CLdail = 0.4751, CLdelev = 0.0983, CLαdot = 1.9724; CLq = 6.1639;

CDwmin = 0.0434; CDwdflap = 0.1467; CDwdelev = 0.0135, CDwdail = 0.0302; CDwdrud = 0.0303;

osw = 0.75; CLminD = 0.23;

CYwbeta = -.4889; CYwdail = 0; CYwdrud = 0.1913; CYwp = -0.0375; CYwr = 0.1500;

Clbeta = -0.0545; Cldail = -0.1646; Cldflap = -0.0692; Cldrud = 0.0115; Clp = -0.4496; Clr = 0.1086;

Cmzero = -0.0278; Cmalpha = -0.7230; Cmdflap = 0.0467; Cmdail = 0.0467; Cmdelev = -0.8488;

Cmq = -13.5664;

Cnbeta = 0.0723; Cndflap = 0.0214; Cndail = 0.0574; Cndrud = -0.1811; Cnp = 0.1180; Cnr = -0.1833;

Si on veut faire de l'identification, les modes rapides d'un avion à cette échelle ont une fréquence de l'ordre de 30Hz. Ainsi, l'acquisition des données devra se faire à une fréquence de 60 Hz minimum.

III-Contrôle de l'avion

De nombreuses méthodes pour automatiser les aéronefs existent. Une liste non exhaustive mais dont on peut trouver de nombreux exemples dans la littérature scientifiques sont :

- Gain scheduling associé aux méthodes LPV, H∞ etc,
- Backsteping
- PID
- Commande adaptative
- Réseaux de neurones
- Inversion de dynamique
- Fuzzy Control

Les protocoles de réglages et de dimensionnement des différents contrôleurs se basent sur la modélisation du système à automatiser. Cette modélisation qui repose sur le modèle mathématique connu et dont une forme a été décrite précédemment s'appuie sur l'identification des coefficients aérodynamiques qui caractérisent l'aéronef contrôlé autour de la position voulue. Cette identification n'a pas pu être faite, d'une part, par l'absence d'un pilote tôt dans le stage et d'autre part car la précision n'est pas un objectif de la commande. Il a donc fallu trouver une méthode qui ne nécessite pas l'utilisation et la connaissance d'une modélisation précise de l'aéronef.

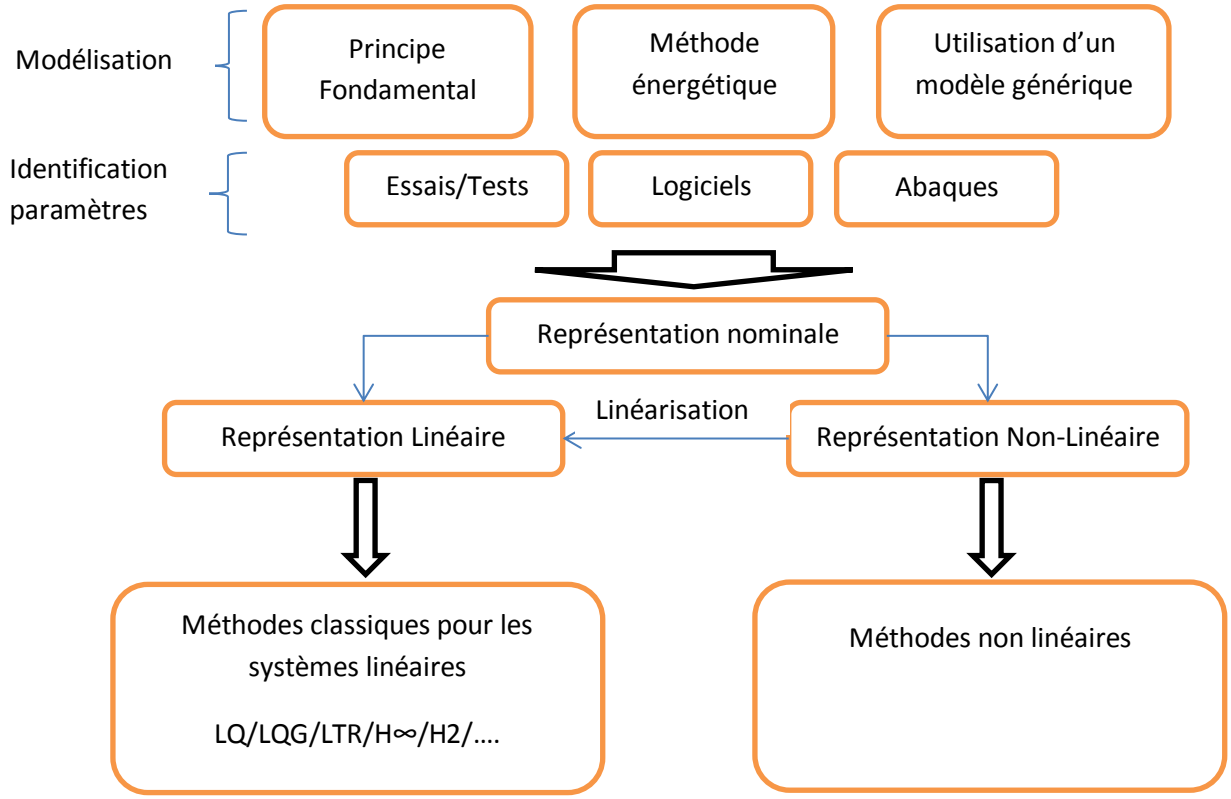
Pour certains systèmes sans contrôleur qui sont pilotés par l'homme il est possible d'établir une commande qui reproduit la réaction humaine. Cette commande passe par l'utilisation d'un contrôleur PID. Ce contrôleur est simple car on peut le régler empiriquement ou par l'utilisation de méthodes qui existent déjà. Cependant, l'ordre de grandeur et les premiers réglages des gains ce sont faits à l'aide du simulateur, puis ils ont été ajustés lors des tests. Autour de cette méthode, plusieurs objectifs m'ont été donnés :

- Vol en palier
- Virage en palier
- Suivi de ligne
- Décollage
- Atterrissage

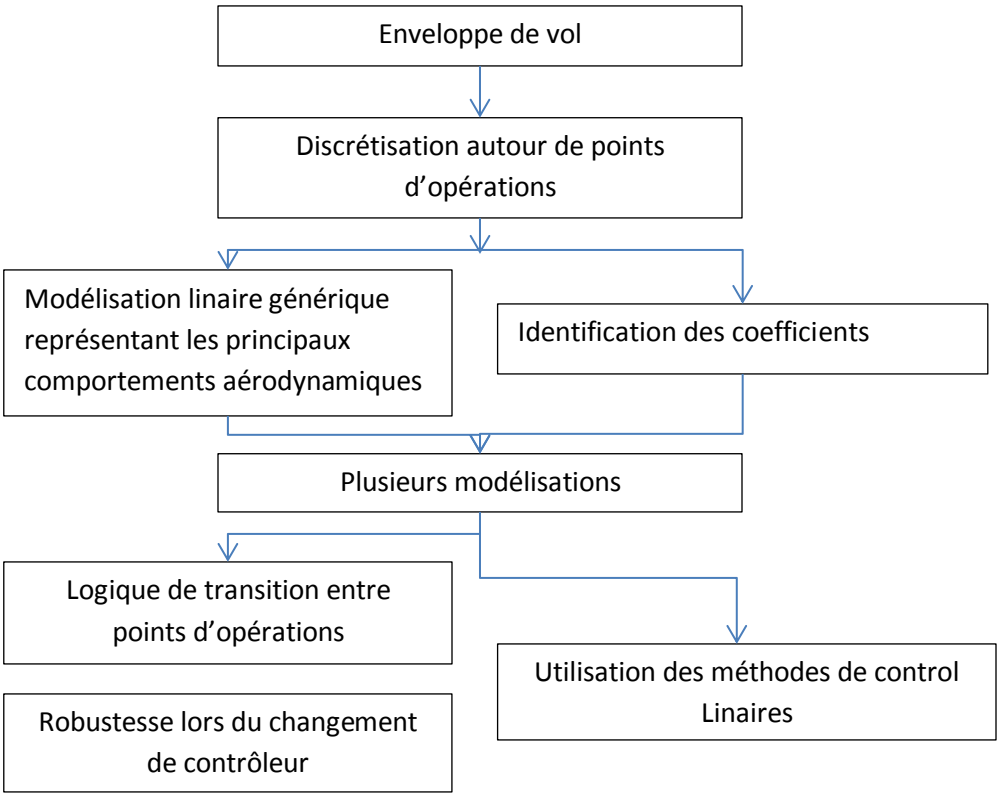
La réalisation de l'ensemble de ces taches a nécessité la création d'un autopilote. Classiquement, cet autopilote utilise les variables d'état qui viennent des capteurs. Cependant je ne dispose pas de quoi mesurer l'ensemble de ces variables notamment, les angles d'attaque et de dérapage. Pour pallier à ce problème on utilise les angles d'Euler qui y sont fortement liés.

1-Choix du contrôleur

Les méthodes de régulation telles qu'elles m'ont été enseignées sont difficiles à mettre en œuvre. En effet, lors de mes études j'ai toujours disposé d'une représentation d'état linéaire. On linéarisait ainsi la dynamique du système autour d'un point d'opération. Ce modèle linéaire représente alors les principaux comportements dynamiques du système autour de ce point.



L'application des méthodes classiques d'asservissement garantissaient une certaine robustesse. Or pour ce projet, l'absence des coefficients aérodynamiques spécifiques à notre appareil nous empêche d'avoir une représentation d'état nominale, de linéariser puis d'appliquer des méthodes dont on peut garantir la robustesse. Cependant, la structure de l'avion dont je dispose est déjà stable et l'utilisation de correcteur PID associé à une logique permet d'éviter de passer par cette phase difficile de modélisation et d'identification. A titre d'information, ce qui est fait classiquement est :



L'idée est donc d'obtenir une certaine robustesse non pas par les techniques classiques mais par une logique qui elle serait robuste sous certaines hypothèses associée à l'utilisation d'un avion stable et d'un contrôleur PID. L'objectif n'est, ni la rapidité ni la précision mais la simplicité. Ainsi seuls les termes proportionnels sont utilisés dans un premiers temps. Le nombre de vols lors des sorties de tests n'ont pas permis de déterminer les coefficients des intégrateurs.

La connaissance du comportement du système ainsi régulé permettra alors d'établir les trajectoires possibles.

2-Architecture de l'autopilote

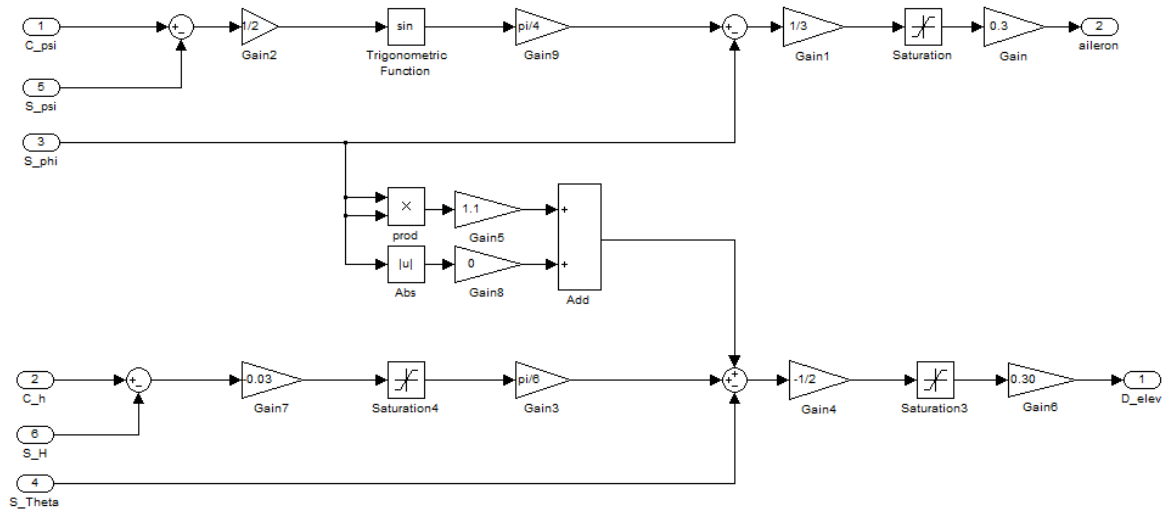
Il existe de nombreux autopilotes dans le commerce qui sont destinés au modélisme. On peut citer par exemple l'ArduPilot, Pixhawk ou Paparazzi. L'utilisation de ce genre de système nous éloigne d'un des objectifs qui est l'acquisition d'un savoir-faire. Le but est de commencer simplement pour ensuite complexifier et affiner. La pratique et l'expérimentation étant les meilleures sources d'apprentissage et d'enseignement. Il faut donc maîtriser au mieux l'autopilote, cela passe notamment par l'élaboration de son architecture et l'écriture de son code.

Un autopilote classique utilise plusieurs boucles. Le pilotage le guidage et la navigation y sont séparés. Ainsi le contrôle de l'orientation, le calcul de la position et la planification du trajet du centre de

gravité agissent de concert pour assurer la réalisation de la mission. Ces boucles, notamment le contrôle de l'attitude doivent respecter une règle. La bande passante de chaque boucle successive doit être supérieure à celle de la boucle précédente.

L'architecture que je vais décrire a pour but d'être le plus simple possible pour plusieurs raisons. Le peu de coefficients à déterminer permet de rapidement trouver une erreur si elle survient. De plus, au vu de la disponibilité du pilote, des difficultés météorologiques (vent, bourrasque et pluie) et du temps restant, il faut pouvoir rapidement les modifier.

Les autopilotes contrôlent les variables de vitesse et l'angle de dérapage et d'attaque. Cependant, pour pouvoir faire ce type de contrôle, il faut pouvoir avoir accès à ces variables grâce à des capteurs. Les capteurs permettant de mesurer ces angles n'existent pas à l'échelle du modélisme dans le commerce. Il faut les fabriquer soi-même à l'aide de capteurs de rotation possédant très peu de frottements. Cela est possible et se fait. Mais un pilote de modélisme n'a pas besoin de connaitre la direction et la norme de la vitesse de son appareil pour le contrôler. Il sait que si il cabre ou pique son appareil va monter ou descendre. C'est ce genre de connaissances qui sert à l'élaboration de l'autopilote. Un autre exemple est celui du virage. Un virage est composé d'une inclinaison puis du mouvement permettant de descendre ou de relever le nez de l'appareil. A partir du moment où l'on peut faire un virage et changer d'altitude, on peut par la suite faire de la navigation. Finalement, nous pouvons utiliser uniquement les rotations selon les axes X et Y pour amener l'avion où l'on veut. Ainsi seules les actions sur les ailerons et les gouvernes de profondeurs sont utilisées. Par soucis de sécurité, le contrôle du moteur est laissé en permanence au pilote.

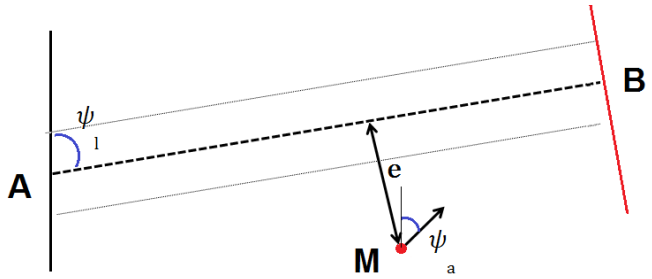


Le vol en pallier est assuré par la mesure de l'erreur en hauteur entre la hauteur de référence et la hauteur mesurée. Normalement, les modélistes confirmés règlent une valeur de trim pour une vitesse donnée qui permet d'assurer le vol en pallier. Cependant, l'avion construit est amené à évoluer au

niveau de sa charge utile. Ainsi la configuration et le centrage de l'avion peuvent varier modifiant ainsi la valeur de ce trim. De plus, pour chaque vitesse il existe une valeur de trim. Cette connaissance de l'appareil aurait pu être à ma portée si les essais avaient été possibles plus tôt. Cependant, le correcteur ainsi défini permet d'atteindre la valeur de trim quelles que soient la vitesse et la configuration. Il suffit alors de maîtriser l'erreur en hauteur. Par exemple, on sait qu'à 15m/s on a un vol en pallier avec une erreur en hauteur de 2 m. Ce type d'erreur n'est pas un problème à partir du moment où l'on vole à une altitude convenable. Cela fait partie des connaissances à posteriori dont il faudra prendre compte.

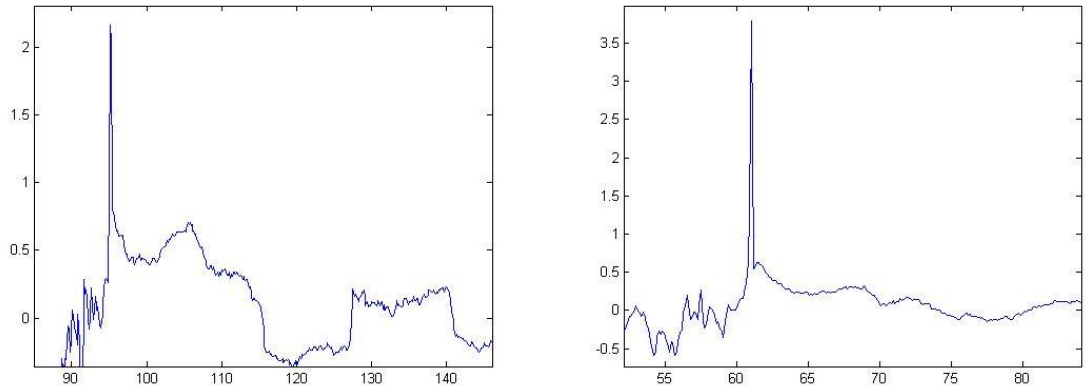
Le virage est l'association d'une inclinaison puis d'un cabrage. Ainsi, lorsque l'on veut changer de cap il faut faire un virage. On lie donc l'inclinaison à l'erreur en cap, puis on lie le cabrage à l'inclinaison. Ainsi on cabre en même temps que l'inclinaison. Le lien entre l'inclinaison et le cabrage se fait par un gain qui assure le virage et le virage en pallier en même temps. L'inconvénient de cette méthode est que le taux de changement de cap est fixé à V fixée. Ainsi lors de la planification de trajectoire il faudra connaître le rayon de courbure minimum qui fait partie des connaissances à posteriori.

Le suivi de ligne est une base pour la navigation. En considérant les coordonnées GPS des différents points A, B et M il suffit d'appliquer des formules mathématiques très simples pour obtenir ce suivi. Le franchissement du point B se mesure par le signe de $\overrightarrow{BM}.\overrightarrow{AB}$, des qu'il devient positif la droite perpendiculaire à $\overrightarrow{AB}$ passant par B est franchie. La consigne en cap fournie à l'autopilote dépend du cap de la ligne et de l'éloignement. Cet éloignement est mesuré par $e = \det(\frac{\overrightarrow{AB}}{\|\overrightarrow{AB}\|}, \overrightarrow{AM})$. Finalement la consigne est $\psi c = \psi l + a * \operatorname{atan}(\frac{e}{r})$. Les coefficients a et r sont utilisés pour gérer la réactivité de la consigne.



Enfin, le dernier point important est le décrochage. On sait que si l'angle d'incidence est trop important, l'appareil est susceptible de décrocher. Il faut donc limiter l'angle de tangage. On introduit dont une saturation pour limiter cet angle en prenant en compte les éventuels dépassements. On impose ainsi que la consigne soit comprise entre -10 degrés et +20 degrés. Cet intervalle de valeurs doit lui aussi être réglé empiriquement. De même, la vitesse de décrochage peut être approximée à l'aide de la formule $P = \frac{1}{2}\rho S V^2 C l$. Il faudra veiller à rester au-dessus avec un coefficient de sécurité.

Le manque de temps pour effectuer les tests ne m'a pas permis de réaliser les autres missions qui m'ont été demandées à savoir le décollage et l'atterrissage. Cependant, une fois que le suivi de ligne fonctionne il est simple de pouvoir faire un décollage. Le décollage sélectionné est dans un premier temps un décollage avec lancement manuel. Ce lancement serait détecté par un pic d'accélération selon l'axe X.



Pic d'accélération mesurée lors d'un lancement manuel

L'atterrissage est quant à lui un challenge qui nécessitera très certainement, l'ajout d'un capteur supplémentaire. L'idée initiale est de couper le moteur à partir d'une hauteur puis de maintenir un angle de tangage supérieur à 0. Les principales difficultés viennent des bourrasques de vent qu'il peut y avoir au sol.

3-Réglage des gains

La simplicité de l'autopilote et l'utilisation d'un correcteur proportionnel ont une contrepartie. En effet, ce réglage n'est valable qu'autour d'une vitesse donnée. La vitesse choisie pour l'appareil correspond à une puissance du moteur de 200W, soit un peu moins du tiers de la puissance maximale. On peut remarquer deux choses, la première est que comme cela avait été prévu le moteur est largement surdimensionné. Deuxièmement, la vitesse de l'avion autour des 10m/s est faible pour permettre à un débutant de facilement réagir et donc de le piloter. Ce dernier choix peut être controversé. Plus la vitesse est élevée moins l'avion est sensible aux vents ceci est aussi vrai pour le poids.

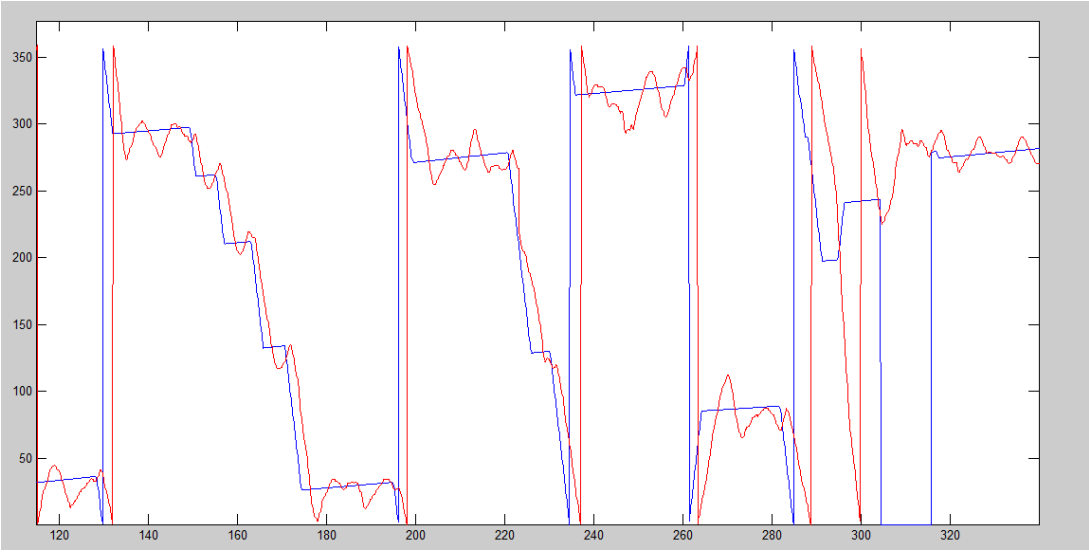
```
float gainErrPhi = 0.3;
float gainErrTheta = 1.5;
float gainCompensationRoulis = 0.02;

// A regler
float gainErrPsi = 2;
float gainErrH = 3;
```

Gains utilisés

Autour de cette vitesse, il a fallu séparer chaque boucle pour vérifier les gains. La télécommande disposait d'un bouton 3 positions qui a été utilisé pour changer de boucle. La première position du bouton, la plus naturelle et la plus simple (tout en bas) permet au pilote de contrôler normalement l'avion. La position 2 permet de contrôler le roulis et le tangage. La positon 3 permet de contrôler la hauteur et le cap.

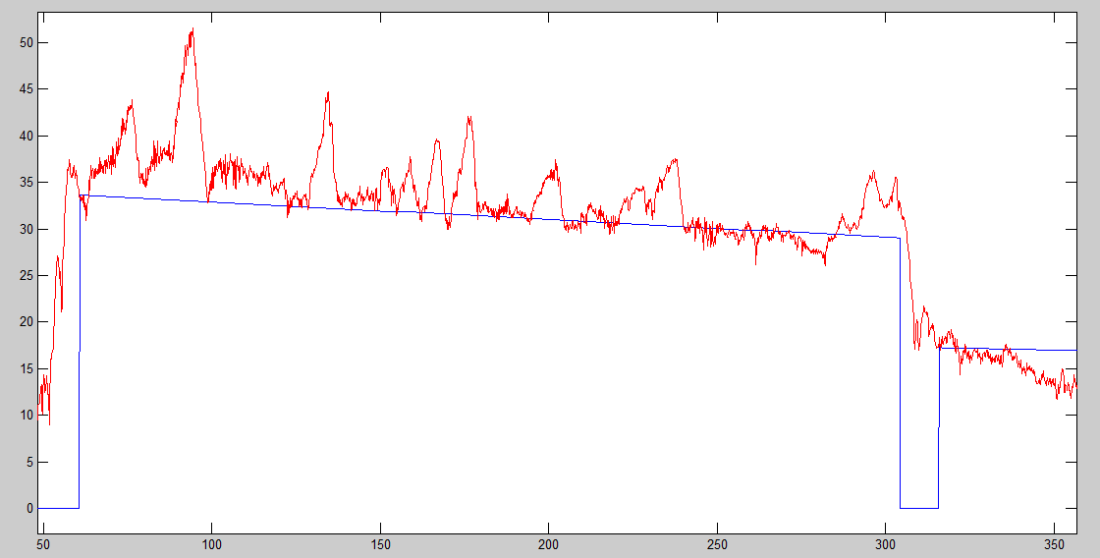
Position du bouton	Joystick	Effet
Pour toutes les positions du bouton	Droit, mouvement haut/bas	Contrôle de la puissance moteur
Position 1	Droit, mouvements latéraux	Contrôle directe des ailerons
	Gauche, mouvements haut/bas	Contrôle directe gouvernes de profondeur
Position 2	Droit, mouvements latéraux	Contrôle de la consigne de roulis
	Gauche, mouvements haut/bas	Contrôle de la consigne de tangage
Position 3	Droit, mouvements latéraux	Contrôle de la consigne de cap
	Gauche, mouvements haut/bas	Contrôle de la consigne de hauteur

L'affinement pour les réglages des gains ont été compliqués mais il est à noter que les gains estimés avec le simulateur ont très bien fonctionnés. Sur les 4 demi-journées de tests que m'ont permis la météo et la disponibilité du pilote, seule une, la première journée, s'est déroulée sans vent et sans bourrasques. Les données suivantes sont issues de la dernière journée de tests avec du vent et des bourrasques.



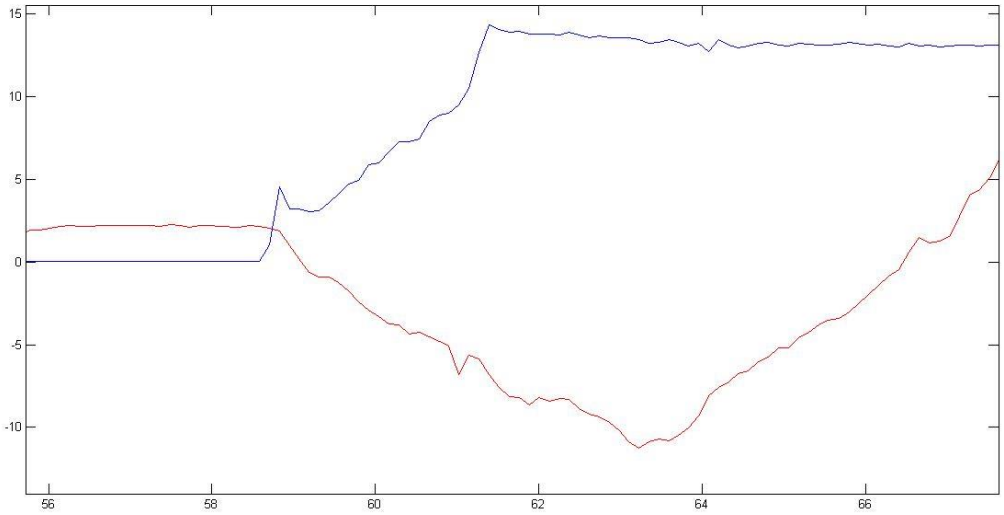
Suivi de cap par l'autopilote avec vent et bourrasques

Les données ci-dessus qui représentent le suivi en cap de l'appareil. La consigne de cap est représentée en bleu, la mesure du cap est représentée en rouge. Le pilote règle le cap à l'aide de la manette puis laisse l'autopilote gérer. Même si le suivi semble bon, les observations faites au sol ont permis de discerner une déviation, l'avion vole en crabe. Cette déviation devra être corrigée par l'algorithme de suivi de ligne. Cette déviation est due aux vents transversaux. Les données de hauteur sont issues du baromètre. On peut voir d'importantes erreurs entre la consigne de hauteur en bleu et la mesure du baromètre en rouge. Les variations aussi importantes de hauteur viennent de deux effets qui se combinent. Le premier est le fait que l'avion oscille et monte parfois brusquement sous l'action du vent, dans ce cas de figure il est donc normal de voir de telles mesures. Le deuxième est que le flux d'air fait varier la pression au niveau du baromètre. L'altitude qui est estimée est alors biaisée.

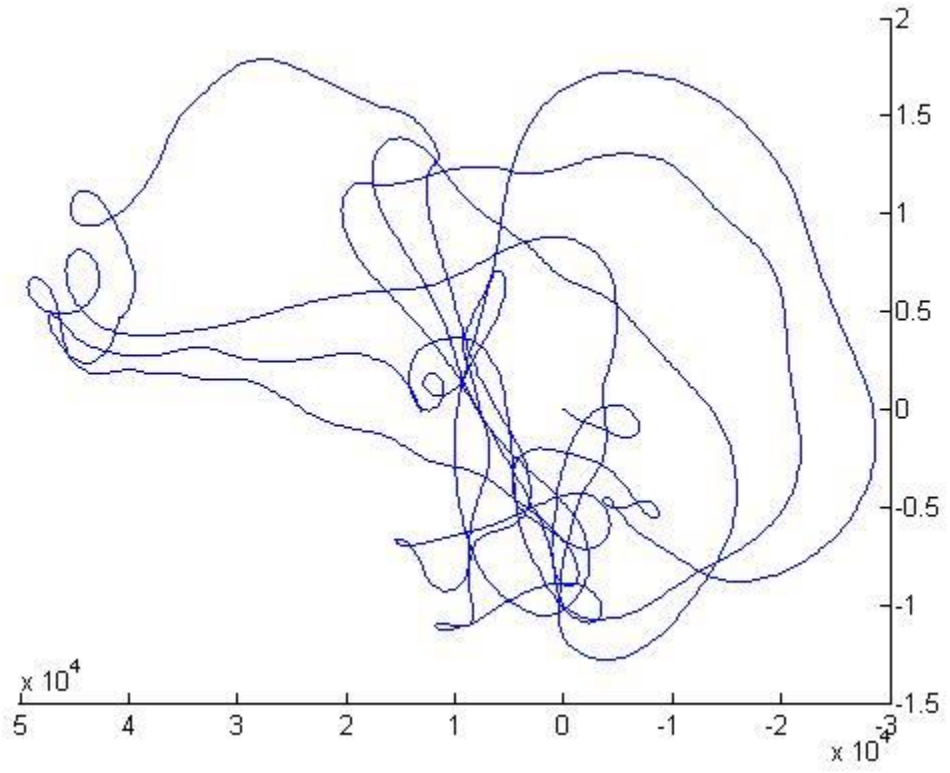


Suivi de hauteur par l'autopilote avec vent et bourrasques

Cet effet sur le baromètre est particulièrement visible avec notre appareil puisque le flux d'air de l'hélice est directement projeté sur le fuselage.



Diminution de l'altitude estimée lors de l'accélération
(Rouge altitude estimée, Bleu puissance moteur)



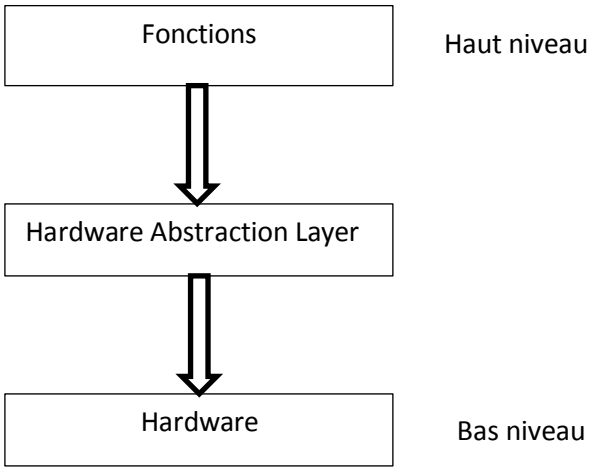
Trace GPS d'un vol

4-Electronique

Le développement d'une électronique complète prend beaucoup de temps. Beaucoup de capteurs sont nécessaires et beaucoup des données que l'on obtient nécessitent un traitement. Les angles d'Euler qui sont utilisés par l'autopilote à défaut d'avoir accès aux angles de dérapage et d'attaque, sont calculés grâce à la fusion des données venant des gyroscopes, des accéléromètres et des magnétomètres. Cette fusion nécessite un bon algorithme comme le filtre de Kalman ou les DCM (direct cosine matrix) et requiert la connaissance des incertitudes des capteurs. De même, le calcul de l'altitude en se basant sur le modèle de l'atmosphère standard nécessite la comparaison d'une valeur de la pression au sol, le zéro, à la valeur actuelle. Les capteurs de pression du tube Pitot nécessitent souvent un étalonnage. Or des dispositifs comme une soufflerie n'est pas à ma disposition. Enfin il faut initialiser l'ensemble des capteurs lors de la mise sous tension.

Ce travail est conséquent et nécessite à lui seul plusieurs mois de travail. Or de nombreux autopilotes existant ont besoin des mêmes données. L'idée est donc d'utiliser leur architecture et leurs capteurs mais de modifier le code de manière à ce que ce soit notre autopilote qui commande l'avion. L'un d'eux, l'ArduPilot à base d'atmega était déjà disponible au laboratoire. C'est donc sur ce système que mon travail s'est centré. Il se code en C et utilise l'environnement de l'arduino simple à mettre en œuvre. L'élaboration du code, la simplification et l'ajout de matériel sont les trois points sur lesquels j'ai travaillé pour ce système.

Le code nécessite l'utilisation des classes et de l'HAL fournit avec la carte. Le code est mis sur une Atmega2560 qui est connectée aux capteurs et aux récepteurs de la télécommande à l'aide de ses pins. De même les communications par les ports I2C ou UART se font sur des pins précises. Il faut donc une couche intermédiaire le HAL (hardware abstraction layer) pour perdre un niveau d'abstraction. Sur ce HAL, les méthodes plus classiques de communications et de communication avec des capteurs comme le GPS sont développées. Mon code utilise donc l'ensemble de ses classes pour pouvoir remplir sa fonction. Le code est donné en Annexe



Le travail originalement fait sur l'ArduPilot repose sur une grosse communauté. Ainsi quand on lit le code et que l'on suit le cheminement du traitement et de l'élaboration de la commande on remarque un désordre relatif. Le point principal est l'utilisation d'un tableau que l'on parcourt en 20ms et qui contient l'ensemble des fonctions à faire. Ces fonctions propres au code original sont dans 25 scripts. L'écriture m'a donc permis de rassembler l'appel aux méthodes les plus importantes et respectant l'idée du tableau initial.

Le dernier des trois points consiste en l'ajout d'une mémoire externe facilement accessible et utilisable ainsi qui l'ajout d'un module xbee. Un kit de communication que je n'ai pas pu me fournir existe. La carte contient une mémoire flash sur laquelle il est autant compliqué d'écrire que d'ylire. Ainsi une mémoire SD a été ajoutée. Les problèmes de voltages, de communication et de bus ont été réglés au fur et à mesure par l'utilisation d'un shield recueillant la carte SD et le module xbee et une arduino.

IV- Expérimentation

Les tests sont l'aboutissement du travail fait. Ils permettent de s'assurer du bon fonctionnement de l'avion, de tester l'électronique, et de valider l'autopilote. Ces tests n'ont été possibles que grâce à l'aide d'un pilote de modélisme très expérimenté. Malgré la présence de plusieurs modéliste au sein de l'ENSTA aucun ne s'est senti d'assurer la sécurité de l'avion. C'est donc très tard dans le stage qu'un pilote avec plus de 20 ans de pratique a été trouvé et m'a offert son aide.

Le recours et l'aide d'un vrai pilote est l'un des points essentiels qui a permis de compléter une grande part de ce projet. Initialement, la demande d'un de mes tuteurs était de construire un avion facilement réparable et pilotable destiné à des élèves qui sont inexpérimentés. Pour essayer de répondre à cette requête, je me suis personnellement entrainé sur simulateur. Cependant malgré l'apprentissage des réflexes de base, le passage à la réalité est complexe. La conclusion est simple, le recours à quelqu'un d'expérimenté est obligatoire.

Il est vrai que l'aide d'un vrai pilote permet d'améliorer la sécurité des personnes au sol et celle de l'appareil. Cependant, cela ne doit pas être un prétexte pour faire voler l'appareil dans un environnement inadapté. La pratique avant mon arrivée était de tenter de faire voler les appareils par les élèves sur le stade de foot de l'ENSTA. Ce stade est entouré de la gendarmerie, de la prison et des locaux de l'ENSTA. Le nouveau terrain quant à lui est plus adpaté.

C'est donc au cours de 4 demi-journées qui ont permis 12 vols que j'ai pu faire les tests et les réglages.

Ce travail a donc permis de

- Récolter des résultats
- Exploiter les résultats des tests
- Gérer les tests
- Changer les habitudes

1-Préparation des vols

Les sorties pour tester l'appareil et l'autopilote ont été rares pour 3 raisons. La première est le temps qu'il a fallu pour trouver le pilote et sa disponibilité. La deuxième est les conditions météorologiques : le vent et la pluie sont fréquents. Il faut faire les tests le matin car le vent y est le plus faible. La troisième est la difficulté à trouver un moyen de locomotion qui puisse transporter l'avion. L'établissement possède des voitures que mon statut m'a empêché d'utiliser.

Terrain où ont eu lieu les essais

Pour que le temps passé sur les essais soit optimal il faut être prêt à réagir aux imprévus. Ces imprévus sont de plusieurs sortes. Les deux principaux étant les casses et des erreurs électriques notamment au niveau des branchements.

La prise en compte du risque de casses a mené à devoir prévoir plusieurs pièces de rechanges. Ainsi les pièces capitales comme les hélices, les élastiques ou le contrôleur brushless étaient présentes en plusieurs exemplaires.

Les pannes liées à des mauvais branchements sont les plus problématiques. On peut les détecter grâce à la communication avec les modules xbee. Ils permettent de voir ce qui se passe dans l'autopilote, de s'assurer que l'écriture des données se déroule normalement. Cette étape est l'étape ultime pour vérifier le fonctionnement de l'ensemble avant de lancer l'appareil.

Terrain de l'ENSTA

La rédaction d'un guide de 25 pages qui permet de mettre en œuvre l'avion fait l'objet d'une checklist. Ainsi l'objectif de ce guide est de fournir un support pour toute nouvelle personne voulant utiliser l'appareil. Il contient les principales erreurs qui peuvent être commises notamment au niveau des branchements électriques et pour le centrage.

2-Comportement de l'appareil

Les différents vols ont permis d'apprécier le comportement de l'appareil. Ces vols doivent être repartis en deux grandes phases. La première est celle du vol classique où l'avion est commandé entièrement par le pilote, puis la seconde où l'autopilote est testé.

Mon inexpérience dans la fabrication d'avion et le caractère artisanal ont abouti à plusieurs erreurs qu'il a fallu compenser. On peut citer 4 grands problèmes.

Le premier est l'orientation du moteur électrique. Le flux d'air qui appuie sur l'empennage dévie l'avion d'un axe normal de vol. Plusieurs tests ont permis d'ajuster l'axe du moteur à l'aide de rondelles. Le deuxième est un problème de centrage. Lors d'un vol les masses utilisées pour faire le centrage ont été oubliées. Cet oubli a failli être fatal à l'avion si le pilote n'avait pas était expérimenté. Le troisième défaut vient de la construction des ailes, une erreur dans le raccord a conduit à un vrillage des ailes et dans la répartition des masses. La seule solution fut de rajouter une masse en bout d'aile. Enfin, une erreur de réglage dans la télécommande a limité le mouvement des ailerons. Cette erreur limite les virages de l'appareil dans un sens.

Une fois ces erreurs détectées et réglées, l'autopilote a été testé. Son architecture très simple fonctionne très bien quand on prend l'autopilote dans sa globalité, c'est-à-dire quand il a pour consignes une hauteur et un cap. Cependant, la boucle interne qui contrôle l'angle de tangage et de roulis a initialement posé problèmes. La première erreur est lors d'une ascension quand une bourrasque a retourné l'appareil. Le pilote a pu récupérer l'avion en repassant sur le mode de commande manuel à l'aide du bouton prévu pour ce type de situation. La solution est de diminuer l'angle de tangage, de voler avec un vent constant et sans bourrasques et de prendre en compte la dérivée de l'erreur dans le correcteur. Le dernier problème vient du lien entre l'angle de roulis et de tangage. Lors des premiers tests ce lien a été oublié. Lors d'un virage serré, l'avion est resté coincé dans sa position. L'inertie, le dièdre des ailes et sa vitesse ont réduit l'efficacité des ailerons. Pour régler ce problème, il faut diminuer l'angle maximum de roulis.

3-Données

Ce sont 12 vols repartis lors de 4 demi-journées que j'ai pu faire. Lors de ces vols les données suivantes ont été enregistrées :

- 3 accélérations
- 3 taux de rotation
- 3 angles Euler
- Les 5 commandes venant de la télécommande
- Les 2 commandes envoyées aux servomoteurs
- Les coordonnées, altitude et vitesse GPS
- Altitude barométrique
- Les consignes pour l'autopilote (hauteur, cap, roulis, tangage)
- La vitesse du flux d'air

Cependant, lors du post traitement il est très compliqué de comprendre ce qui se passe. Pour réduire cet effet on a découplé les manoeuvres, pris des notes et fait des vidéos.

CONCLUSION

Le travail effectué au cours de ce stage a permis de poser les fondements d'une activité drone à voilure fixe. Le projet initial beaucoup trop important pour une personne seule a dû être revu à la baisse. Ainsi je suis passé d'un avion-sous-marin à un drone volant à voilure fixe classique. Malgré cet aspect commun, les drones volants étant un objet d'étude classique de beaucoup d'universités, il a fallu élaborer un autopilote le plus simple possible pour me permettre de fournir des résultats à la fin, fabriquer plusieurs prototypes, les tester et donner des recommandations.

La totalité des travaux faits ont été nouveaux pour moi. Je n'ai pu m'aider d'aucun travail déjà fait sur les drones volants sur mon lieu de travail. Je ne possédais aucunes connaissances dans les drones, le modélisme ou l'électronique. C'est donc avec beaucoup de plaisir que j'ai pu combler mes lacunes en remplissant le travail demandé. Ainsi, avec peu de moyen, un avion stable et un autopilote très simple ou peut faire un drone. Ce mémoire présente les aspects principaux de ce que j'ai fait, mais d'autres travaux s'étant révélés peu utiles pour le rendu final n'y figurent pas.

L'évolution des mentalités reste encore un important challenge et je souhaite que mon insistance sur certains points soit l'élément qui reste le plus. Ainsi, des pratiques comme le vol dans des zones non recommandées, par des pilotes non expérimentés et avec des conditions météorologiques non adaptées devraient être abandonnées. Enfin le partenariat naissant avec le pilote est à entretenir car les compétences et le savoir-faire qu'il apporte sont l'une des clefs de la réussite de ce type de projet.

Le futur de la partie drone de ce projet réside dans l'achat d'un avion du commerce. Cet aspect mainte fois abordé reste pour moi fondamental malgré l'objectif pédagogique de mes tuteurs. L'achat du même modèle que celui de l'université du Minnesota permet d'utiliser leurs travaux d'identification et confère donc une bonne base pour poursuivre. L'aspect sous-marin de ce projet n'a pas été abordé. Plusieurs problématiques d'ordre mécanique ont été soulevées au début du stage mais restent sans réponses et sont à elles seules de véritables challenge.

BIBLIOGRAPHIE

Randal Beard, Timothy Mclain : Small Unmanned Aircraft - Theory and Practice

Eugene A. Morelli, Vlasdislav Klein : Aircraft System Identification: Theory And Practice

Thor I. Fossen: Mathematical models for control of aircraft and satellites

Mario Landry : Commande de vol non-linéaire en temps réel d'un drone à voilure fixe

Damien Poinsot : Commande d'un drone en vue de la conversion vol rapide – vol stationnaire

Andrei Dorobantu, Austin Murch, Bérénice Mettler, and Gary Balas University of Minnesota, Minneapolis, Minnesota 55455: System Identifiaction for small, low-cost, fixed wing unamanned aircraft

ANNEXE 1

```
////////////////////////////////
//  A regler en Premier
// GAINS POSITIFS
float gainErrPhi = 0.3;
float gainErrTheta = 1.5;
float gainCompensationRoulis = 0.02;

// A regler
float gainErrPsi = 2;
float gainErrH = 3;

////////////////////////////////
// Depend du montage sur l'avion
// en degré
// course max des surfaces mobiles
float angleMaxAileron = 13; // utiliser dans degreToPWMaileron
float angleMinAileron = -13;

float angleMaxElevator = 26; // utiliser dans degreToPWMelevator
float angleMinElevator = -26;

////////////////////////////////
// Pour gerer les angles d'inclinaison max
// utiliser pour la saturation des commandes
float thetaMax = 30;
float thetaMin = -15;
float phiMax = 45;
float phiMin = -45;

////////////////////////////////
float consigneH = 0;
float consignePsi = 0;
float varPsi=0;
float varH=0;
float consignePhi = 0;
float consigneTheta = 0;

////////////////////////////////
// Initialisation
```

```
// PWM en entree
int pwmmode = 1950;
int pwmInRoll = 1500;
int pwmInPitch = 1500;
int pwmInThrottle = 1100;
int pwmInRudder = 1500;
int pwmNavigation = 1500;
int pwmZeroCap = 1500;

// PWM en Sortie
int pwmOutRoll = 1500;
int pwmOutPitch = 1500;

////////////////////////////////
int ticLog=0;
float compteur=0;

////////////////////////////////
int firstSwitch = 1;
int firstSwitchSW3 = 1;
int correctionCapInitialized = 0;


////////////////////////////////
float capNorth = 0;
float memoireCapNorth = 0;
int nbcoor = 4;
float consigneEcart = 0;
//long tabPoint [4][2];
//long tabLatPnt[4];
//long tabLonPnt[4];
int nav = 0;
long distmax = 500; // en degréDecimaux environ 4m decimales 5 6 7
long latA=0;
long latB=0;
long lonA=0;
long lonB=0;
long latM=0;
long lonM=0;
float ecart = 0;
float capligne = 0;
// Lat colonne 0
```

```
// Lon colonne 1
// ATTENTION le GPS donne une donnée avec 7 decimales
// exemple 48.1234567 donne 481234567
// ATTENTION googleMap donne des données avec 6 decimales
// exemple 48.123456 il faut convertir 481234560


////////// Stade de l'ENSTA
/*
//point 1
tabPoint[0][0] = 484181510; // 48.418151
tabPoint[0][1] = -44733040; // -4.473304
// point 2
tabPoint[1][0] = 484188720;
tabPoint[1][1] = -44734380;
// point 3
tabPoint[2][0] = 484187940;
tabPoint[2][1] = -44745690;
// point 4
tabPoint[3][0] = 484180800;
tabPoint[3][1] = -44744490;
*/
long tabLatPnt[4] = {484181510,484188720,484187940,484180800};
long tabLonPnt[4] = {-44733040,-44734380,-44745690,-44744490};
/*
////////// Terrain du Club de modelisme
//point 1
tabPoint[0][0] = 484764400;
tabPoint[0][1] = -45432210;
// point 2
tabPoint[1][0] = 484768910;
tabPointPoint[1][1] = -45439000;
// point 3
tabPoint[2][0] = 484764240;
tabPoint[2][1] = -45447000;
// point 4
tabPoint[3][0] = 484759780;
tabPoint[3][1] = -45441010;
*/

//////////////////////////////
// Pour simplifier on se place à V=cste imposee par une puissance moteur fixe
```

```
// ATTENTION la puissance fournie varie d'une batterie à l'autre
// => si le joystick est a 50% pour deux batteries differentes la puissance est differente
// A cette vitesse on regle le trim sur la gouverne de profondeur telque H=cst pour cette
Puissance/Vitesse
// Si on va plus vite alors l'avion monte
// SI on va moins vite alors l'avion descent
// LES GAINS DES CORRECTEURS SONT REGLES POUR CETTE VITESSE
// Il n'y a pas de gains adaptatifs => il est imperatif de rester autour de cette puissance/vitesse
// Si on veut le faire evoluer à differentes vitesses de croisiere => Gains scheduling ou retroAction
linearisante
// Sinon pour chaque V il faut reregler le TRIM et les gains
int trimmer = 0;

//////////////////////////////
//                    aileronDroit   aileronGauche   PWM
// Inclinaison a droite      se leve      se baisse     1900
// Inclinaison a gauche      se baisse     se leve      1100

//////////////////////////////
//           gouverne de profondeur     PWM
// monter           se leve           1100
// descendre         se baisse          1900

//////////////////////////////
// interrupteur SW6/7 bas=1900, milieu=1500, haut=1100
// interrupteur SW3 bas = 1500, haut = 1900

//////////////////////////////////////////////////
// OutputArdupilot fait reference aux chiffres indiques sur la carte [0,8]
//OutputArdupilot    hal.rcout
//     1            0
//     2            1
//     3            2
//     ...          ...
//     8            7
// channel 1 2 3 testees

static void testRcout(){
  hal.rcout->write(0, hal.rcin->read(0));
  hal.rcout->write(1, hal.rcin->read(1));
  hal.rcout->write(2, hal.rcin->read(2));
}
```

```
//////////////////////////////////////////////////
//InputArdupilot    hal.rcin
//     1            0
//     2            1
//     3            2
//     ...          ...
//     8            7
// channel 1 2 3 testees

//TEst
static void myLogChannel(){
  cliSerial->printf_P(PSTR("IN:\t1: %d\t2: %d\t3: %d\t4: %d\t5: %d\t6: %d\t7: %d\t8: %d\n"),
    (int)hal.rcin->read(0),
    (int)hal.rcin->read(1),
    (int)hal.rcin->read(2),
    (int)hal.rcin->read(3),
    (int)hal.rcin->read(4),
    (int)hal.rcin->read(5),
    (int)hal.rcin->read(6),
    (int)hal.rcin->read(7));
}


 // apres  ahrs_update
 // Fonctionne
 // Brancher/Televerser/Outils Moniteur Serie/attendre 10s
static void myLogAhrs(){
  Vector3f gyros  = ins.get_gyro();
  Vector3f accels = ins.get_accel();
  cliSerial->printf_P(PSTR("rpy %4d %4d %3d  g %5.1f %5.1f %5.1f  a %5.1f %5.1f %5.1f\n"),
     (int)ahrs.roll_sensor / 100,
     (int)ahrs.pitch_sensor / 100,
     (uint16_t)ahrs.yaw_sensor / 100,
     gyros.x, gyros.y, gyros.z,
     accels.x, accels.y, accels.z);
}

// apres gps_update
/*
static void myLogGps(){
  const Location &loc = gps.location();
```

```
  cliSerial->printf_P(PSTR("Lat: %ld, Lon %ld, Alt: %ldm, #sats: %d\n"),
    (long)loc.lat,
    (long)loc.lng,
    (long)loc.alt/100,
    (int)gps.num_sats());
}*/

static void myLogGps(){
  //const Location &loc = gps.location();
  cliSerial->printf_P(PSTR("Lat: %ld, Lon %ld, Alt: %ldm, #sats: %d\n"),
    (long)gps.location().lat,
    (long)gps.location().lng,
    (long)gps.location().alt/100,
    (int)gps.num_sats());
}


// apres update baro
// fonctionne
static void myLogBaro(){
  cliSerial->printf_P(PSTR("Alt: %0.2fm, Raw: %f Temperature: %.1f\n"),
    barometer.get_altitude(),
    barometer.get_pressure(),
    barometer.get_temperature());
}


static void myLogAirspeed(){
  airspeed.myUpdateAirspeed();
  cliSerial->printf_P(PSTR("%.1f m/s fin \n"), airspeed.get_airspeed());
}


////////////////////////////////////////
// ON SUPPOSE QUE:
// Telecommande en mode 1
//  Controle    ArdupilotInput    ArdupilotOutput    RecepteurXR16     hal.rc
//  Aileron          2              2              2            1
//  Profondeur       1              1              1            0
//  accelerateur     3              3              3            2
//  rudder           4              4              4            3
//  Switch 6/7       6              X              6*           5    // * regler sur la telecommande
```

```
//   Switch 3        5              X              5*          4    // * regler sur la telecommande


static void myUpdateCommand(){
  //le switch SW6/7 possede 3 positions
  // Bas (pwm=1900) = mode manuel/ milieu (1500) = controle phi et theta/ haut(1100) = controle H et
psi
  pwmmode = (int)hal.rcin->read(5);         // On recupere la position du SW6/7 indique les "modes" 1 2 ou
3
  pwmInRoll = (int)hal.rcin->read(1);       // On recupere mouvements lateraux du joystick droit =>
ModeUN aileron/ Mode2 angleRoulis/ Mode3 angle de Cap
  pwmInPitch = (int)hal.rcin->read(0);      // On recupere mouvements Avant/Arriere du joystick gauche
=> ModeUN elevator/ Mode2 angleTangage/ Mode3 hauteur
  pwmInPitch = (int)(pwmInPitch - trimmer);
  pwmInRudder = (int)hal.rcin->read(3);     // On recupere les mouvements gauche/droit du joystick
gauche
  pwmInThrottle = (int)hal.rcin->read(2);   // On recupere la commande moteur
  pwmNavigation = (int)hal.rcin->read(4);   // On recupere la position du SW3
  pwmZeroCap = (int)hal.rcin->read(6);   // on recupere la position du SW2

  hal.rcout->write(2, pwmInThrottle); // On garde toujours la main sur le moteur
  hal.rcout->write(3, pwmInRudder);

  /*
  cliSerial->printf_P(PSTR("\n prof %d ail %d moteur %d dir %d ch5 %d ch6 %d"),
    pwmInPitch,
    pwmInRoll,
    pwmInThrottle,
    pwmInRudder,
    pwmNavigation,
    pwmmode
  );*/

  if(pwmmode > 1700){



    if(pwmZeroCap>1700){
      if(correctionCapInitialized ==0){
        //Le compas s'initialise mal
        //cela pose un probleme des que l'on veut faire de la navigation
```

```
    //On oriente l'avion plein nord et on actionne le switch2
    //cela memorise la valeur du cap direction nord
    setZeroCap();
   }
   else{
    capNorth = memoireCapNorth;
   }
 }
 else{
    capNorth = 0;
 }

 if(pwmNavigation>1700){ //Mode Navigation
 //cliSerial->printf_P(PSTR("\nMode Nav"));
   if(firstSwitchSW3 == 1){
    const Location &loc = gps.location();
    latA = (long)loc.lat; // on initialise à la position ou on se trouve
    lonA = (long)loc.lng; // on initialise à la position ou on se trouve
    nav = 0;
    latB = tabLatPnt[nav];  // le point B est le premier point du tableau que l'on doit atteindre
    lonB = tabLonPnt[nav];
    firstSwitchSW3 = 0;
    consigneH = barometer.get_altitude();
   }

   // on commande la hauteur
   varH = 0.0001*1500 - 0.0001*pwmInPitch;   // Au max 2m/s
   consigneH += varH;
   commandeHauteur(consigneH);
   // on commande le cap
   navigation();
 }

 else { // Mode Manuel
   pwmOutPitch = pwmInPitch + trimmer;
   pwmOutRoll  = pwmInRoll;
   hal.rcout->write(0, pwmOutPitch); //Profondeur
   hal.rcout->write(1, pwmOutRoll);  //Aileron

   consigneH = 0;
   consignePsi = 0;
   consignePhi = 0;
```

```
    consigneTheta = 0;

    latA = 0;
    latB = 0;
    lonA = 0;
    lonB = 0;
    nav = 0;

    firstSwitch = 1;
    firstSwitchSW3 = 1;
   }

   //cliSerial->printf_P(PSTR("Mode Manuel"));
  }

  else if ( (pwmmode < 1700) && (pwmmode > 1300)){
   // Mode 2 Controle PHI et theta
   consigneH = 0;
   consignePsi = 0;
   // Phi > 0 inclinaison a droite
   // Phi < 0 inclinaison a gauche
   // Calcul de PHI [-45°, 45°]  [1100, 1900]
   // 45/400 = 0.1125
   consignePhi = -0.1125*1500 + 0.1125*pwmInRoll; // on convertit le PWM en angle

   // Calcul de THETA [-30°, 30°] [1900, 1100]
   // 30/400 = 0.075
   consigneTheta =0.075*1500 - 0.075*pwmInPitch;  //on convertit le PWM en angle
   consigneTheta = saturation(consigneTheta,thetaMin,thetaMax);
   //cliSerial->printf_P(PSTR("Roll %d Pitch %d consignePhi %f consigneTheta %f
\n"),pwmInRoll,pwmInPitch,consignePhi, consigneTheta);
   commandeTheta(consigneTheta);
   commandePhi(consignePhi);
   firstSwitch = 1;
   firstSwitchSW3 = 1;
  }
  else{
   // Mode 3 Controle H et PSI

   if(firstSwitch){ //initialisation
    consignePsi = ((uint16_t)ahrs.yaw_sensor / 100);
    consigneH = barometer.get_altitude();
```

```
    firstSwitch = 0;
   }

   //
   // Calcul variation de PSI       Lacet a droite (1900) / Lacet a gauche (1100)
   // Psi s'accroit avec une rotation horaire
   // Psi [0,359]
   // Parametre à regler (X/400) ici 0.6°/400 = 0.0015
   varPsi = -0.0015*1500 + 0.0015*pwmInRoll; // Au max 30°/s
   consignePsi += varPsi;      // Attention 1 addition chaque 20ms
   if (consignePsi > 359) {consignePsi -= 359;}
   else if (consignePsi < 0  ) {consignePsi += 359;}
   else{}


   // Calcul variation de H        Monter (1100) / Descendre (1900)
   // Parametre à regler (X/400) ici 0.04m/400 = 0.0001
   varH = 0.0001*1500 - 0.0001*pwmInPitch;   // Au max 2m/s
   consigneH += varH;

   //cliSerial->printf_P(PSTR("ConsigneH %f varH %f consignePsi %f varPsi %f
\n"),consigneH,varH,consignePsi,varPsi);
   //cliSerial->printf_P(PSTR("Roll %d Pitch %d ConsigneH %f varH %f consignePsi %f varPsi %f
\n"),pwmInRoll,pwmInPitch,consigneH,varH,consignePsi,varPsi);
   //cliSerial->printf_P(PSTR("Roll %d Pitch %d varH %f  varPsi %f \n"),pwmInRoll,pwmInPitch,varH,
varPsi);
   //cliSerial->printf_P(PSTR("Roll %d Pitch %d ConsigneH %f consignePsi %f
\n"),pwmInRoll,pwmInPitch,consigneH,consignePsi);

   commandeHauteur(consigneH);
   commandePsi(consignePsi);

  }
}

static float saturation(float valeur, float Vmin, float Vmax){
  //cliSerial->printf_P(PSTR("valeur %f Vmin %f Vmax %f \n"),valeur, Vmin, Vmax);
  if (valeur > Vmax){return Vmax;}
  else if(valeur< Vmin) {return Vmin;}
  else {return valeur;}
}
```

```
static int degreToPWMaileron(float angleAileron){
  // angleAileron>0 alors aileronDroit se BAISSE
  // Attention CELA depend de la partie mecanique de l'avion (SERVO-Tringle-gignol-aileron)
  // aileron droit se baisse => PWM<1500
  // ICI evolution lineaire mais le comportement est plus celui d'un polynome de degré 3
   return (int)(1500 - 400*(angleAileron/angleMaxAileron));
}

static int degreToPWMelevator(float angleElevator){
  // angleElevator > 0 alors elevator se BAISSE
  // Attention CELA depend de la partie mecanique de l'avion (SERVO-Tringle-gignol-Gouverne de
profondeur)
  // ICI evolution lineaire mais le comportement est plus celui d'un polynome de degré 3
  float t1 = (1500 + 400*(angleElevator/angleMaxElevator));
  //cliSerial->printf_P(PSTR("t1 %f angleElevator %d\n"),t1,angleElevator);
   return (int)t1;

}

static void commandePhi(float consignePhi){
    // Calcul de l'erreur
    float erreurPhi = consignePhi - ((int)ahrs.roll_sensor / 100);
    // commande non bornee
    float cmdeDeA = -1*erreurPhi*gainErrPhi;
    // gain>0
    // cmdeDeA<0 => inclinaison a droite => aileron droit se leve => commandeDegreAileron<0;
    // commande degre
    float commandeDegreAileron = saturation(cmdeDeA,angleMinAileron,angleMaxAileron);
    // commande PWM
    // Aileron droit s'abaisse pour PWM<1500
    pwmOutRoll = degreToPWMaileron(commandeDegreAileron);

    //cliSerial->printf_P(PSTR("erreurPhi %f consignePhi %f commandeDegreAileron %f \n"), erreurPhi,
consignePhi, commandeDegreAileron);
    //cliSerial->printf_P(PSTR("pwmOutRoll %d \n"),pwmOutRoll);
    hal.rcout->write(1, pwmOutRoll);  //Aileron
}

static void commandeTheta(float consigneTheta){
    // Calcul de l'erreur
    int roll = ((int)ahrs.roll_sensor / 100);
    float erreurTheta = consigneTheta + gainCompensationRoulis*roll*roll - ((int)ahrs.pitch_sensor / 100);
```

```
    // Commande non bornee
    float cmdeDeE = -1*erreurTheta*gainErrTheta;
    // Commande Degre
    float commandeDegreElevator = saturation(cmdeDeE,angleMinElevator,angleMaxElevator);
    // commande PWM
    pwmOutPitch = degreToPWMelevator(commandeDegreElevator) + trimmer;

    //cliSerial->printf_P(PSTR("erreurTheta %f consigneTheta %f commandeDegreElevator %f \n"),
erreurTheta, consigneTheta, commandeDegreElevator);
    //cliSerial->printf_P(PSTR("pwmOutPitch %d \n"),pwmOutPitch);
    hal.rcout->write(0, pwmOutPitch); //Profondeur
}

static void commandeHauteur(float consigneHauteur){
    // Calcul de l'erreur
    float erreurH = consigneHauteur - barometer.get_altitude();
    // Consigne en Theta
    // On borne theta
    consigneTheta = saturation(erreurH*gainErrH, thetaMin,thetaMax);
    // Commande theta
    commandeTheta(consigneTheta);
}

static void commandePsi(float consignePsi){
    // Calcul de l'erreur
    // On pose ePsi = cPsi - mPsi (consigne - mesure)

    float erreurPsi = consignePsi - ((uint16_t)ahrs.yaw_sensor / 100);
    // Correction de l'erreurPsi [-359 , 359] vers [-180, 180]
    if ( erreurPsi > 180) {erreurPsi = erreurPsi - 360;}
    if (erreurPsi < -180) {erreurPsi = erreurPsi + 360;}
    // erreurPsi > 0 alors cPhi > 0
    // cPhi > 0 alors virage a droite
    // Consigne en PHI
    // On borne Phi entre -45 et 45
    consignePhi = saturation(erreurPsi*gainErrPsi,phiMin,phiMax);
    // Commande
    commandePhi(consignePhi);
}


static void allLog(){
```

```
  //Vector3f mag = compass.get_field();
  airspeed.myUpdateAirspeed();
  Vector3f gyros  = ins.get_gyro();
  Vector3f accels = ins.get_accel();
  const Location &loc = gps.location();
  battery.myRead();
  float time = hal.scheduler->millis();
  //cliSerial->printf_P pour affichage sur le moniteur
  cliSerial->printf_P(PSTR("\nt %f rpy %4d %4d %3d g %5.1f %5.1f %5.1f a %5.1f %5.1f %5.1f crpy %3.1f
%3.1f %4.1f %3.1f InTRPM %d %d %d %d %d OutRP %d %d AS %.1f m/s AB %0.2f m LaLoA %ld %ld %ld m
%4.1f m/s %.2f V %.2f A END\n"),
    time,
    (int16_t)ahrs.roll_sensor/100,
    (int16_t)ahrs.pitch_sensor/100,
    (uint16_t)ahrs.yaw_sensor/100,
    gyros.x,
    gyros.y,
    gyros.z,
    accels.x,
    accels.y,
    accels.z,
    consignePhi,
    consigneTheta,
    consignePsi,
    consigneH,
    pwmInThrottle,
    pwmInRoll,
    pwmInPitch,
    pwmInRudder,
    pwmmode,
    pwmOutRoll,
    pwmOutPitch,
    airspeed.get_airspeed(),
    barometer.get_altitude(),
    (long)loc.lat,
    (long)loc.lng,
    (long)loc.alt/100,
    gps.ground_speed(),
    battery.voltage(), // tension en volt
    battery.current_amps() // intensité en Ampere
    );
}
```

```
static void myLogAHRS(){
  //Vector3f mag = compass.get_field();
  Vector3f gyros  = ins.get_gyro();
  Vector3f accels = ins.get_accel();
  float time = hal.scheduler->millis();
  //cliSerial->printf_P pour affichage sur le moniteur
  cliSerial->printf_P(PSTR("t %f rpy %4d %4d %3d g %5.1f %5.1f %5.1f a %5.1f %5.1f %5.1f "),
    time,
    (int16_t)ahrs.roll_sensor/100,
    (int16_t)ahrs.pitch_sensor/100,
    (uint16_t)ahrs.yaw_sensor/100,
    gyros.x,
    gyros.y,
    gyros.z,
    accels.x,
    accels.y,
    accels.z
    );
}

static void myLogCh(){
  //cliSerial->printf_P pour affichage sur le moniteur
  cliSerial->printf_P(PSTR("InTRPM %d %d %d %d OutRP %d %d "),
    pwmInThrottle,
    pwmInRoll,
    pwmInPitch,
    pwmmode,
    pwmOutRoll,
    pwmOutPitch
    );
}

static void myLogBGPS(){
  const Location &loc = gps.location();
  airspeed.myUpdateAirspeed();
  //cliSerial->printf_P pour affichage sur le moniteur
  cliSerial->printf_P(PSTR("AB %0.2fm AS %.1f m/s LaLoA %ld %ld %ldm \n"),
    barometer.get_altitude(),
    airspeed.get_airspeed(),
    (long)loc.lat,
```

```
    (long)loc.lng,
    (long)loc.alt/100
    );
}

static void myLog10Hz(){
 ticLog++;
 if(ticLog>5){ticLog = 0;}
 if(ticLog==0){
  if(pwmNavigation>1700){logNav();}
  else {allLog();}
 }
}

static void getTrimmer(){
 float cumul = 0;
 for(int i = 0; i<10;i++){
  cumul += hal.rcin->read(0) - 1500;
  delay(50);
 }
 cumul = cumul/10;
 trimmer = (int)(cumul);
 cliSerial->printf_P(PSTR("\n trimmer %d \n"),trimmer);
}

static void myReadBattery(){
 battery.myRead();
 cliSerial->printf_P(PSTR("tension %.2f intensite %.2f \n"),
 battery.voltage(), // tension en volt
 battery.current_amps() // intensité en Ampere
 );

}

static void testXBEE(){
 //
 ticLog++;

 if(ticLog>4){ticLog = 0;}
 if(ticLog==8){
  gcs_send_text_P(SEVERITY_LOW, PSTR("\nbonjour"));
  //gcs_send_text_fmt(PSTR("\nbonjour"));
```

```
  }
  if(ticLog==0){
    compteur++;
    cliSerial->printf_P(PSTR("\nBONJOUR %f END"),compteur);
  }

}


static void readXbeeIncomingData(){
  // Le port de telemetrie est le meme que le port USB
  // donc cliSerial pour l'usb et le Xbee sur le port telemetrie
  // lit 1 octet
  // int16_t aaa = cliSerial->read(); // data dans base 10
  // aaa = -1 si pas de donnee
  char bbb = cliSerial->read(); // char
  cliSerial->printf_P(PSTR("\ndata %c"),bbb); // envoie un char

}

static void myReadSerial(){
  //
  int test6 = 0;
  char incomingData[20];
  int nb=0;
  int16_t aaa = cliSerial->read();
  while((aaa != -1) && (nb<20)){
    char bbb = aaa;
    incomingData[nb] = bbb;
    nb ++;
    aaa = cliSerial->read();
    test6 = 1;
    //cliSerial->printf_P(PSTR("\n IN"));
  }

  if(test6 == 1){
    //cliSerial->printf_P(PSTR("\n data %d %c"),0,incomingData[0]);
    //cliSerial->printf_P(PSTR("\n data %d %c"),1,incomingData[1]);
    //cliSerial->printf_P(PSTR("\n data %d %c"),2,incomingData[2]);
    //cliSerial->printf_P(PSTR("\n data %d %c"),3,incomingData[3]);
    //cliSerial->printf_P(PSTR("\n data %d %c"),4,incomingData[4]);
```

```
    cliSerial->printf_P(PSTR("\n data %d %c %d
%d"),0,incomingData[0],incomingData[0],(int)(incomingData[0]));
  }
}

static void testAtan2(){
 cliSerial->printf_P(PSTR("\n TEST ATAN2 %f %f %f %f"),
   atan2(1,1),
   atan2(-1,1),
   atan2(1,-1),
   atan2(-1,-1)
   );
}

static void navigation(){
  // on travail en degres decimaux
  // On va du point A au point B
  // L'avion se trouve au point M
  const Location &loc = gps.location();
  latM = (long)loc.lat;
  lonM = (long)loc.lng;
  //latM = tabLatPnt[3];  // pour tester en interieur
  //lonM = tabLonPnt[3];
  //latA = tabLatPnt[2];  // pour tester en interieur
  //lonA = tabLonPnt[2];

  // On test si on depasse la droite perpendiculaire a AB passant par B
  // utilisation du produit scalaire ps = BM.AB
  long ps = (latM - latB)*(latB-latA) + (lonM-lonB)*(lonB-lonA);
  if (ps>0){
    // on a franchi la ligne
    latA = latB;
    lonA = lonB;
    nav++;
    if (nav>(nbcoor-1)) {nav = 0;}
    latB = tabLatPnt[nav];
    lonB = tabLonPnt[nav];
  }

  // on calcul l'ecart à la ligne
  // utilisation du determinant et d'un vecteur unitaire de direction AB
  // det = det( AB/||AB||, AM)
```

```
  ecart = ((lonB-lonA)*(latM-latA)-(latB-latA)*(lonM-lonA))/(sqrt((latB-latA)*(latB-latA)+(lonB-
lonA)*(lonB-lonA)));
  //cliSerial->printf_P(PSTR("\n sqrt %d "), sqrt((latB-latA)*(latB-latA)+(lonB-lonA)*(lonB-lonA)));
  //cliSerial->printf_P(PSTR("\n num %d "), ((lonB-lonA)*(latM-latA)-(latB-latA)*(lonM-lonA)));
  // on calcul le cap de la ligne
  // cap = atan2(dlon,dlat)
  capligne = atan2(lonB-lonA,latB-latA); // [-pi,pi]
  // on transforme en degrés [0,359]
  if (capligne<0){capligne = (2*3.1415 + capligne)*(180/3.1415);}
  else {capligne = capligne * (180/3.1415);}

  capligne = capligne + capNorth;

  if (capligne > 359) {capligne -= 359;}
  else if (capligne < 0  ) {capligne += 359;}
  else{}

  //correction pour l'ecart à la ligne
  consigneEcart = (45*2*atan(ecart/(4*distmax)))/3.1415;

  //elaboration de la consigne de cap
  consignePsi = capligne + consigneEcart;
 /*
  if(ecart>distmax){consignePsi = capligne + 20;}    // on est a gauche et trop loin de la ligne
  else if(ecart<-distmax){consignePsi = capligne - 20;}    // on est a droite et trop loin de la ligne
  else{consignePsi = capligne;}*/

  //on limite la consigne [0,359]
  if (consignePsi > 359) {consignePsi -= 359;}
  else if (consignePsi < 0  ) {consignePsi += 359;}
  else{}

  commandePsi(consignePsi);




}


static void logNav(){
   const Location &loc = gps.location();
```

```
  float time = hal.scheduler->millis();
  //cliSerial->printf_P pour affichage sur le moniteur
  cliSerial->printf_P(PSTR("\nt %f "), time);
  cliSerial->printf_P(PSTR("capN %4.1f "), capNorth);
   cliSerial->printf_P(PSTR("cap %d "), (uint16_t)ahrs.yaw_sensor/100);
   cliSerial->printf_P(PSTR("Ec %f "), ecart);
   cliSerial->printf_P(PSTR("CapLigne %3.1f "), capligne);
   cliSerial->printf_P(PSTR("ConsEc %3.1f "), consigneEcart);
   cliSerial->printf_P(PSTR("NAV %d "), nav);
   cliSerial->printf_P(PSTR("CPTPH %3.1f "), consignePhi);
   cliSerial->printf_P(PSTR("%3.1f "), consigneTheta);
   cliSerial->printf_P(PSTR("%4.1f "), consignePsi);
   cliSerial->printf_P(PSTR("%3.1f "), consigneH);
   //cliSerial->printf_P(PSTR("PNM %d "), pwmNavigation);
   //cliSerial->printf_P(PSTR("%d "), pwmmode);
   cliSerial->printf_P(PSTR("LaLoA %ld "), latM);
   cliSerial->printf_P(PSTR("%ld "), lonM);
   cliSerial->printf_P(PSTR("%ld "), (long)loc.alt/100);
   cliSerial->printf_P(PSTR("ptA %ld "), latA );
   cliSerial->printf_P(PSTR("%ld "), lonA);
   cliSerial->printf_P(PSTR("ptB %ld "), latB);
   cliSerial->printf_P(PSTR("%ld "), lonB);
   cliSerial->printf_P(PSTR("%4.1f m/s END"), gps.ground_speed());
}

static void setZeroCap(){
  capNorth = ((uint16_t)ahrs.yaw_sensor / 100);
  memoireCapNorth = capNorth;
  cliSerial->printf_P(PSTR("\n capNord pour correction %f END"), capNorth);
  correctionCapInitialized = 1;
}
```

ANNEXE 2

Avion et servomoteurs

1- servomoteurs et effets
2- moteur et hélice
3- fixations des ailes

Télécommande/récepteur

1- réglage télécommande en mode 1
2- visualisations des informations envoyées

Branchements ArduPilot

1- Branchement entre le récepteur XR-16 ifs et l'INPUT de l'ArduPilot :
2- Branchement entre le GPS et l'ArduPilot :
3- Branchement entre le module de puissance et l'ardupilot :
4- Branchement entre le tube Pitot et l'ArduPilot :
5- Branchement entre les actionneurs et l'ArduPilot :
6- Branchement du module télémétrie :
7- Alimentation de l'arduMega :

Logiciel

1- Paramétrage Xbee
2- Arduino
3- Teraterm
4- Données

Mise en œuvre de l'avion

1- Circuit de puissance
2- Boitier
3- Montage avion
4- Avant le décollage
5- Effet des switch
6- Sortir le boitier

Mise en gardes

1- Centrage Avant
2- Durée de Vie Batterie LiPo
3- Ne pas s'improviser pilote
4- Règlementation
5- Charge utile et contrôleur

6- Plateforme Saine

Avion et servomoteurs :

1-Servomoteurs et effets

Les servomoteurs sont fixés à l'avion. Donc un signal PWM [1000-2000] impose un mouvement défini.

- ⇨ Le programme et la télécommande doivent s'adapter aux contraintes mécaniques dues à la fixation des actionneurs.

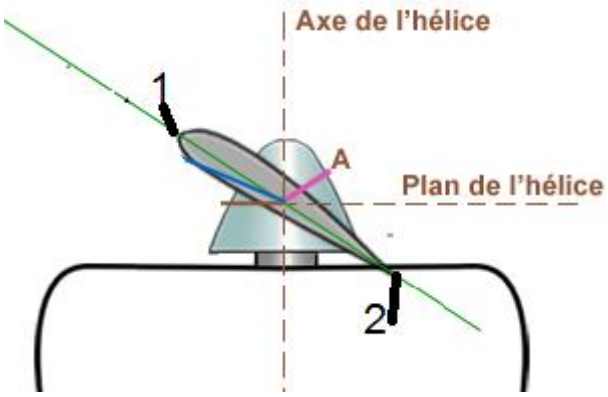
Les servomoteurs des ailerons sont liés par un câble ''y'' et sont montés symétriquement. Ils reçoivent donc le même ordre, mais agissent dans un sens opposé.

Surfaces de contrôle	PWM = 1100	PWM = 1900
	Aileron Gauche se lève	Aileron Gauche se baisse
Ailerons	Aileron Droit se baisse	Aileron Droit se lève
	⇨ Inclinaison à gauche	⇨ Inclinaison à droite
Gouverne de profondeur	La GP se lève	La GP se baisse
	⇨ L'avion cabre	⇨ L'avion pique

- ⇨ Un virage est l'action conjuguée d'une inclinaison et d'un ''cabrage ''.

2- Moteur et hélice

L'hélice à un sens de rotation et un sens de fixation sur le moteur.



1 : Bord d'attaque de l'hélice (**partie bombée**)

2 : Bord de fuite de l'hélice (**partie fine/tranchante**)

Sens de fixation :

- ⇨ Les bords d'attaques de l'hélice doivent être en avant de l'avion.

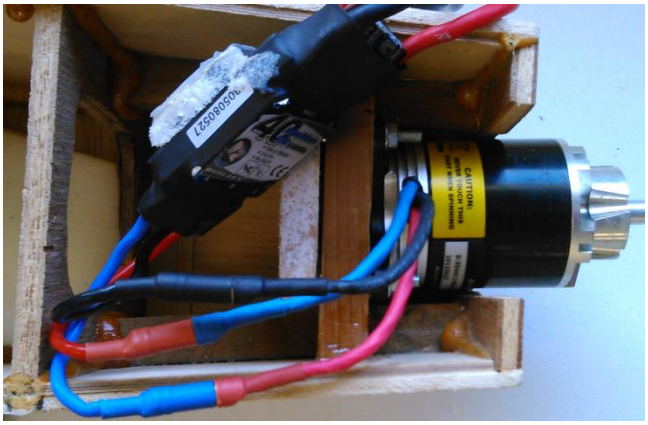
Sens de rotation :

- ⇨ Les bords d'attaques doivent attaquer l'air en premier

Le sens de rotation du moteur dépend du branchement des trois fils triphasés.

- ⇨ Il faut tester pour voir dans quel sens tourne le moteur (à faible vitesse…)
- ⇨ Si le moteur tourne en sens inverse, il suffit de permuter 2 fils sur 3.

Sur l'avion, il faut avoir qu'une connexion de même couleur (exemple : Noir-noir, Rouge-Bleu, Bleu-rouge).



3- Fixation des ailes

On utilise 4 élastiques pour fixer les ailes :

- ⇨ Deux élastiques croisés et deux élastiques en parallèles (voir image)
- ⇨ On attache les ailes après avoir fait les branchements
- ⇨ Prévoir des élastiques de modélisme de rechange car ils ont une durée de vie limitée

Télécommande et récepteur :

- ⇨ Les informations suivantes sont là pour un usage rapide de la télécommande. En cas de problème la documentation de la télécommande peut être trouvée sur internet (Graupner mx-16 iFS 2.4Ghz)

1-Reglage de la télécommande en mode 1 :

Image1

En mode 1 :

Commande du moteur :

- ⇨ Joystick Droit mouvement haut/bas

Commande Aileron :

- ⇨ Joystick Droit mouvement gauche/droit

Commande Profondeur :

- ⇨ Joystick Gauche mouvement haut/bas

Commande Direction :

- ⇨ Joystick Gauche mouvement gauche/droit

En modélisme on allume tout le temps la télécommande en premier, et on l'éteint tout le temps en dernier.

Les variateurs empêche le démarrage si le la commande des gaz n'est pas sur sa positon zéro.

Rectangle Bleu : Masse commune

Rectangle Rouge : +5V commun à tous

Rectangle Jaune : Pins de données toutes indépendantes

ENTER>Regl.servo :

Ce menu permet de régler les commandes envoyées par les joysticks

S1	=>	0%	100%	100%		Joystick Gauche haut/bas	Récepteur voie 1
S2	<=	0%	100%	100%		Joystick Droit gauche/droit	Récepteur voie 2
S3	=>	0%	100%	100%		Joystick Droit haut/bas	Récepteur voie 3
S4	=>	0%	100%	100%		Joystick Gauche gauche/droit	Récepteur voie 4

- ⇨ La première colonne fait référence aux voies du récepteur (exp : S1 = voie 1 récepteur)
- ⇨ La deuxième colonne permet de changer les sens de variation du signal PWM d'un joystick (exemple, [1900,1100] à [1100,1900])
- ⇨ Colonne 3 permet de changer la valeur neutre (1500 lorsqu'a 0%)
- ⇨ Colonne 4 permet de changer la plage de variation [1500-400*X%,1500]
- ⇨ Colonne 5 permet de changer la plage de variation [1500, 1500+400*X%]

ENTER>Regl.Contr :

Ce menu permet d'envoyer des commandes supplémentaires à l'aide des boutons de la télécommande.

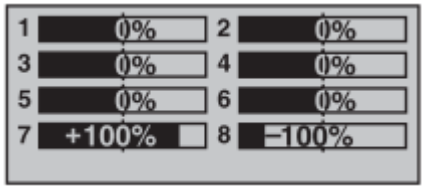
E5	3	+0%	+100%		Switch 3	Récepteur voie 5
E6	Comm8	+100%	+100%		Switch 6/7 (Comm8)	Récepteur voie 6
E7	2	+0%	+100%		Switch 2	Récepteur voie 7

E8			Récepteur voie 8

- ⇨ Colonne 1 fait référence aux voies du récepteur (exp : E5 = voie 5 récepteur)
- ⇨ Colonne 2 fait référence au bouton sur la télécommande
- ⇨ Colonne 3/4 fait référence aux variations du bouton

2- Visualisation des informations envoyées :

Appuyer une fois sur la roulette à droite de l'écran pour obtenir :

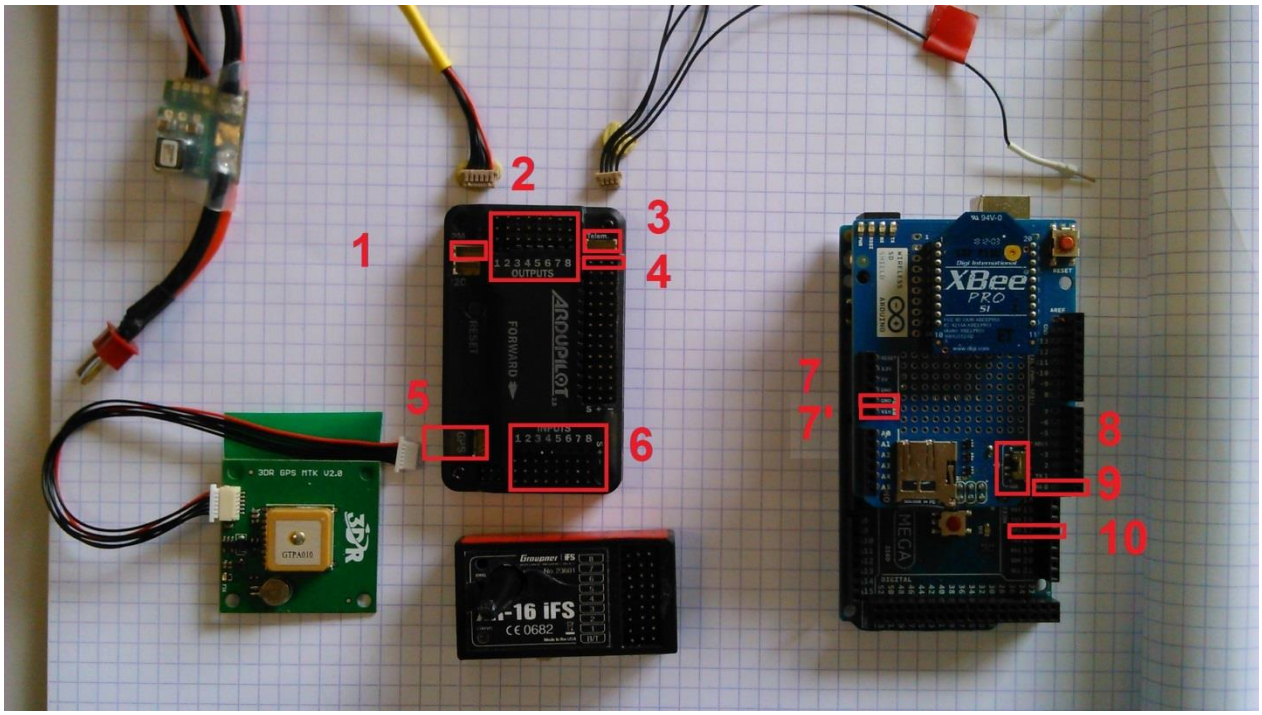


Ceci est un **exemple**

- ⇨ En touchant les joysticks, les variations à l'écran doivent correspondre à l'image 1.
- ⇨ On peut visualiser les ''trims'' en laissant les joysticks au repos pour les voies 1 2 et 4
- ⇨ 0% = 1500 ; -100% = 1100 ; 100% = 1900 ;
- ⇨ La voie 4 à un trim de +28% pour compenser certains défauts de l'avion

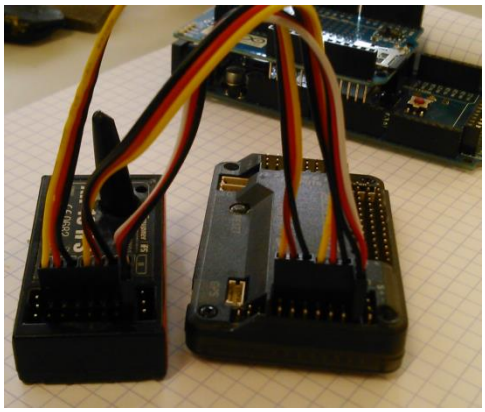
Il est important d'avoir les trims des ailerons et profondeur à 0 car le programme sur l'ArduPilot se base par rapport à une valeur de référence qui est 1500 (repos).
Le trim décale cette valeur à 1500 + trim.

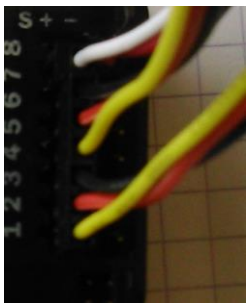
Branchement ArduPilot



Branchement entre le récepteur XR-16 ifs et l'INPUT de l'ArduPilot :

INPUT ArduPilot **(6)**	OUTPUT XR-16 ifs
1	1
2	2
3	3
4	4
5	5
6	6
7	7

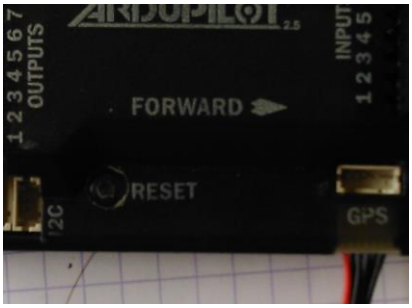




/!\ Attention /!\

- ⇨ On parle bien de la voie 1 et non B/T
- ⇨ Faire attention au branchement (+ sur + etc)
- ⇨ L'émetteur est alimenté par l'ArduPilot
- ⇨ Pour économiser les câbles, il y a un seul câble d'alimentation (branché en 7)
- ⇨ 2 câbles PWM permettent de relier les 2*6 pins de data entre elles

Branchement entre le GPS et l'ArduPilot :

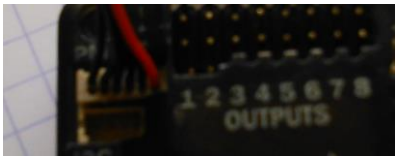


Le connecteur a un détrompeur.

Le connecteur GPS sur l'ArduPilot **(5)** est indiqué sur la coque.

Câble rouge vers l'output de l'ArduPilot

Branchement entre le module de puissance et l'ardupilot :



Le connecteur a un détrompeur.

Le connecteur ''PM'' (power module) est en **(1)**

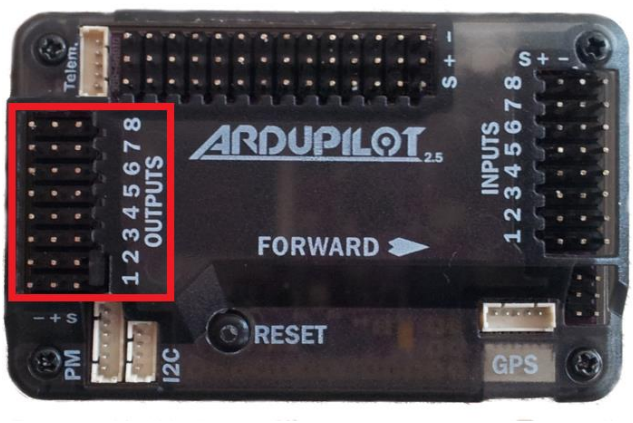
Il permet d'alimenter l'ArduPilot et tous les capteurs (dont le récepteur) qui y sont fixés.

Branchement entre le tube Pitot et l'ArduPilot :

Brancher le connecteur sur A0 (voir indication sur la coque/boitier) **(4)**

Respecter les polarités (+5 sur +5, etc)

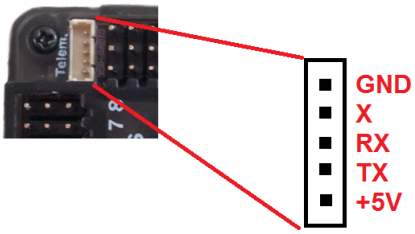
Branchement entre les actionneurs et l'ArduPilot :

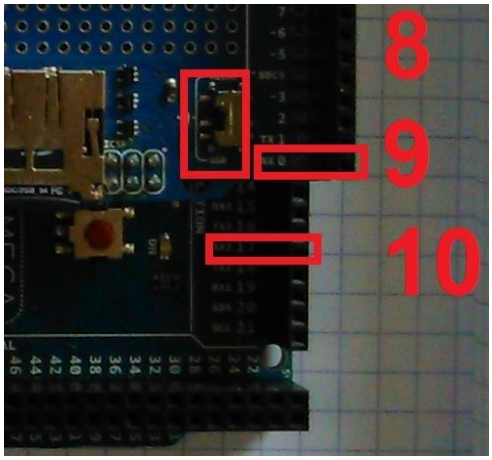


OUTPUTS **(2)**	Actionneurs	
	1	Gouverne de profondeur
	2	Ailerons
	3	Moteurs
	4	Gouverne de direction

⇨ La prise BEC du contrôleur permet d'alimenter les servomoteurs

Branchement du module télémétrie :





/!\ Le connecteur utilisé ne dispose que de 4 câbles /!\
/!\ La pin +5V doit rester libre ⇔ le connecteur est contre le bord de la carte/!\

ArduPilot	Câble	ArduMega + Shield
TX **(3)**	Blanc	RX2 pin 17**(10)**
RX **(3)**	Bleu	RX0 pin 0 **(9)** (CE N'EST PAS UNE ERREUR)
X (rien) **(3)**	Jaune	X (rien)
GND **(3)**	Noir	X (rien)

- ⇨ Le port de télémétrie est actif que si le port USB de l'ArduPilot n'est pas utilisé
- ⇨ Les données sont envoyées par l'ArduPilot sur le port TX, réceptionnées par l'arduMega sur le port RX2 (pin 17) puis loggées sur la carte SD
- ⇨ L'écriture sur la carte SD prend du temps (entre 50 et 130 ms)
- ⇨ Le lien RX-RX0 permet de recevoir des données directement par la carte XBEE
- ⇨ On ne peut recevoir des données par la carte XBEE que si le switch **(8)** est sur MICRO

Alimentation de l'arduMega :

Brancher deux câbles pour relier:

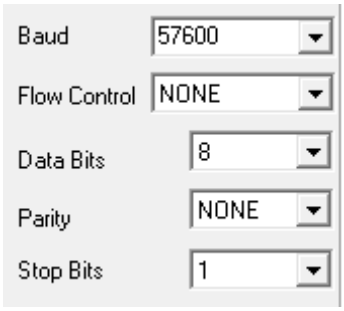
ArduPilot	ArduMega
+5	Vin **(7')**
GND	GND **(7)**

- ⇨ Faire attention à la polarité
- ⇨ Il arrive que les câbles se débranchent lors de la manipulation du ''boitier''

Logiciels :

Paramétrage des Xbee :

On utilise le logiciel X-CTU pour paramétrer les xbee.



Networking & security :

- ⇨ Channel: "C"
- ⇨ PAN ID : Identité du réseau (tous les XBEE du réseau on cette même adresse exp 3332)
- ⇨ Destination Adress High: "0"
- ⇨ Destination Adress Low: "0"

Serial interfacing:

⇨ Interface data rate: 57600

Arduino :

Suivre le tuto disponible sur le lien suivant :

http://dev.ardupilot.com/wiki/building-ardupilot-with-arduino-windows/

Apres avoir installé arduino et git, remplacer le dossier ArduPilot présent dans le dossier GIT par celui fournit.

- ⇨ Certaines librairies ont été modifiées par l'ajout de méthodes
- ⇨ Arduplane.ino est modifié et myStaticM.ino est créé

Le televersement du programme met environ 1 min.

TeraTerm :

Permet de lire et d'enregistrer les données arrivant sur le Xbee :

Pour enregistrer les données qui s'afficheront sur le terminal : Fichier> TTY record

1-

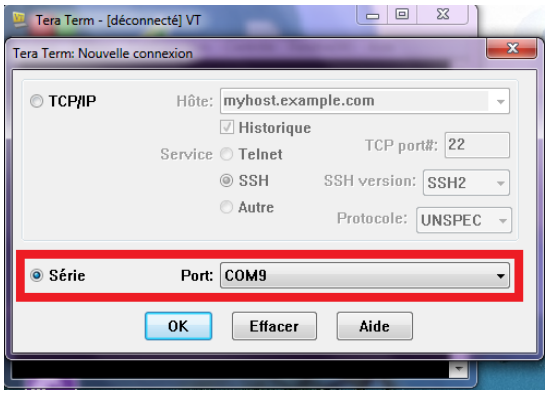
Connecter le Xbee à l'ordinateur

2-

Lancer TeraTerm

3-

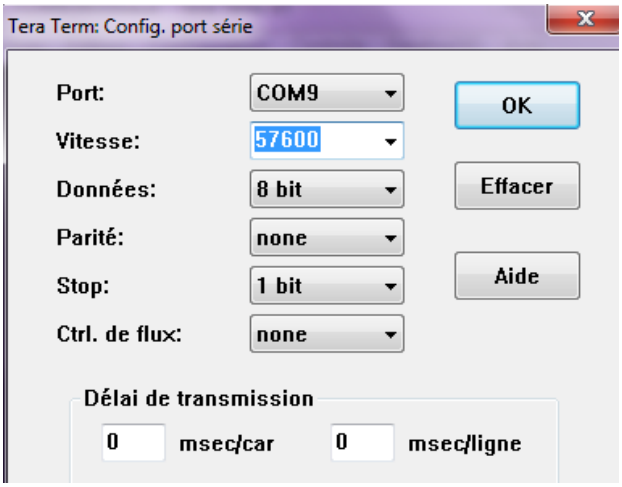
Selectionner le port série (COM X) du Xbee



4-

Configuration>Port Série

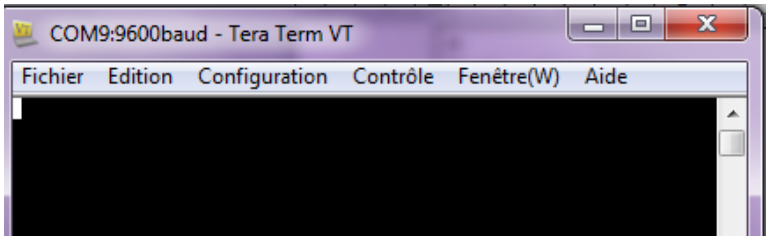
Régler la vitesse à 57600



5-

Le terminal est réglé.

Dès que l'Xbee recevra une donnée elle sera affichée



<u>Données :</u>

Lorsque l'avion est en vol les données reçues par l'Xbee sont souvent incomplètes.

Cependant cela permet de vérifier que :

- Il y a des données
- La carte SD est correctement branchée
- Si les données sont envoyées alors elles sont logées

Les données présentent sur la carte SD sont :

t 12198.00	T pour temps
	Le nombre correspond au temps du timer interne
rpy 0 -1 119	Rpy : Roll Pitch Yaw angles euler en °

g -0.0 -0.0 -0.0	g : gyroscope selon les axes x y et z
a 0.1 0.1 -9.8	a : accéléromètre selon les axes x, y et z
crpy 0.0 0.0 0.0 0.0	Crpy : valeur de commande roll pitch yaw hauteur
InTRPM 1112 1502 1501 1502 1907	Valeur en entrée de ArduPilot input pour : moteur/aileron/profondeur/direction/ switch6/7
OutRP 1502 1502	Valeur en sortie de output ardupilot pour aileron/profondeur
AS 2.5 m/s	Mesure de la sonde pitot
AB 0.04 m	Altitude baro (mesure du baromètre % à son zéro)
LaLoA 484766259 -45406961 115 m 14.4 m/s	Latitude longitude altitudeGPS vitesseSol
	Le gps donne des valeurs avec 7 decimales donc
	484766259 = 48.4766259
	-45406961 = -4.5406961
12.01V 0.14 A	Tension batterie, intensité
END	Mot qui indique à l'arduMega d'écrire les données sur la carte SD

L'écriture sur la carte SD avec l'ardu mega met entre 30 et 130 ms d'où une fréquence d'échantillonnage de 8hz.

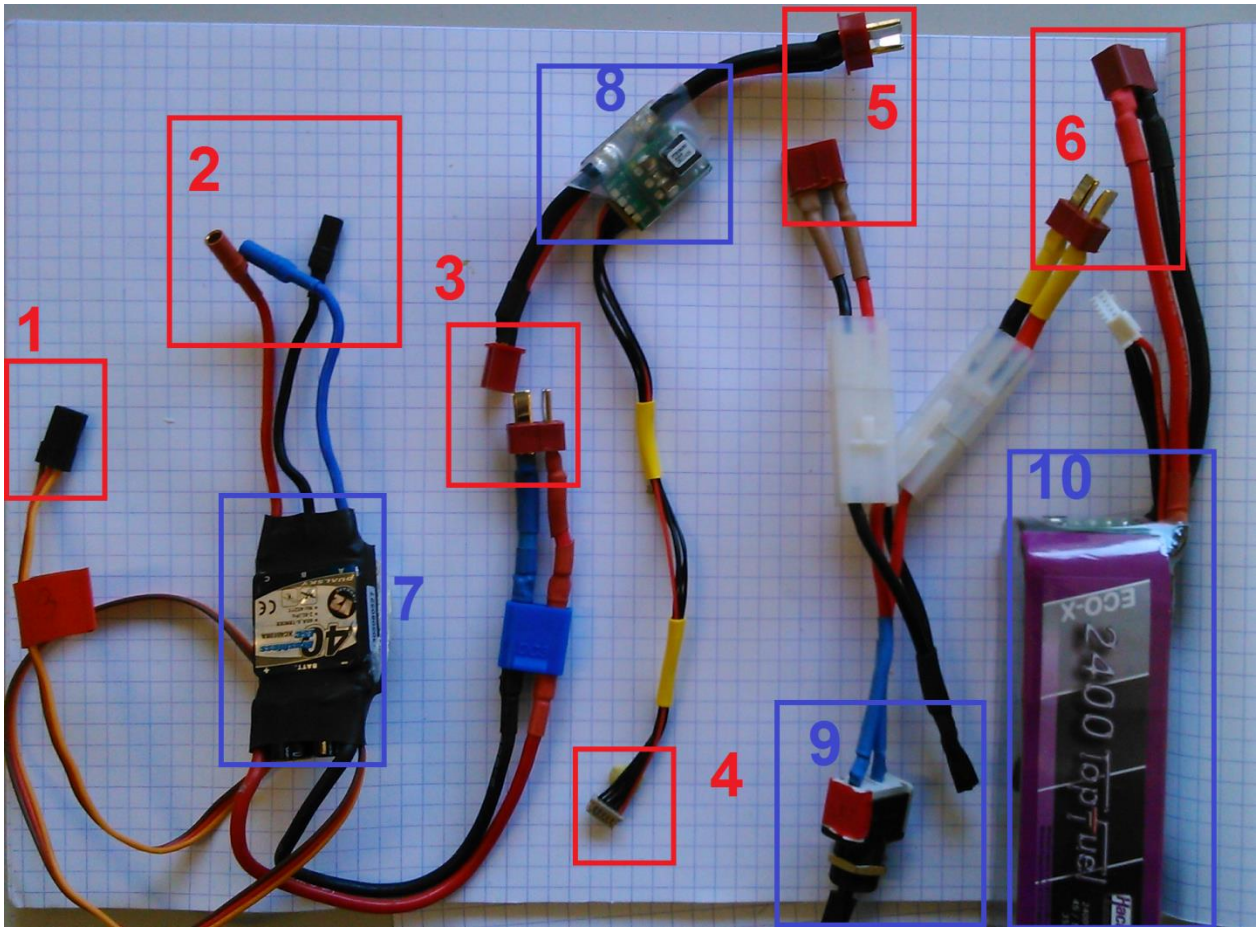
Pour avoir une fréquence plus élevée, deux options :

- Prendre autre chose que l'arduMega (raspberry Pi beaglebone)
- Utiliser la mémoire flash de 4Mo disponible sur l'ArduPilot

<u>Divers :</u>

Mise en œuvre de l'avion :

Circuit de puissance :



1 – Sortie BEC, branchée sur output3

2 – Sortie triphasé, donne la tension au moteur brushless

3 – Connection entre le contrôleur brushless et le module de puissance

4 – Alimentation de l'ArduPilot

Branchée sur la borne ''PM''

5 – Connection entre le power Module et l'interrupteur

6 – Connection entre l'interrupteur et la batterie

7 – Contrôleur brushless

Reçoit les ordres pour le moteur et les convertit en tension

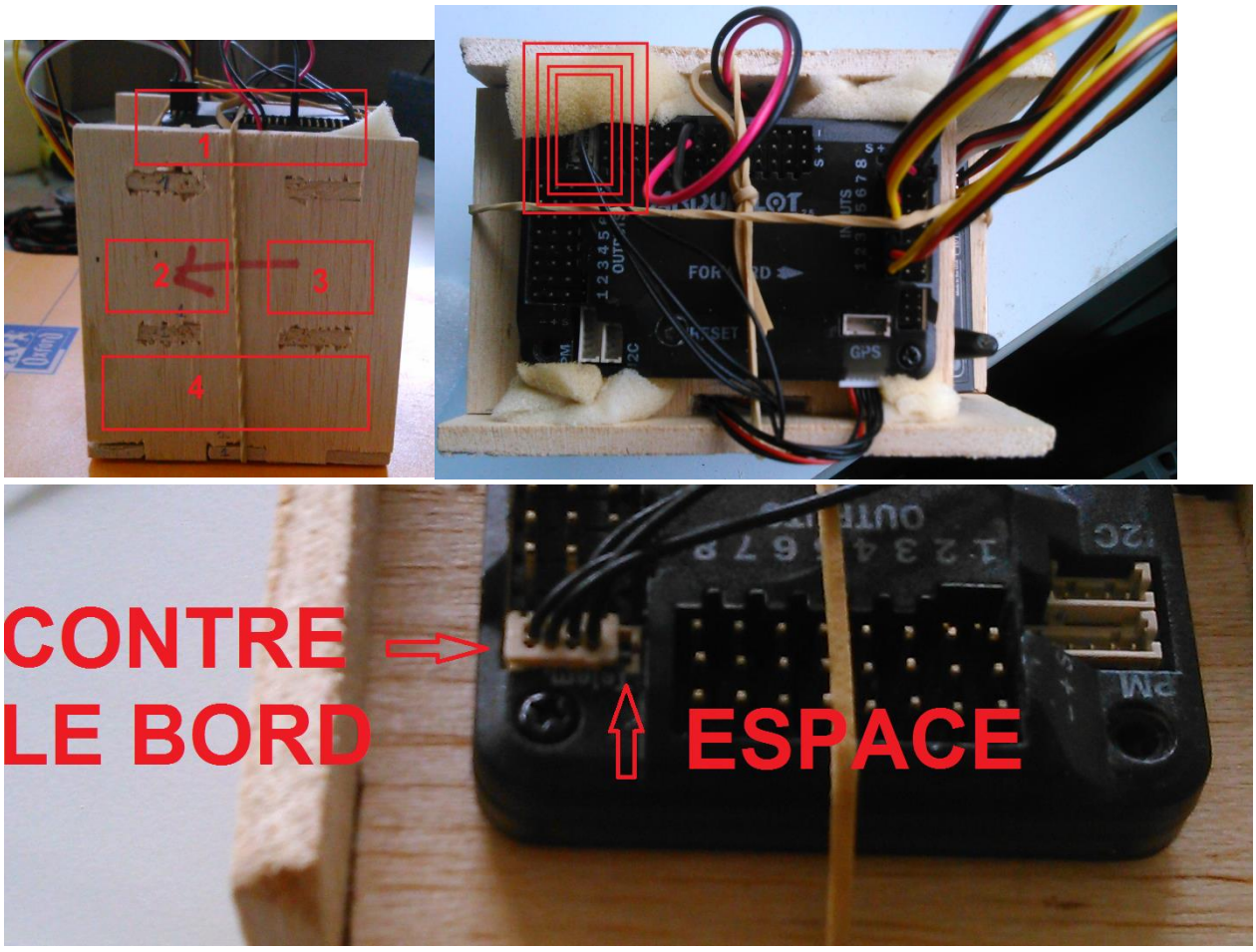
Fournit la tension aux servomoteurs branchés sur l'output de l'ArduPilot

8 – Power module

9 – interrupteur (optionnel)

10 – batterie 3S ou 4S

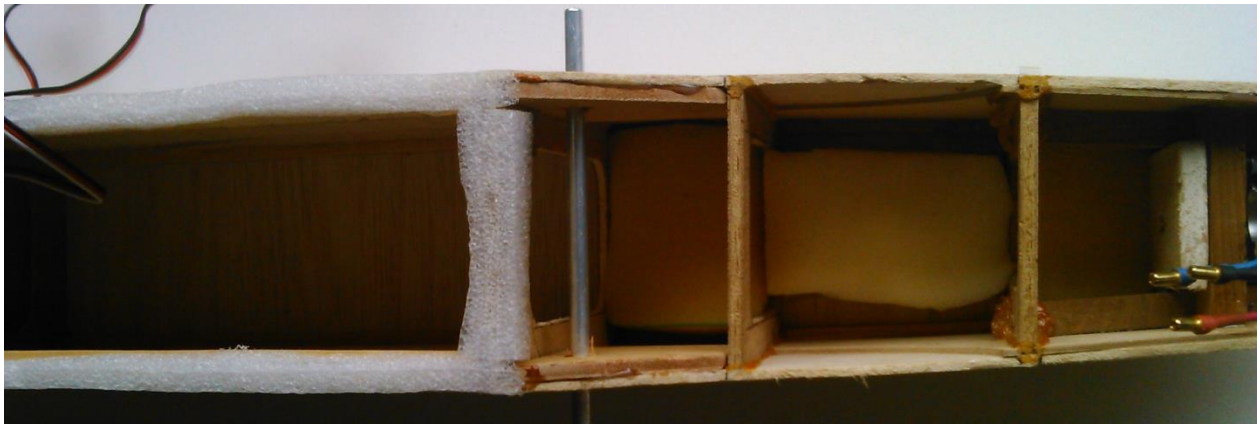
Boitier :



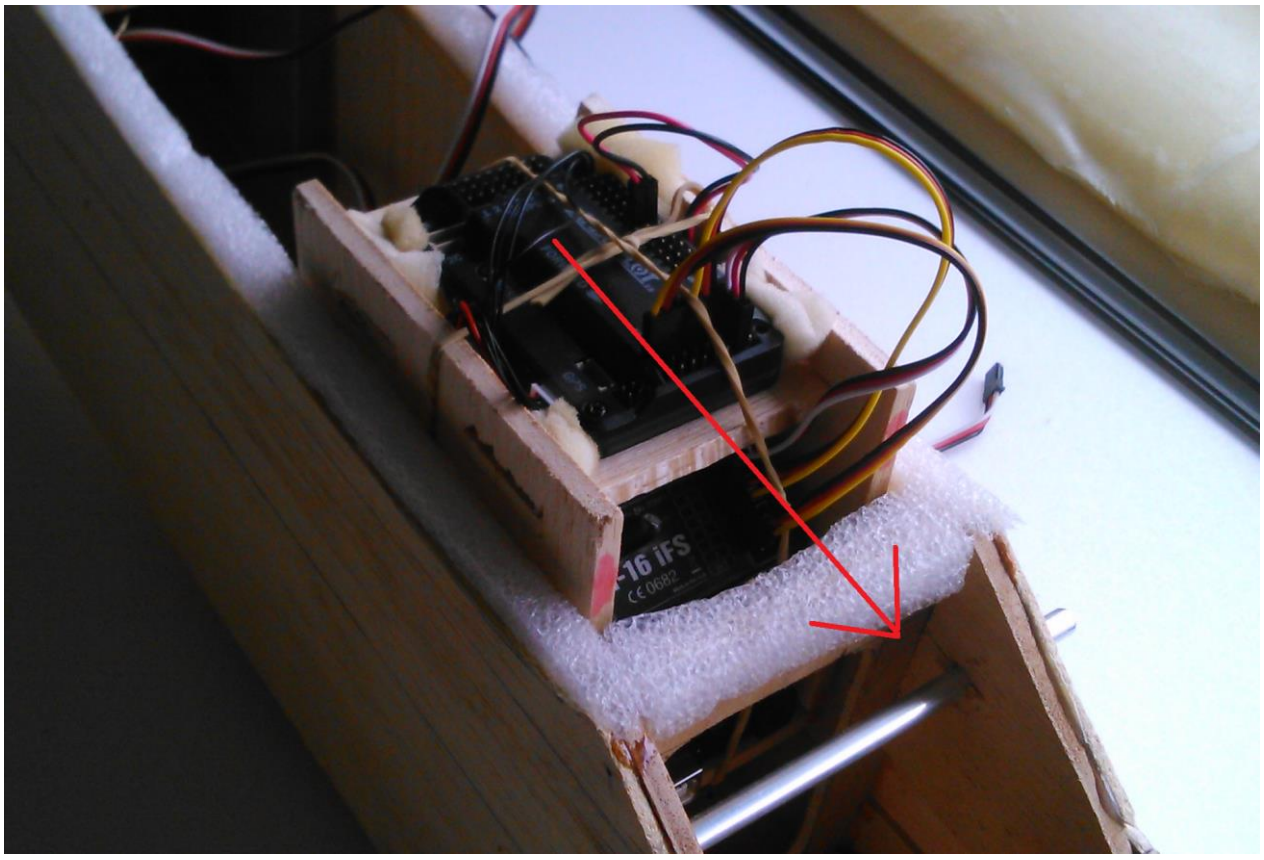
1	ArduPilot
2	Récepteur
3	GPS
4	ArduMega+Shield+xbee

Montage complet avion :

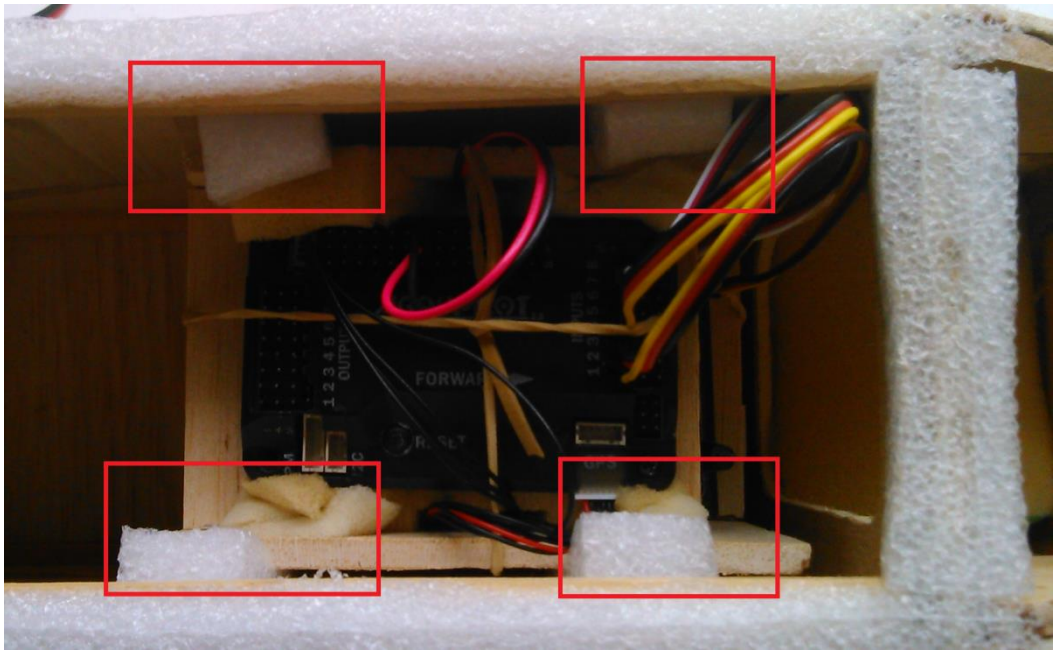
1- on commence avec le fuselage vide :



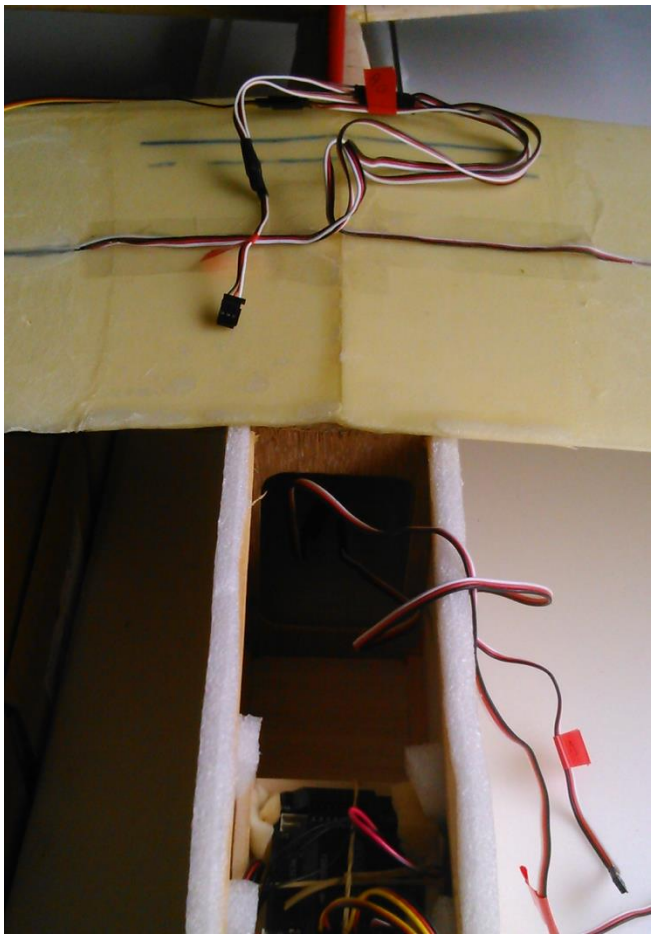
2- on insère le boitier. ATTENTION au sens (flèche forward inscrite sur l'ArduPilot, vers l'avant)



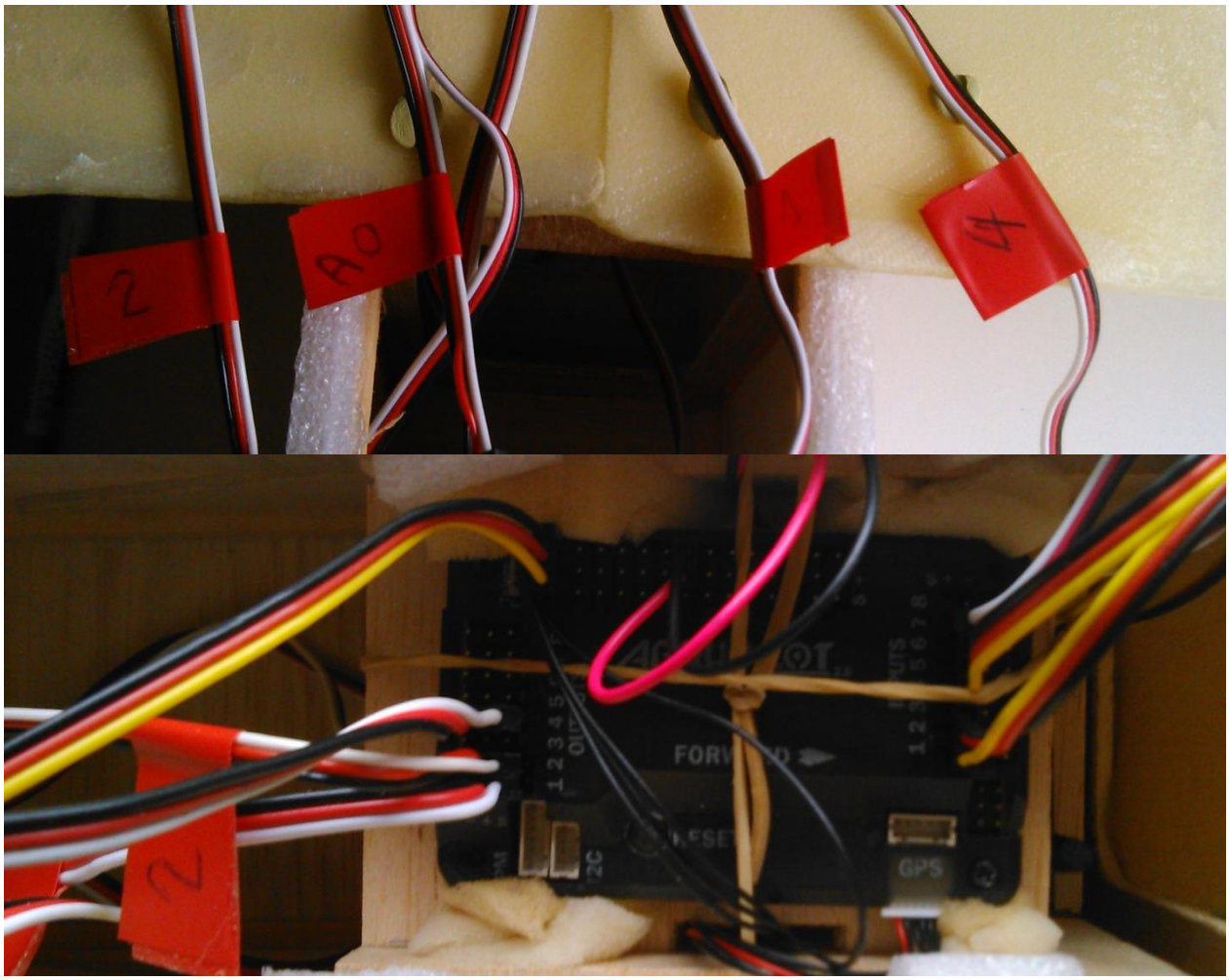
3- on insère des calles en mousse pour bloquer le boitier



4- on pose les ailes ‘’retournées » sur le fuselage. Bord de fuite vers l’avant & intrados vers le haut

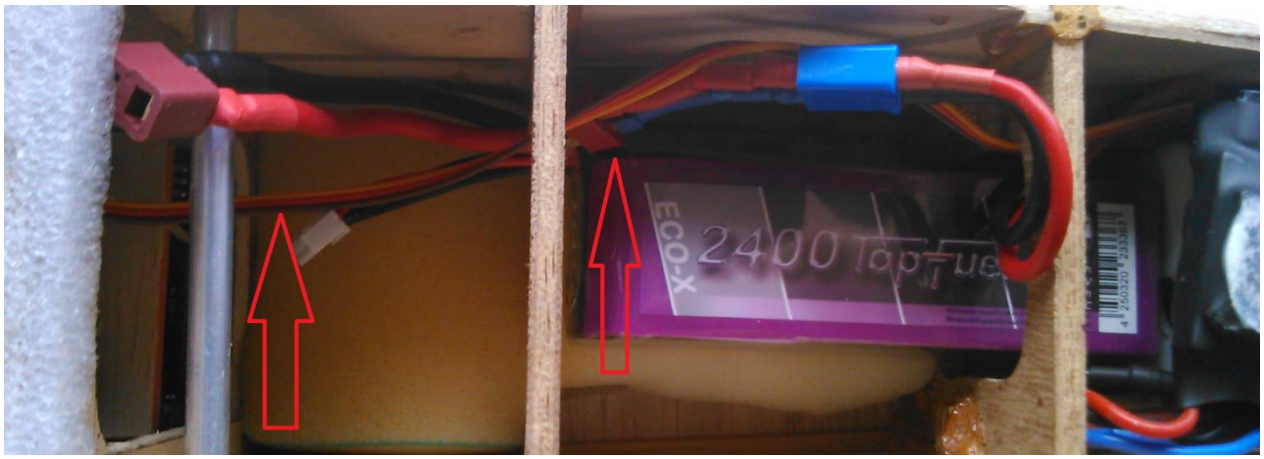


5- Connecter les câbles sur l'ArduPilot

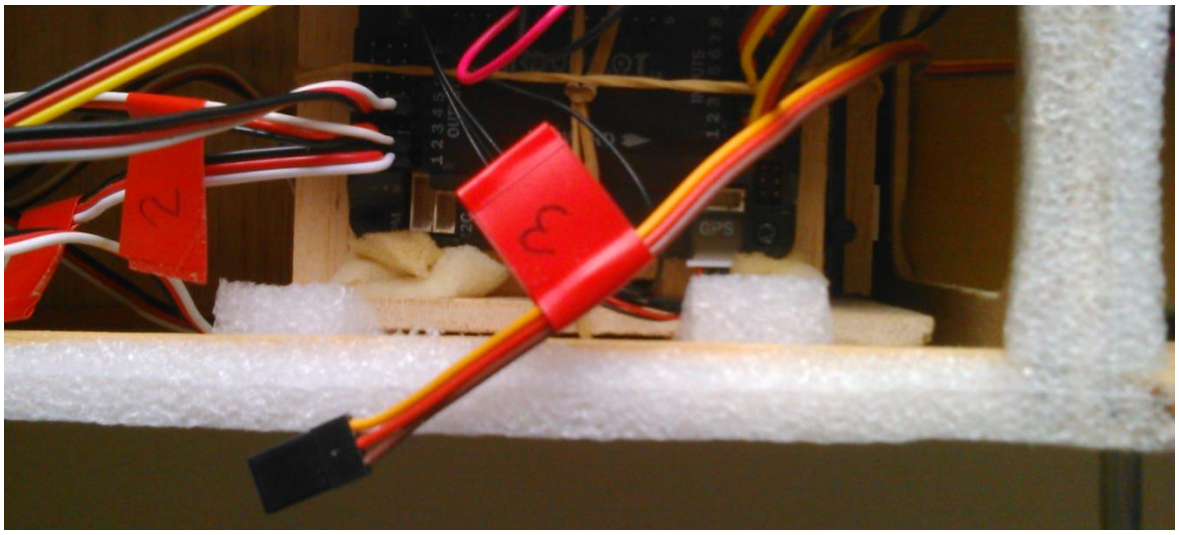


6- Placer la batterie et les masses pour faire le centrage (environ 5 masses)

7- Brancher le contrôleur Brushless au moteur et faites remonter les câbles (alimentation et BEC-PWM vers l'arrière)

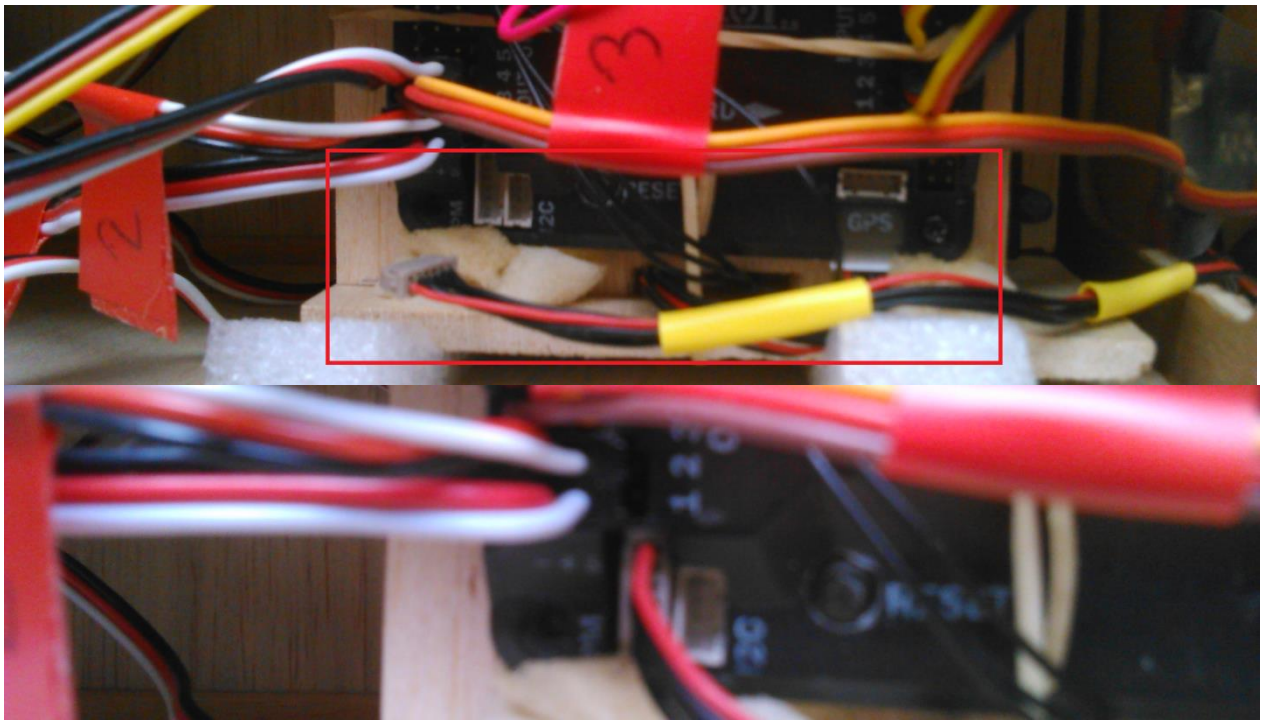


8- Brancher le câble PWM venant du contrôleur sur l'output 3

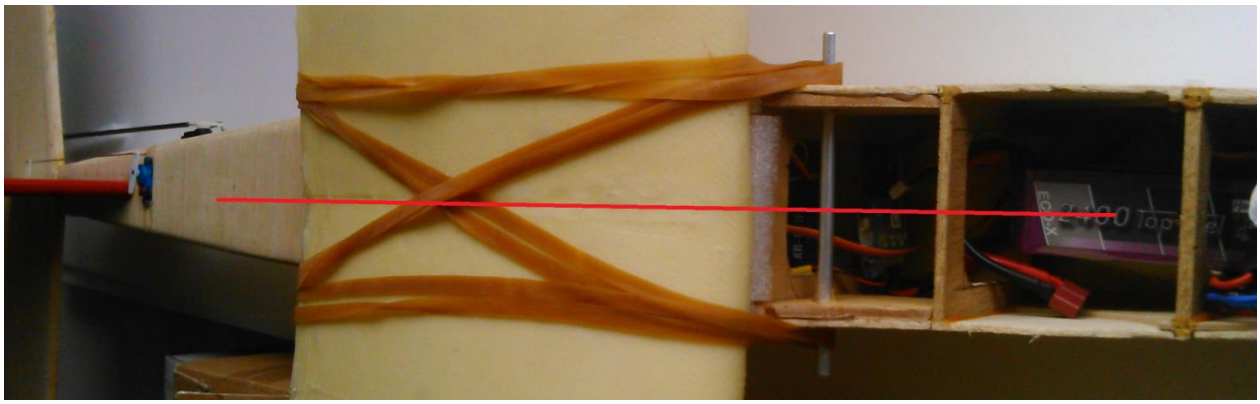


9- connecter le power module et le contrôleur brushless/ faire monter le câble d'alimention pour l'ArduPilot

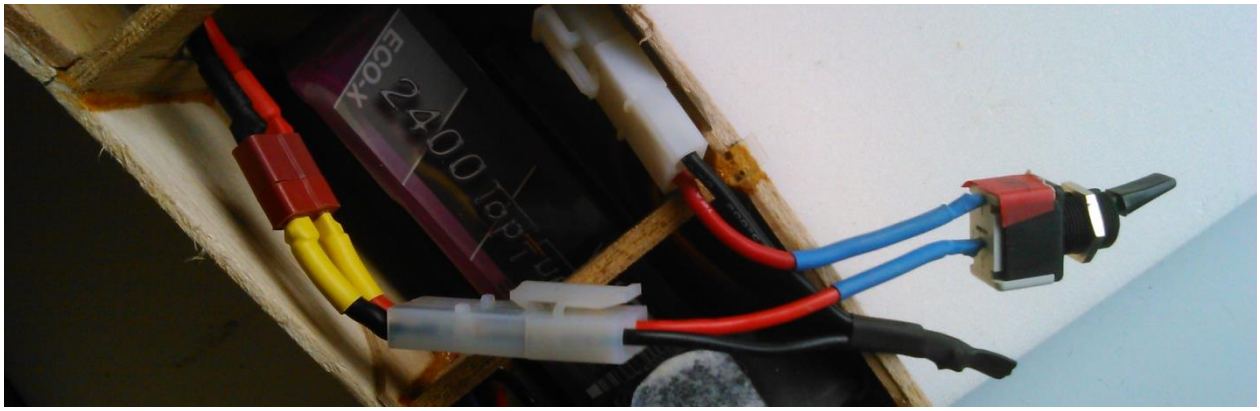
10- connecter le power module et l'ArduPilot



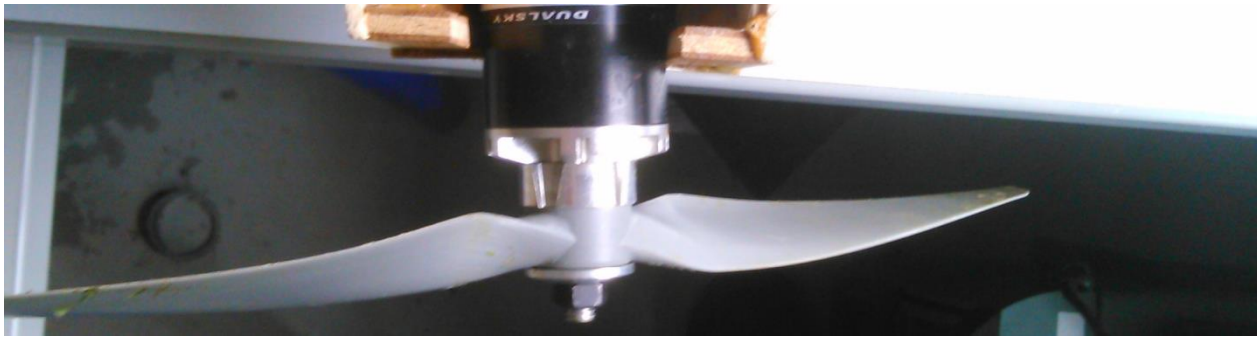
11- Retourner les ailes et les fixer /respecter la sysmetrie



12- Brancher l'interrupteur entre le power module et la batterie (interrupteur sur off)



13-fixer l'hélice



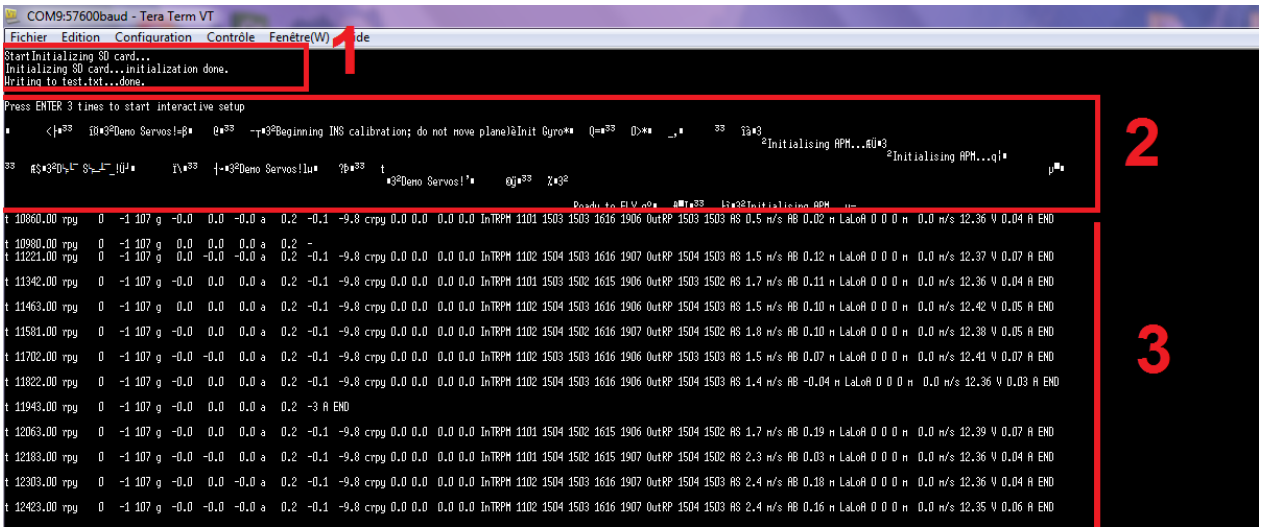
14-fixer le capot

15-Pour tester si l'ArduPilot répond et si l'enregistrement des données s'effectue :

Régler teraterm,

Mettre le manche des gazs sur 0,

Mettre l'interrupteur sur ON,



En 1 on peut voir l'état de la carte SD. (Si ''failed ''=> problème)

En 2 on peut voir des portions de phrases (le problème vient du fait qu'à ce moment-là, l'ArduPilot communique en 115200 et que nous sommes en 57200) ces données ne sont pas utiles.

En 3 on peut voir les données que l'on enregistre. Certaines données sont manquantes (pertes quelque part dues à la connexion sans fil)

Si rien ne s'affiche alors des câbles sont mal branchés.

Si la connexion à la carte SD échoue alors la carte SD est soit absente ou mal connectée.

Si le contrôleur brushless émet un BIP-BIP.. pendant plus de 4 sec le manche des gazs n'est pas à 0.

<u>Avant le décollage :</u>

Vérifier le centrage.

Vérifier que les switch 2, 3 et 6/7 sont :

Vers le bas pour 3 et 6/7

Vers l'avant pour 2

Faire le zéro du cap :

Orienter l'avion vers le Nord

Passer le bouton SW2 vers l'arrière et le laisser dans cette position

/!\ Il se peut que lors de l'initialisation le cap ne soit pas correcte ! /!\

/!\ Cependant il se recale (brutalement) lors du vol (le 0 indique bien el nord)/!\

Faire le zéro ne semble pas être utile si on laisse le temps au compas de se corriger

Effet des switch :

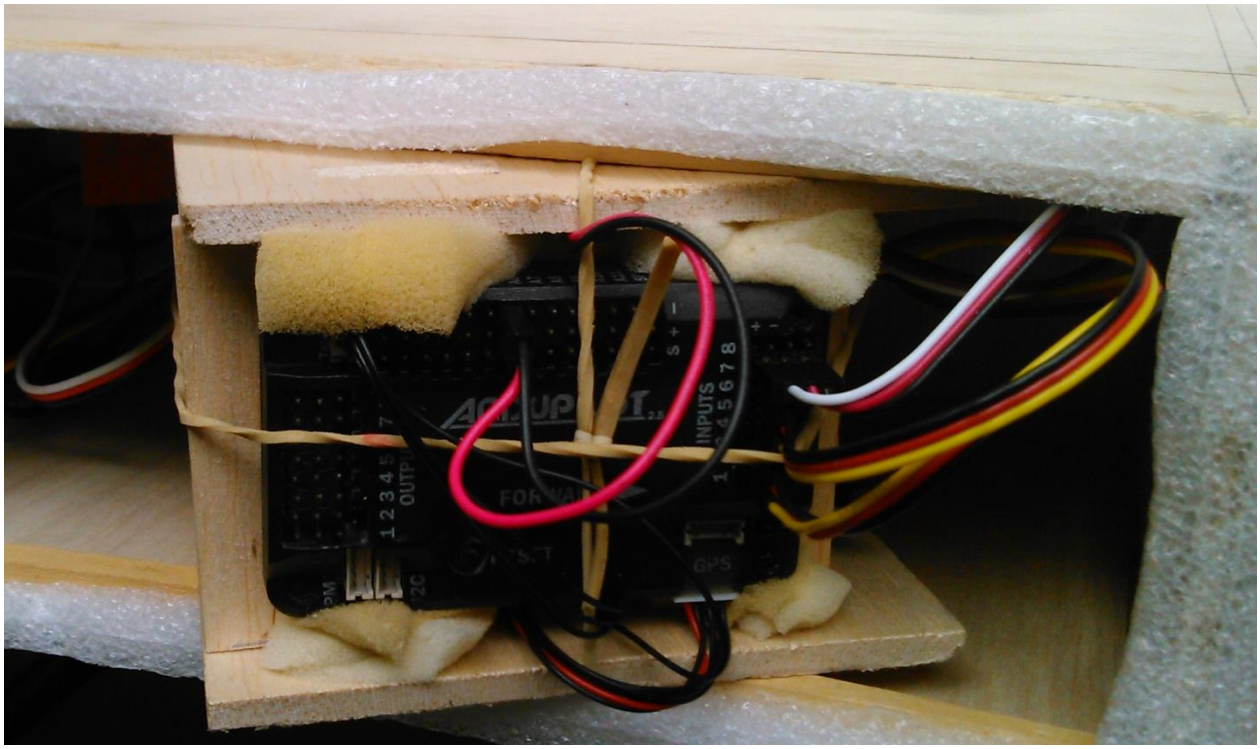
Switch 6/7	effet
Bas	Mode Manuel
Milieu	Joystick droit -mouvements latéraux : permet de virer à droite/gauche (Permet de tourner en pallier en limitant l'angle de roulis)
	Joystick gauche – mouvements haut/bas : permet de piquer cabrer avec des angles limités (Bloque l'angle de piqué et cabrage)
Haut	Joystick droit -mouvements latéraux : permet de changer le cap
	Joystick gauche – mouvements haut/bas : permet de donner une hauteur

Switch 3	effet
Bas	Retour du mode navigation au mode Manuel
Haut	Passe en mode navigation

Switch 2	effet
Bas	Rien
Haut	Enregistre le cap vers lequel l'avion pointe pour faire un « zero »

Sortir le boitier :

Après avoir débranché les câbles, faites sortir un coin du boitier.



Poussée le dessous de la boite pour la faire sortir.

Mises en Garde :

1- Centrage Avant
2- Durée de Vie Batterie LiPo
3- Ne pas s'improviser pilote
4- Règlementation
5- Charge utile et contrôleur
6- Plateforme saine

1-Centrage Avant

Si ce réglage est mal fait => Crash

Il faut que l'avion ait un centrage Avant après l'avoir monté complètement (électronique, batterie, charge utile, aile,….).

Pour vérifier le centrage, placer ses deux index au quart de la corde de l'aile, de part et d'autre du fuselage. Il faut que l'avion pique du nez.

Si ce n'est pas le cas, rajouter des masses.

2-Durée de vie des batteries lipo

Si on n'en prend pas soin => détérioration

Si les batteries lipo descendent en dessous d'une certaine tension elles sont détériorées (capacité et intensité max qu'elles peuvent fournir sont réduites et elles gonflent).

Par exemple pour une batterie :

4S (nombre de cellules qui fournissent la tension)
20C (taux de décharge maximum. Ici 20*2.400 = 48A max)
2400maH (capacité. Ici 2.4Amperes en 1heure)

On a :

La tension chargée max est 4*4.2 = 16.8V
La tension minimum limite à ne pas dépasser 4*3.7 = 14.6V
Capacité totale 2.4A * 14.6 = 35.5Wh
Si l'avion utilise 200Wh alors on a une autonomie théorique de 60*35.5/200= 10minutes

⇨ On prend un marge de sécurité : durée de vol de 6min avec les batteries 4S et le moteur à 30% de sa puissance soit 200W environ

Une batterie 4S lipo complétement déchargée aura une tension de 14.6 mais sa capacité sera proche de 0mah. Ce n'est pas parce qu'il y a de la tension qu'elle est chargée.

3- Ne pas s'improviser pilote

Même si vos tuteurs vous laissent la possibilité de piloter, ne le faite pas à moins que vous ayez une bonne expérience en modélisme.
Même avec un entrainement de 10h sur simulateur, vous n'êtes pas apte à assurer la sécurité de l'appareil et celle des personnes autour de vous.
Le passage du simulateur à la réalité est compliqué car :

- Vous ne savez pas comment l'avion va réagir
- La vision en réel et sur écran est très différente
- Cependant cela vous apprend les réflexes (inversion des commandes…)
- La pratique avec un moniteur reste la meilleure façon et la plus sure

4- Règlementation :

http://www.airshoot-technologie.com/contents/fr/d66_reglementation-drone.html
Catégorie D-E
Le drone est un aéronef et non aéromodèle.
Aéromodèles : aéronef télépiloté utilisé exclusivement à des fins de loisir ou de compétition par un télépilote qui est à tout instant en mesure de contrôler directement sa trajectoire pour éviter les obstacles et les autres aéronefs.

5-Charge utile et contrôleur :

Le contrôleur utilisé est régler pour un vol autour de 10m/s qui correspond à une puissance moteur de 200W.

Si vous voulez aller plus vite il faut réduire les gains.

Le moteur est puissant et les ailes ont une grande surface alaire. Vous pouvez ajouter de la charge utile comme une deuxième batterie ou un téléphone (tant que l'avion reste en dessous des 3.5kg).

6-Plateforme saine :

Cette avion vole bien mais a beaucoup de defaults. L'aide de pilotes confirmés augmente considérablement la durée de vie de l'appareil. Je conseille donc d'acheter un appareil du commerce qui est ''sain''.

L'université du Minnesota dispose de deux avions dont elle met à disposition les modèles mathématiques et simulateur avec les coefficients aérodynamiques extraits des essais en soufflerie et essais en vol. L'achat du même avion permet donc d'avoir :

- Une plateforme saine.
- Une représentation mathématique fidèle qui peut être utilisée pour élaborer un contrôleur.