ENSICA

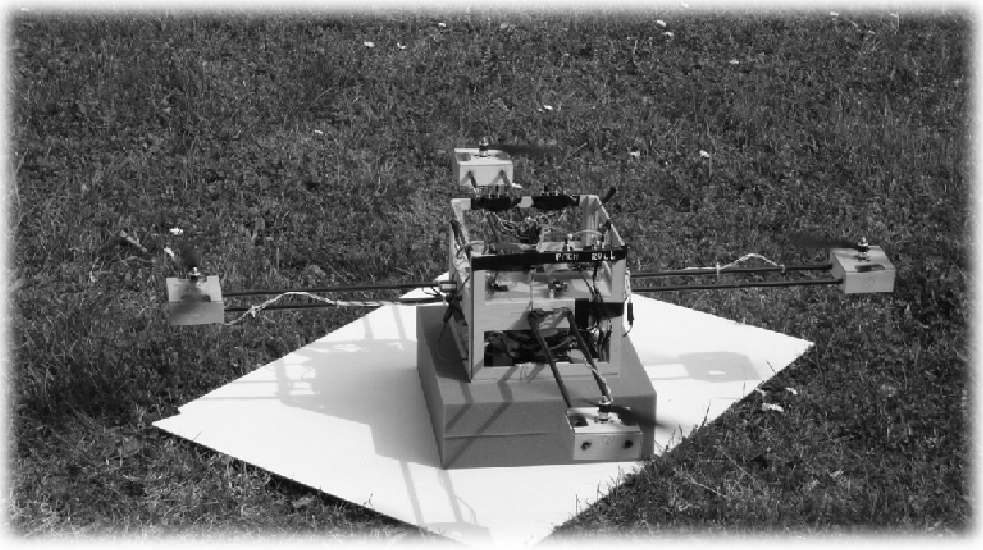
2006-2007

STABILISATION D'UN QUADRIROTOR

Adeline Serrecourt

Julien Chapuis

Tuteur : Yves Brière



SOMMAIRE

PRÉSENTATION DU PROJET

INTRODUCTION

Depuis quelques années à l'ENSICA, de nombreux groupes d'élèves ont tenté, dans le cadre de leur PIP, de réaliser des microdrones.

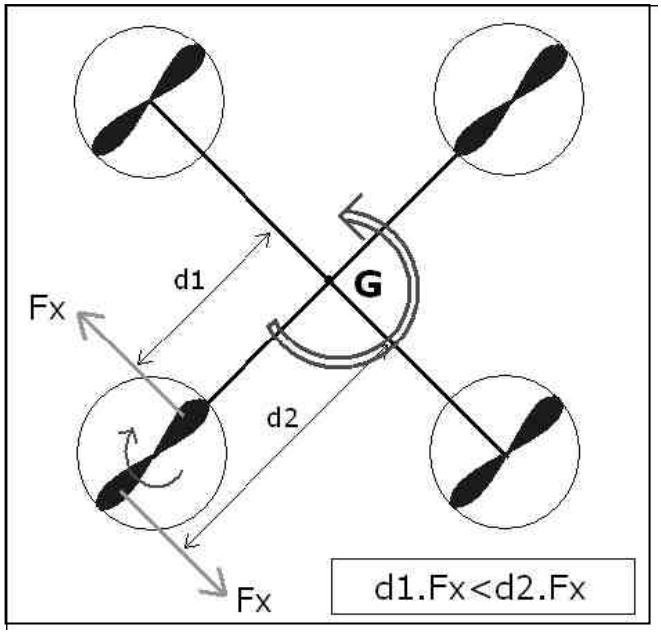
En 2005-2006, Alexis FRENOT, Anthony GOSSMANN et Romaric GUILLERM ont eu l'idée de développer un drone quadrirotor. Ils ont donc fabriqué la structure d'un hélicoptère à quatre rotors, à laquelle ils ont intégré des moteurs et les équipements nécessaires à leur fonctionnement. Puis, ils se sont intéressés à la manière de stabiliser au mieux cet engin. Une seule méthode a été jugée satisfaisante : l'utilisation d'une carte Micropilot. En effet, cette carte est très peu encombrante et légère, étant donné toutes les options qu'elle propose, et le matériel de qualité dont elle est pourvue. De plus, couplée à un GPS, elle permettrait au quadrirotor d'effectuer des vols autonomes. Ils ont alors commencé la stabilisation du drone autour de ses axes de tangage et de roulis. Cependant à la fin de leur PIP, le travail de prise en main du logiciel de la carte Micropilot et de stabilisation était loin d'être terminé.

Nous avons donc décidé de prendre le relais. En début d'année, nous nous sommes fixés pour objectif de réaliser la stabilisation et la commande du drone autour de ses trois axes. Notre PIP s'est déroulé en cinq étapes. Tout d'abord, il nous a fallu prendre en main la carte Micropilot, nous avons ensuite pu écrire un code de commande, et élaborer un banc de test. Puis nous avons cherché de manière empirique les meilleurs coefficients possibles pour les PID contrôlant la stabilité autour des axes de roulis et de tangage. Enfin, lorsque nous avons considéré que notre stabilisation était satisfaisante nous avons voulu la tester en vol. Malheureusement, nous avons alors découvert que le quadrirotor n'était pas suffisamment puissant pour se sustenter.

Vous découvrirez dans ce rapport tous les détails concernant les différentes étapes de notre travail. Nous avons tenu à détailler particulièrement le fonctionnement de la carte Micropilot, de son logiciel et les branchements nécessaires à la mise en route du quadrirotor. De même, tous les coefficients qui interviennent dans le code sont expliqués. Nous proposons également une nouvelle structure plus efficace et un dimensionnement des moteurs et hélices. En effet, nous souhaitons vraiment qu'une nouvelle équipe, l'an prochain, puisse reprendre le projet, réutiliser facilement nos méthodes de stabilisation et parvienne, finalement, à faire voler notre quadrirotor.

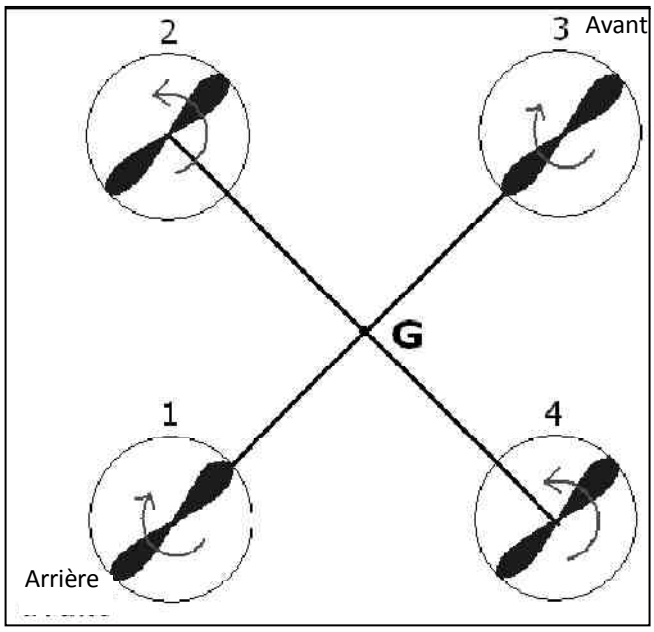
SENS DE ROTATION ET MOUVEMENTS POSSIBLES

Un quadrirotor possède, comme son nom l'indique, quatre rotors pour se sustenter. Pour contrer un mouvement de lacet, il est nécessaire de faire tourner deux hélices dans un sens et les deux autres dans l'autre sens. En effet, lorsqu'on projette les forces aérodynamiques exercées par l'air sur la pale, on s'aperçoit qu'un rotor a toujours tendance à faire tourner le quadrirotor dans le sens inverse de sa rotation.



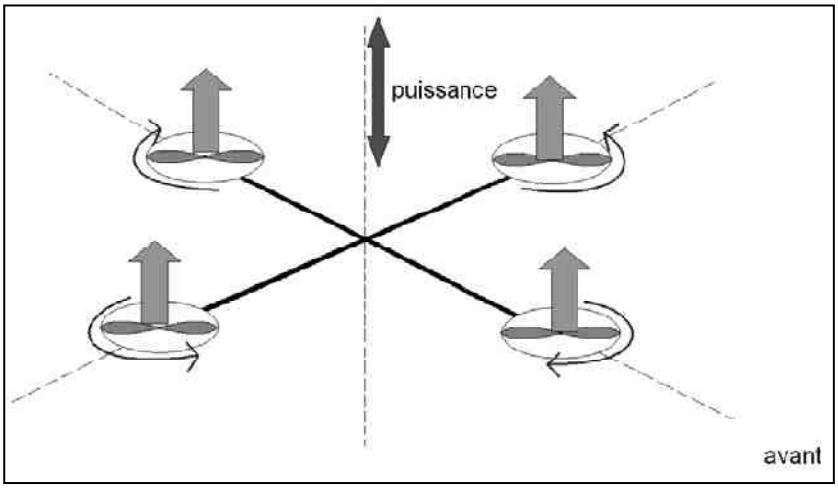
Moment de lacet

De plus pour faciliter la commande en tangage et roulis, on choisit l'avant du quadrirotor au niveau d'un rotor (et non pas entre deux rotors), et on fait tourner les moteurs qui sont face à face dans le même sens.

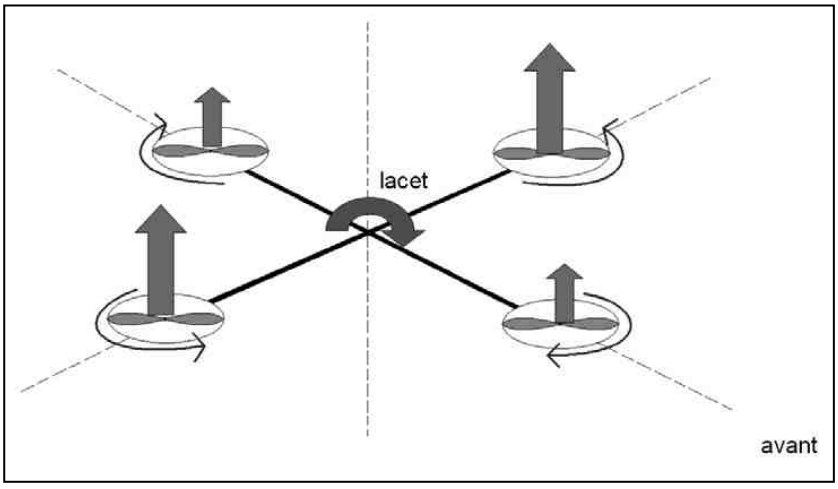


Sens de rotation des moteurs

Les commandes des quatre moteurs seront utilisées pour modifier l'accélération verticale (commande de puissance) et pour faire tourner le quadrirotor autour de son axe vertical (commande de lacet).

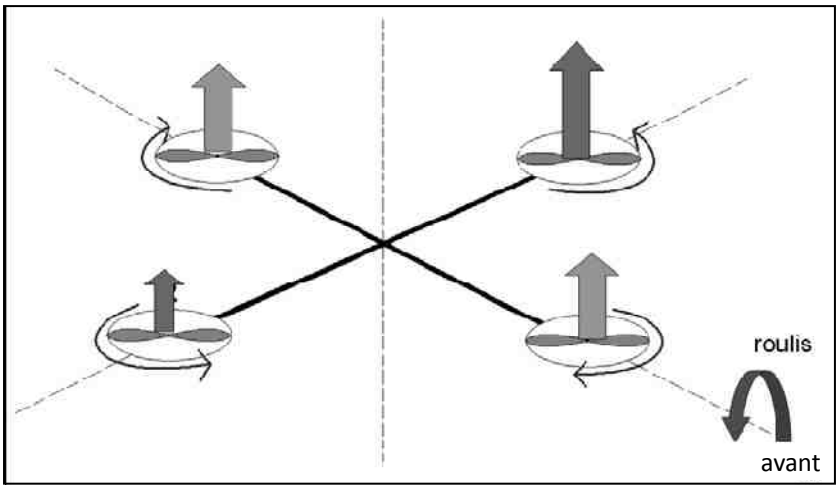


Commande de puissance

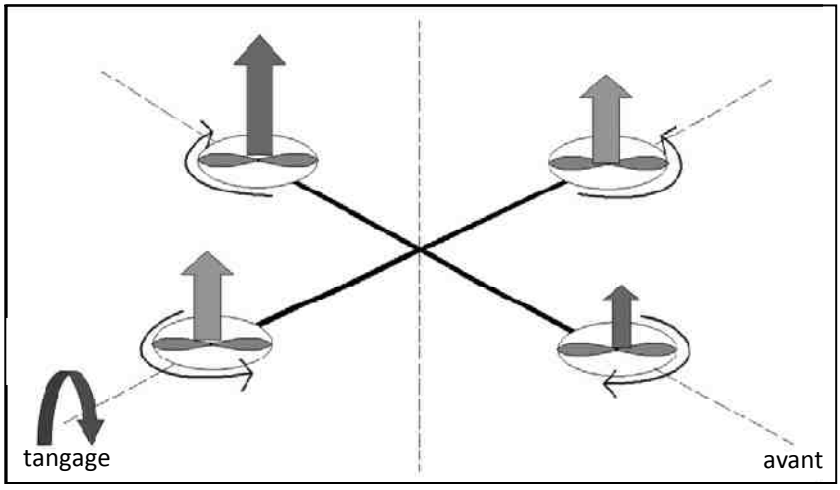


Commande de lacet

Pour obtenir la rotation autour de l'axe longitudinal (commande de roulis), il suffit de commander deux moteurs opposés. De la même manière, la commande des deux autres moteurs permet une rotation autour de l'axe latéral (commande de tangage).



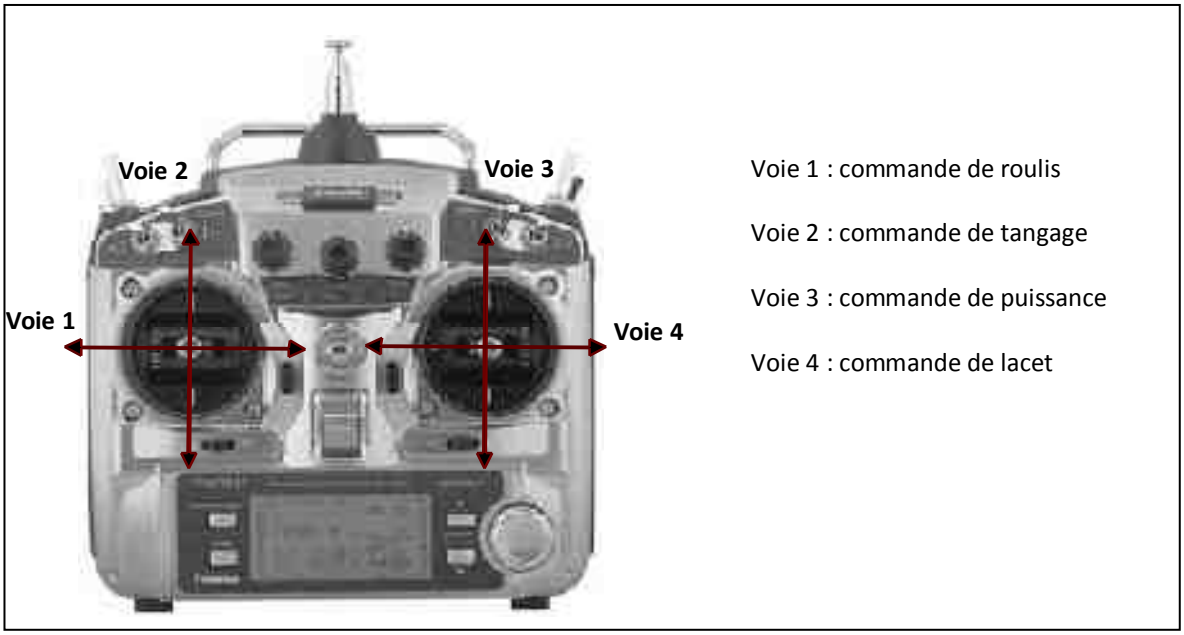
Commande de roulis



Commande de tangage

LA RADIOCOMMANDE: COMMANDE SOUHAITÉE SUR CHAQUE VOIE

La radiocommande utilisée est une radio Futaba (réf : 9CAP/9CHP) de neuf voies. Nous n'en utilisons que quatre. La représentation suivante fait apparaître différentes voies utilisées et les commandes auxquelles elles correspondent.



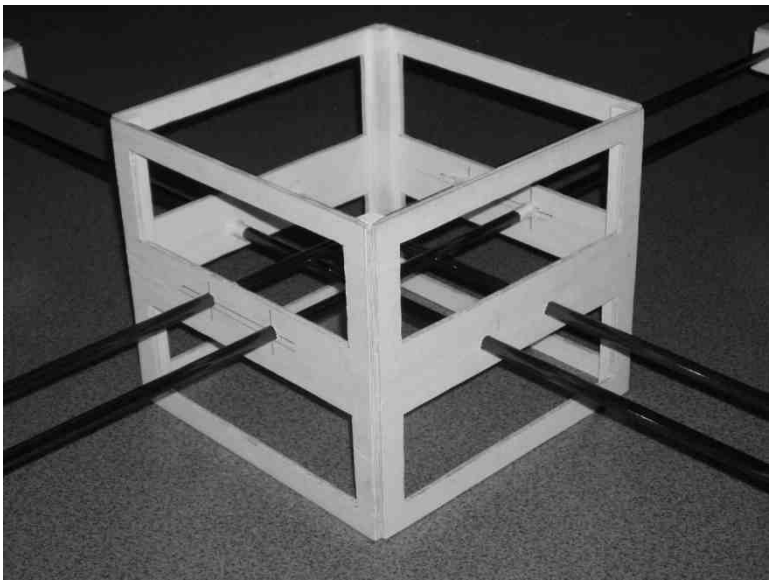
Radiocommande

PRÉSENTATION DE LA STRUCTURE QUADRIROTOR

La structure du quadrirotor est constituée de trois éléments : la partie centrale, les quatre barres de carbone et les supports moteurs.

LA PARTIE CENTRALE

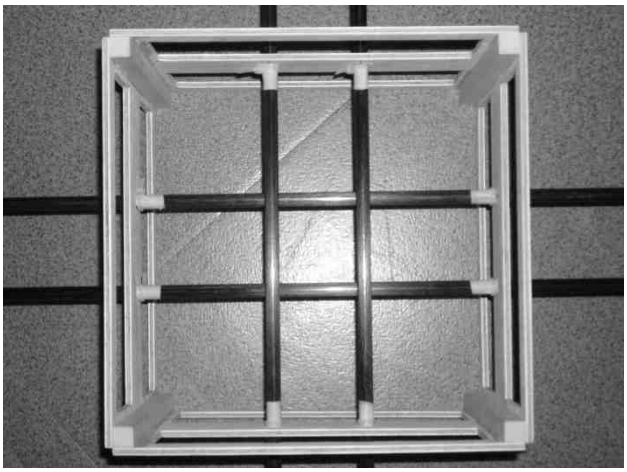
Elle a été réalisée en contre-plaqué de 4 mm d'épaisseur et se présente sous la forme d'un cube très largement évidé. Cette partie comporte deux compartiments : l'un au niveau des moteurs et l'autre en dessous. Dans le compartiment supérieur, sont fixés le récepteur, la carte MicroPilot, le bloc servo board et tous les interrupteurs permettant la mise en marche du drone. Le compartiment inférieur servira à transporter toutes les batteries (moteurs, carte MicroPilot et récepteur).
Toutes les liaisons ont été réalisées par collage en utilisant des renforts de balsa pour rigidifier le tout.



Structure centrale du drone

LES BARRES DE CARBONE

Il s'agit de quatre tubes de carbone de 1m de longueur et de 8 mm de diamètre. Ces tubes ont été placés de manière à ce que deux d'entre eux soient au dessus des deux autres, le décalage de 8mm étant rattrapé au niveau des supports moteurs. Les tubes sont liés à la structure centrale à l'aide de colliers en plastique, ceci empêche toute translation, et collés à celle-ci à l'aide de colle EPOXY pour éviter toute rotation.



Liaison structure centrale/ barres de carbone

LES SUPPORTS MOTEURS

Comme nous l'avons vu précédemment les barres de carbone ne sont pas toutes à la même hauteur, il faut donc que deux des supports moteurs se lient aux barres de carbone, 8 mm plus haut que les deux autres. De plus pour augmenter la tendance du drone à revenir dans sa position d'équilibre, les rotors sont inclinés d'environ 2° vers le centre du drone. C'est pour cela que la partie supérieure des supports est inclinée. Les liaisons entre les supports moteurs et les barres de carbone sont identiques à celles entre les barres de carbone et la partie centrale.

Support moteur

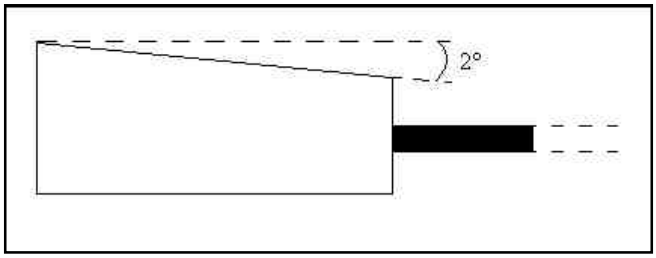
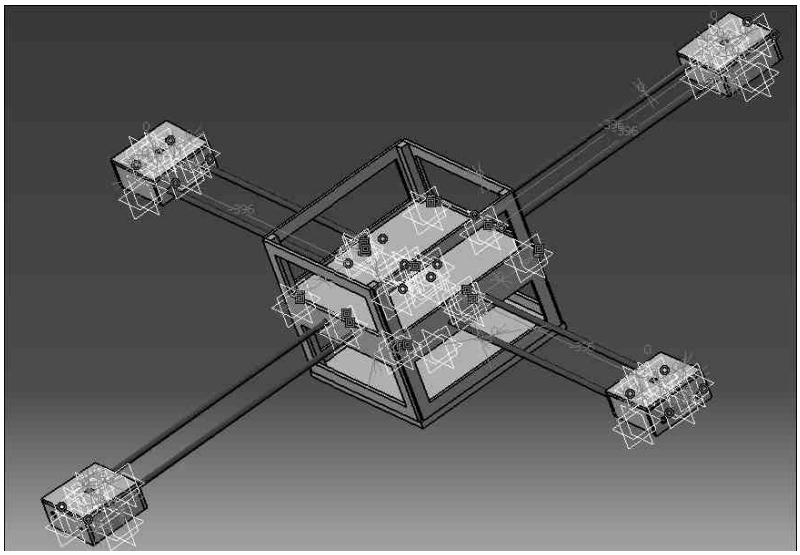


Schéma de l'angle support moteur

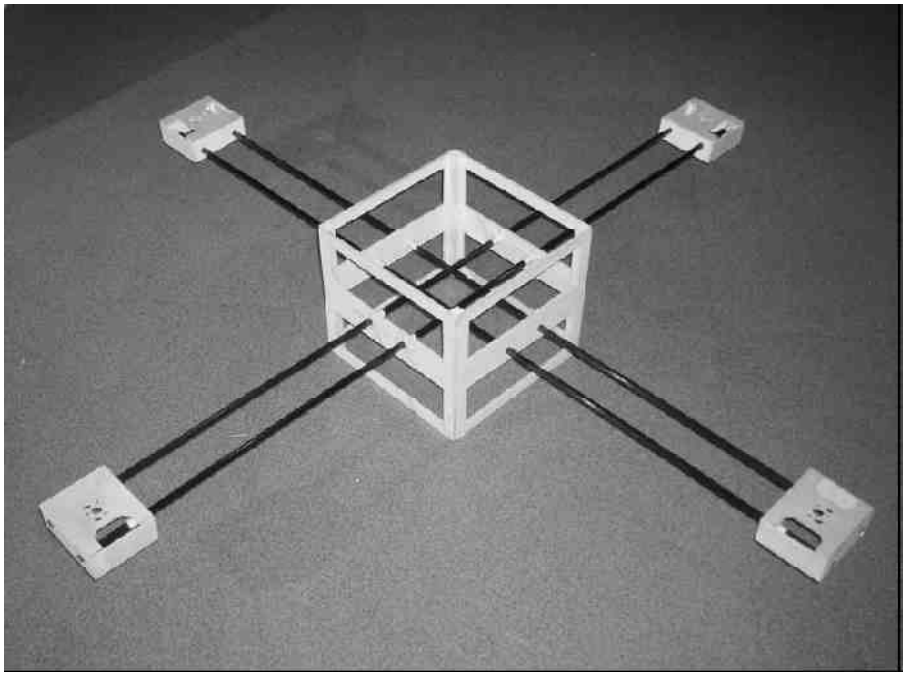
La partie supérieure des supports moteurs est évidées. Cela permet d'une part au flux d'air provenant des hélices de ne pas « heurter » la plaque de bois ce qui créerait une force vers le bas et donc une perte de portance. D'autre part, cela permet une aération des moteurs, pour éviter qu'ils ne chauffent trop.



Conception CATIA de la structure

AVANTAGE DE CETTE CONCEPTION

Cette conception a l'avantage d'être très rigide tout en étant relativement légère : 400 g au total. La rigidité en flexion offerte par les barres en carbone est très bonne et la torsion est quasiment inexistante grâce au collage. De plus, la plaque supérieure en contreplaqué du compartiment supérieur étant amovible, il nous a été possible de travailler sur la carte MicroPilot en dehors du drone.



Photographie de la structure

DESCRIPTIF DU MATÉRIEL UTILISÉ

LE RÉCEPTEUR ET LA RADIOCOMMANDE

La radiocommande et le récepteur utilisés nous ont permis de faire le lien entre le pilote et le système de commande du quadrirotor.

La radiocommande dont nous nous sommes servis est une radiocommande Futaba 9CAP/9CHP. Elle possède neuf voies. Cette année nous n'en avons utilisé que quatre. Cependant, les cinq autres voies pourraient être utilisées, si le projet est prolongé par de nouvelles équipes, au cours des prochaines années.

Radiocommande Futaba 9CAP/9CHP

Récepteur Graupner R700

La radiocommande a été associée à un récepteur Graupner R700.

LES PALES ET LES MOTEURS

Le principe de vol du quadrirotor impose d'avoir deux pales de pas opposés aux deux autres pales. Les hélices qui équipent le drone sont normalement utilisées sur des avions de voltige capables de rester immobiles à la

verticale. En effet, le groupe qui a construit le drone a préféré ne pas utiliser des pales d'hélicoptère proprement dites, car elles sont à pas variables et donc plus lourdes.

Les pales ont été associées à des moteurs de type brushless (réf : brushless typhon-micro 6-23), qui ont l'avantage d'être petits, légers et puissants. L'un des inconvénients de ces moteurs est leur mise en œuvre : ils doivent obligatoirement être associés à des régulateurs particuliers, appelés contrôleurs, qui nécessitent une programmation et une temporisation bien précise.

Paires d'hélices 10x4.5

Moteur Brushless typhon-micro 6-23

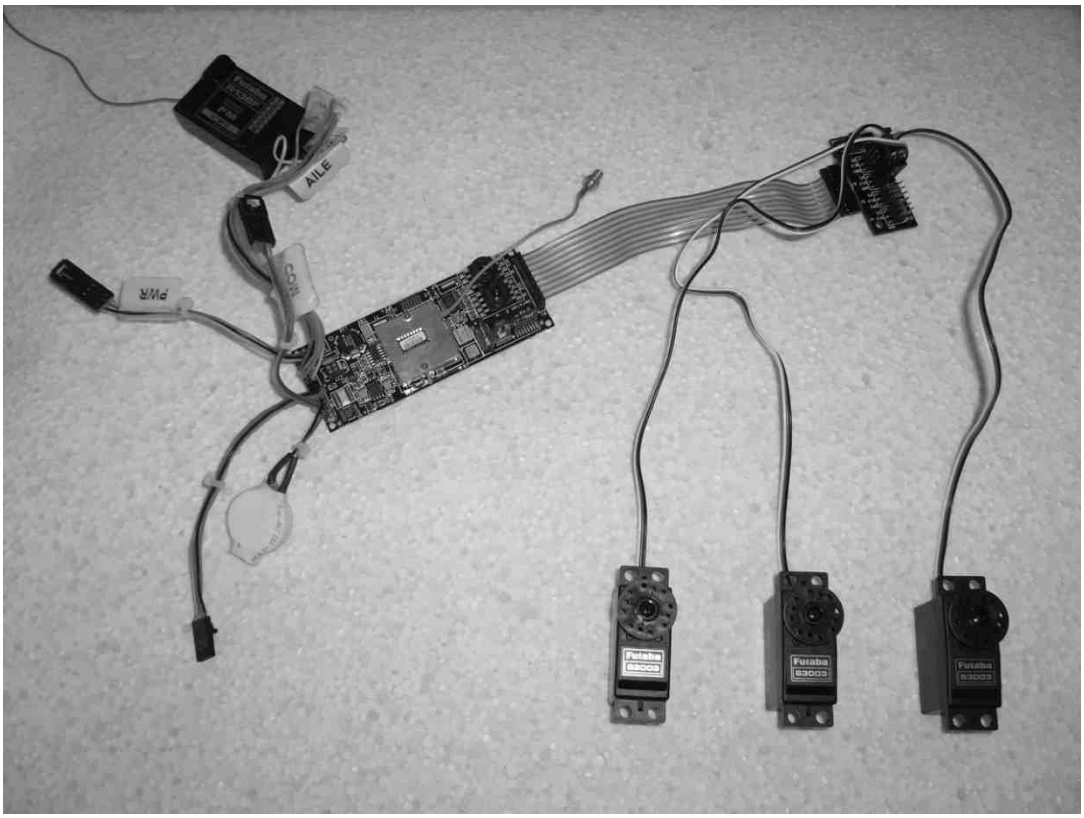
Ce modèle de moteur, de puissance 90 W, peut être alimenté sous tension variant de 7,2 V à 14,4 V. Il pèse 42g et, à la base, convient à un engin de 250 à 700 grammes.

LES CONTRÔLEURS

Un contrôleur fait l'interface entre le signal de commande et le circuit de puissance d'un moteur. Il est relié à un accumulateur qui fournira la puissance nécessaire au moteur. Quatre contrôleurs sont utilisés. Ils sont programmés en mode 1, mode par défaut qui permet une sélection automatique du nombre d'éléments de l'accumulateur. En raison de la chaleur importante dégagée par ces composants, les contrôleurs sont fixés à l'extérieur de la partie centrale, entre les tiges de carbone.

Contrôleur Wena-sinus-12-bec

LA CARTE MICROPILOT



Carte MicroPilot

Bien que la carte MicroPilot soit, à la base, programmée pour stabiliser et commander des drones de type avion, cette carte est très adaptable, et permet de créer son propre code de commande. Ainsi, nous avons pu stabiliser un quadrirotor. Cet élément est présenté de manière plus détaillé plus loin dans le dossier. (Présentation de la carte MicroPilot).

LES ACCUMULATEURS

L'accumulateur servant à alimenter le récepteur et le servo board est un accumulateur classique de 4,8 V et 800 mAh.

L'accumulateur servant à alimenter la carte MicroPilot est identique à celui qui alimente le récepteur et le servo board.

Accumulateur 4,8 V, 800 mAh

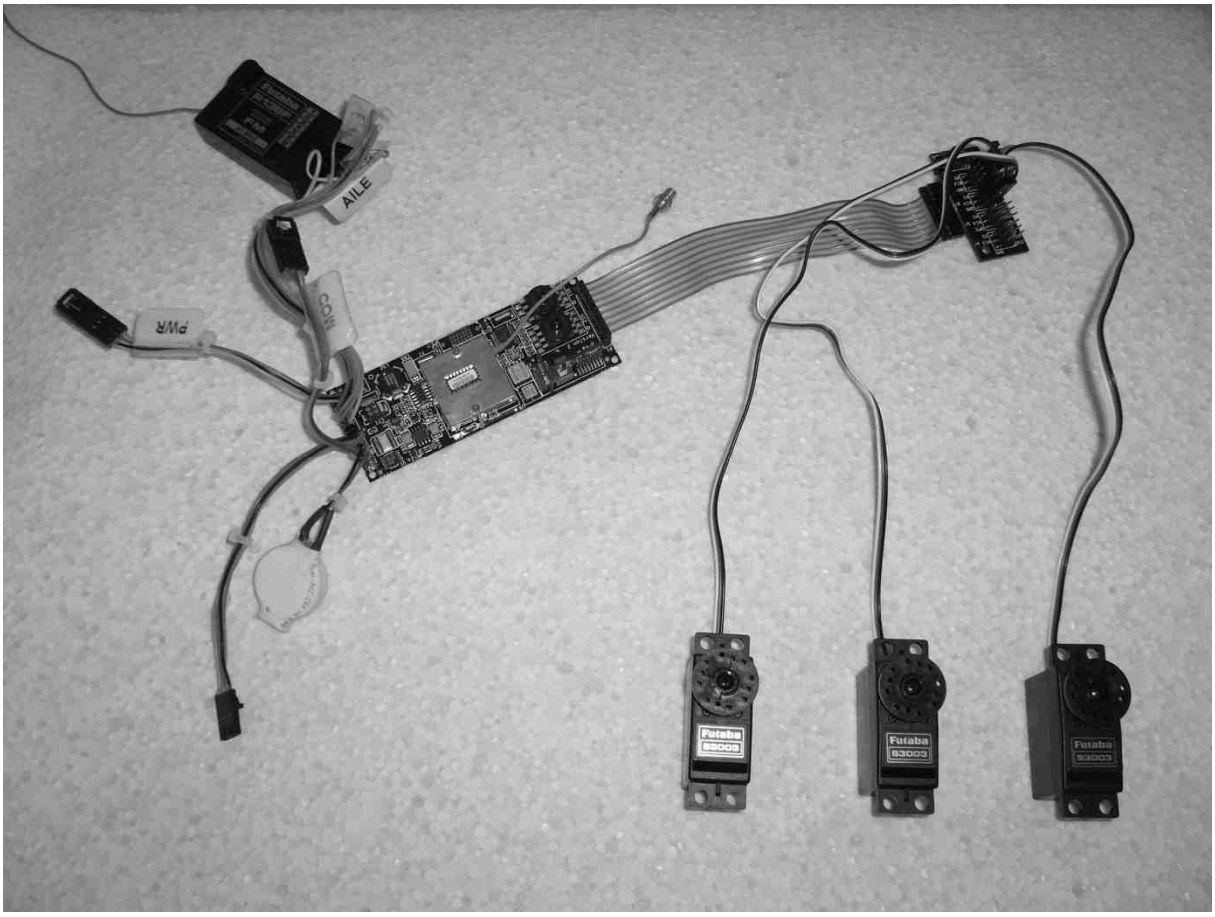
Pour alimenter les quatre moteurs, on utilise deux accumulateurs de type Li-Po à deux éléments. Lors des tests, ils étaient remplacés par une alimentation stabilisée puissante.

Accumulateur Li-Po à deux éléments

PROGRAMMATION DE LA COMMANDE

PRÉSENTATION DE LA CARTE MICROPILOT

Le drone quadrirotor sera stabilisé à l'aide d'une carte Micropilot MP 2028g. Il s'agit d'une carte conçue pour le contrôle de modèles réduits d'avions. Elle est composée d'un microprocesseur programmable, de systèmes de mesure (accéléromètres, deux prises de pression, GPS), et d'entrées sorties typiques du monde du modélisme (prises pour récepteur de télécommande, carte fille de sorties vers des servocommandes). Cette carte peut évoluer par l'adjonction de cartes filles qui lui donnent des capacités de gestion d'un flux vidéo, ou de communication vers un PC via un radio modem. On peut aussi lui ajouter d'autres cartes de sorties pour d'autres servocommandes.



Carte Micropilot reliée à un récepteur radio et à trois servocommandes

La communication avec un PC se fait par la sortie COM de la carte, qui est reliée au port série d'un ordinateur. On peut ainsi programmer la carte ou récupérer l'état des différents capteurs.

Pour le quadrirotor, nous allons détourner la carte de son usage initial –le contrôle des avions- pour une utilisation sur un engin plus exotique. Cette carte n'est pas spécialement prévue pour le contrôle d'un appareil au design si particulier. Nous devrons donc nous passer de nombre d'éléments préprogrammés.

Heureusement, il est possible de reprogrammer la carte pour piloter le drone, en utilisant le logiciel Xtender 3. Cependant cette solution n'est pas aussi pratique que la programmation d'un microcontrôleur. En effet, nous ne pouvons accéder qu'à une partie du code, et Micropilot conservera toujours un noyau non modifiable qui pourra perturber le bon fonctionnement du drone.

En résumé :

Avantages de la solution Micropilot :

- Solution tout intégré (accéléromètres + microcontrôleur + carte de sortie)
- Légèreté
- Réactivité

Inconvénients :

- Utilisation détournée de l'usage initial
- Impossibilité de contrôler totalement la carte
- Manque de documentation, obscurantisme de certaines fonctions
- Non évolutivité (impossible de rajouter des entrées/sorties non prévus)

Il semble y avoir de nombreux inconvénients, cependant il reste très difficile de réaliser une carte légère disposant d'une bonne puissance de calculs et d'accéléromètres de qualité. Nous allons donc utiliser cette carte pour le contrôle du drone.

La programmation se fera sous Visual C. La compilation et l'écriture du code dans la ROM du drone sont effectuées par le logiciel Xtender 3. Enfin, l'initialisation du drone et la lecture de données en vol, ainsi que le réglage des correcteurs sont effectués avec le logiciel Horizon. La lecture des datalog se fait avec le logiciel LogViewer.

PROGRAMMATION DE MICROPILOT

La carte Micropilot peut être programmée par l'utilisateur. Cette fonctionnalité correspond en fait à l'usage détourné des User Pids.

En effet, la carte dispose d'un code standard destiné à faire voler un petit avion de modélisme. L'utilisateur rentre dans la carte les données de son matériel et la carte permet alors d'en assurer le contrôle. L'utilisateur avancé peut cependant aller plus loin en réglant lui-même la dynamique de sa commande via les User Pids. Il s'agit de tronçons de code permettant de modéliser plus finement le modèle réduit et qui peuvent éventuellement être activés par l'utilisateur. S'ils sont activés, la carte exécutera le code qu'ils contiennent en plus de son propre code.

Notre démarche consiste à contrôler totalement le drone par ces User Pids, en ne nous servant pas du tout du code standard destiné à contrôler un avion. Nous ne rentrerons pas seulement dans ce code des coefficients préprogrammés de dynamique avion, mais un code complet de contrôle. Par ce moyen détourné, nous allons nous approprier le fonctionnement de la carte.

Pour cette raison, nous ne pouvons utiliser les sorties utilisées par le code standard (donc non modifiable) de la carte. Il s'agit des sorties servo 1,2,3 et 4. Comme nous n'utilisons que quatre sorties (une par moteur), nous allons brancher nos moteurs sur les sorties servo 5,6,7 et 8.

Tous les paramètres de fonctionnement de la carte sont stockés dans des variables d'état (le listing de ces variables est détaillé en annexe du manuel Micropilot). Ici, pour activer le User Pid 1 dans lequel nous allons placer notre code, nous éditons la variable EnableUserCode (n°588) :

EnableUserCode = 2

Ainsi que la variable d'activation du User Pid 1 (n°5451) :

EnableUserPid1 = 3

Le 3 signifie que ce User Pid sera exécuté à une fréquence de 30Hz.

Une description plus détaillée du réglage de ces paramètres est fournie en annexe.

De la même manière, dans le code, chaque paramètre du drone (entrée, sortie, état d'une variable) est contrôlé en référence à son numéro de variable, par exemple :

N° de variable	Alias	Description
1231	fServo5	Commande moteur 2
1232	fServo6	Commande moteur 3
1233	fServo7	Commande moteur 4
1234	fServo8	Commande moteur 1

Par soucis de simplification, il ne sera pas fait explicitement état de ces numéros de variable dans le code, mais plutôt de leurs alias, autrement plus compréhensibles.

La table de conversion de ces numéros et alias est située dans le fichier mpfields.h, dans le répertoire include de Xtender.

Lors de l'utilisation de ces variables en programmation, une attention particulière au type utilisé sera requise, selon que l'on utilise un pointeur ou une variable.

En effet, en C, une distinction est faite entre le contenu d'une variable (l'information) et son adresse en mémoire (le pointeur).

La notion de pointeur reflète l'utilisation différente que l'on peut faire d'un nombre entier, à savoir indiquer une adresse mémoire. Cette utilisation est très différente d'une utilisation arithmétique, d'où la création de registres processeurs spécifiques (les registres d'adresse) et d'un type de donnée spécifique dans les langages de programmation.

L'utilisation des pointeurs permet d'accéder à l'adresse mémoire de la variable, dans le cadre d'une utilisation spécifique du code (saut de variables, optimisation de code, attaques de type buffer overflow auxquelles le C est très sensible…). Cependant, cette utilisation dépasse le cadre du drone.

En C, la déclaration d'un pointeur s'effectue en utilisant le préfixe * dans la déclaration d'une variable. L'attribution d'une valeur à cette variable en utilisant le pointeur s'effectue en déréférençant le pointeur, toujours avec le préfixe *.

Cette particularité peut être source d'erreur dans le code, notamment pendant l'utilisation des variables publiques du Micropilot (celles précitées). Pour lever toute ambiguïté, une variable système sera donc utilisée de cette façon :

```
Static long* tpitch ;                    //déclaration

getMPVarPointer( &tpitch, MPFID_ELEVATOR_PULSE)  ;     //affectation à la variable système 1231 (alias : MPFID_ELEVATOR_PULSE)

resultat = MPVAR(tpitch) + 5 ;           //utilisation
```

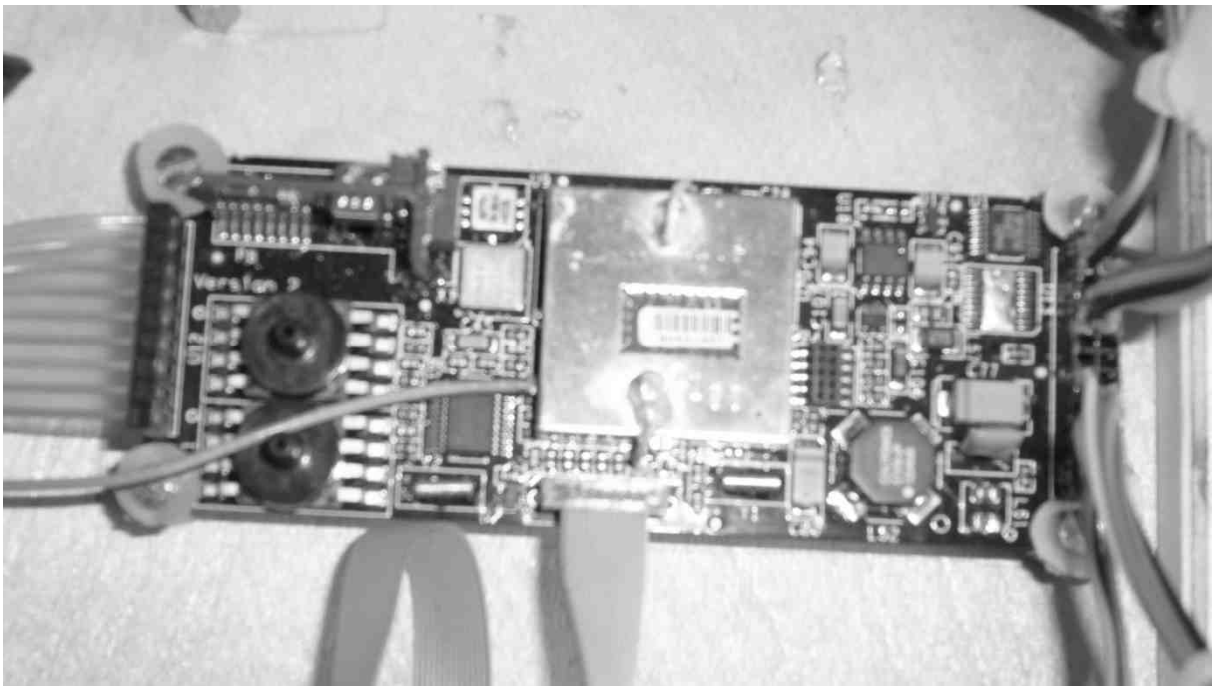
En pratique, toute référence à une variable système se fera par la syntaxe MPVAR(nom_de_variable).

Dernière chose, pour faciliter la lisibilité du code, une lettre sera placée en préfixe des variables pour désigner ce dont il s'agit :

- Un m désigne une donnée mesurée : renvoyée telle quelle par les capteurs de la carte.
- Un t désigne une donnée de télécommande : reçue telle quelle par l'entrée télécommande
- Un c désigne une donnée de commande : variable de type t après mise en forme
- Un p désigne une grandeur dérivée une fois
- Un pp désigne une grandeur dérivée deux fois

Par exemple dans le code ci-dessus, tpitch représente la donnée de la commande de tangage (en anglais pitch) envoyée par la télécommande et donnée par les capteurs de la carte.

Enfin, il faut noter que la carte n'exécute pas immédiatement au démarrage le code utilisateur. Elle commence par une séquence d'initialisation des accéléromètres, du GPS, et de contrôles divers (tension, ampérage, branchements…). Cette phase dure environ 2 minutes et est détaillée en annexe.



Carte Micropilot

ACQUISITION DES DONNÉES

Avant de se lancer dans la programmation de la carte, il convient de réunir les données suffisantes. Il faut donc comprendre comment la carte gère ses différentes entrées sorties. Pour contrôler le drone, nous avons besoin de :

- L'attitude du drone, donnée fournie par les accéléromètres
- La commande utilisateur, donnée envoyée par la télécommande et fournie par le récepteur
- La commande envoyée aux moteurs, données envoyées aux sorties servos.

Le drone est contrôlé par la variation de vitesse des moteurs, les variateurs des quatre moteurs brushless sont donc reliés à la carte servoboard.

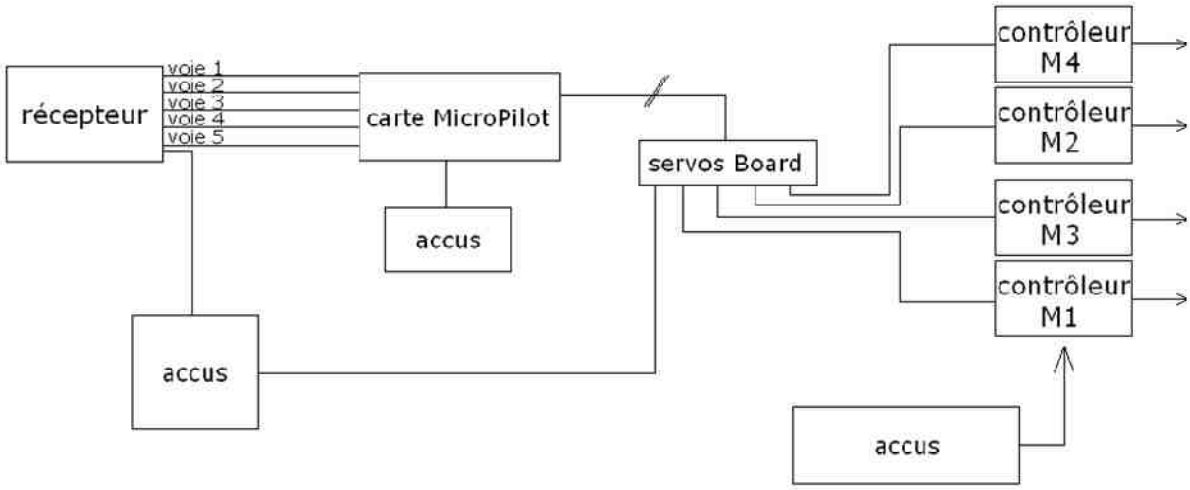


Schéma de principe des connexions

Nous disposons donc de quatre entrées (roulis, tangage, lacet et poussée), et de quatre sorties (une par moteur, ainsi que des données d'attitude fournies par la carte pour chaque axe, ainsi que leur dérivée.

Le but de cette section est d'analyser le comportement de ces entrées/sorties.

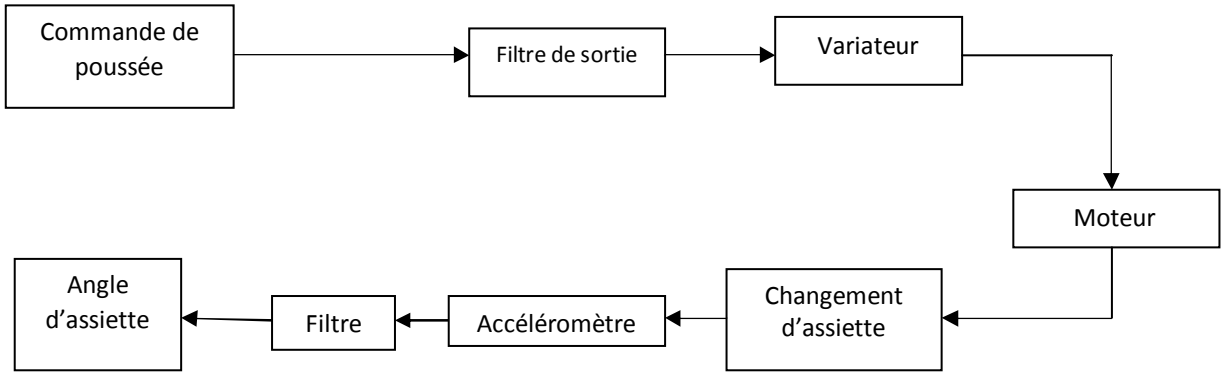
MESURE DU TEMPS DE RÉPONSE DES VARIATEURS

Le code qui sera programmé dans la carte Micropilot sera exécuté à une fréquence de 30 Hz. Pour un bon contrôle d'attitude, il est primordial que les variateurs qui commandent les moteurs aient un temps de réaction suffisamment court pour assurer le contrôle.

Nous allons donc mesurer le temps de réaction du système à une variation brusque d'attitude. C'est-à-dire le temps qui s'écoule entre le moment où un ordre de changement d'attitude est envoyé au moteur et le moment ou ce changement d'attitude est mesuré par la carte. Dans le code, nous programmons la carte pour

mettre à fond un moteur lorsque nous faisons varier la poussée sur la télécommande. L'hélice de ce moteur générera donc une portance qui fera brusquement varier l'assiette du drone. Ce changement d'assiette sera ensuite enregistré par les accéléromètres.

Nous mesurons ainsi le temps de réponse de la chaine de décision-acquisition :

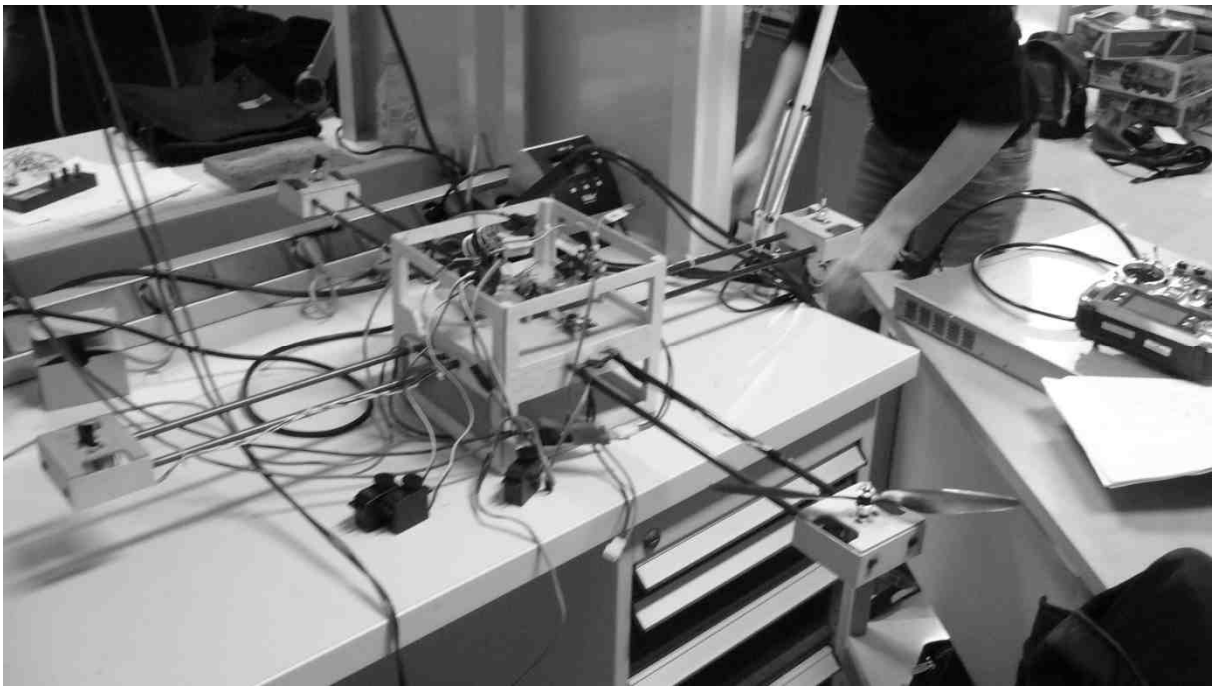


Le temps de réponse est mesuré grâce aux fonctions de télémétrie de la carte. Au cours de son fonctionnement, celle-ci enregistre périodiquement les données d'attitude et d'état des servocommandes. Nous mesurerons donc le temps entre la variation de la servocommande (commande de poussée) et la variation de la mesure de roulis (angle d'assiette).

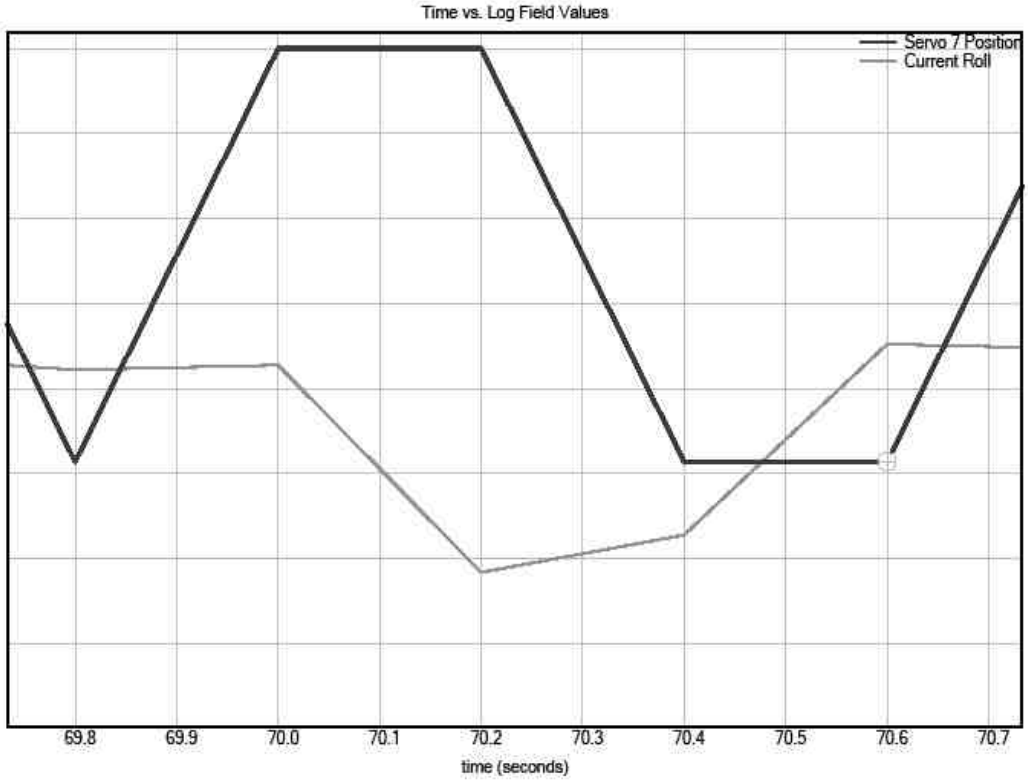
Le code de commande est le suivant :

```
thrust = (MPVAR(tthrust)-3000)*30;        //mise en forme de la commande de poussée

if(thrust>10000){                         //si la poussée est grande (commande à mi-course au moins)
        MPVAR(fServo7)=23000;             //moteur 4 presque à fond (relié au servo 7)
}
else{                                     //si la poussée est faible (commande à mi-course au plus)
        MPVAR(fServo7)=-32767;            //moteur 4 coupé
}
```

Nous chargeons alors ce code dans la carte et exécutons une série de va et vient sur la commande de poussée de la télécommande. Le drone effectue donc des impulsions sur le moteur 4 qui fait osciller l'appareil sur l'axe de roulis.



Après essai, les données de datalog sont récupérées sur l'ordinateur et lues avec le logiciel LogViewer. Nous traçons l'évolution de la commande (Servo 7) et de la mesure d'angle (Current Roll) :



Nous constatons sur ce graphe que toute modification de la commande se traduit en apparence par une variation en roulis exactement 0.2 secondes plus tard. En réalité ces données sont inexactes. En effet, le système de mesure par datalog enregistre les paramètres avec une fréquence de 5 Hz, soit toutes les 0.2 secondes. Le temps de réponse est donc ici probablement inférieur mais nous ne pouvons pas le mesurer plus précisément par ce moyen.

Décortiquons le graphe : la commande fonctionne en tout ou rien, si Servo 7 est en bas, le moteur doit être coupé, s'il est en haut, le moteur reçoit une commande de mise en route à environ les 2/3 de sa poussée maximale. Ce qui se traduit par une variation négative du roulis. Sur cet essai, nous lançons le moteur à fond puis l'y maintenons quelques temps, avant de le couper quelques temps. Le roulis varie 0.2 secondes après la commande, puis se stabilise et remonte légèrement 0.2 secondes après la commande de moteur maintenu en mouvement (ceci s'explique facilement : lorsque le moteur partait à fond, le drone se soulevait du côté du moteur 4 et allait cogner sur une butée : nous n'allions pas le laisser partir dans tous les sens tout seul ! cette stabilisation vient donc de l'arrivée en butée de l'engin). Le roulis remonte ensuite puis regagne l'horizontale 0.2 secondes après l'arrêt de la commande.

Nous pouvons donc affirmer que la chaîne d'information a un temps de réponse inférieur à 0.2 secondes. Ce temps semble convenable. Il aurait été trop lent s'il avait dépassé 0.3 secondes.

Nous pouvons donc continuer d'utiliser Micropilot, le système semble suffisamment rapide pour assurer une stabilisation décente.

MISE EN FORME DES DONNÉES

Pour garder un code efficace et compréhensible, il faut mettre en forme les données d'entrée sortie afin de toujours travailler avec des ordres de grandeur équivalents. Cette mise en forme n'est pas effectuée par la carte, qui attribue des coefficients différents à toutes les données.

Pour des raisons de lisibilité, nous avons décidé de modifier toutes les entrées de manière à utiliser un ordre de grandeur similaire à la sortie.

MISE EN FORME DU SIGNAL DE COMMANDE DES MOTEURS

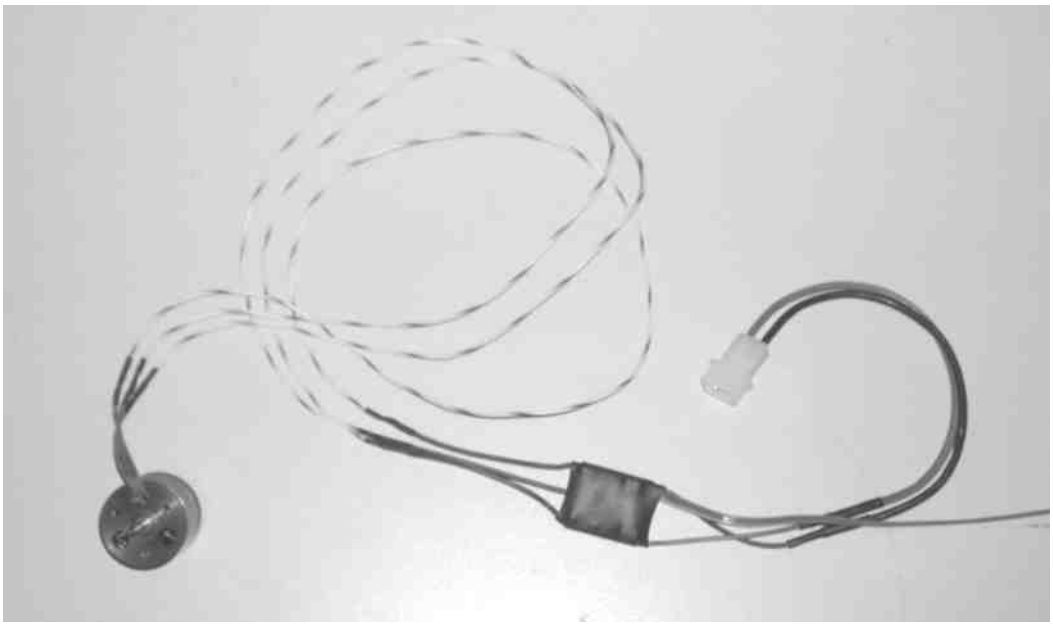
Comme indiqué précédemment, nous allons aligner les entrées sur les ordres de grandeur des sorties. Il n'y aura donc pas de mise en forme des données envoyées aux moteurs.

Les moteurs sont commandés par les variateurs, qui assurent le découpage et l'amplification du signal correspondant au modèle de moteur utilisé. Ce sont donc les variateurs qui sont commandés par la carte.

Le signal de commande d'un variateur est standardisé pour le modélisme. Il s'agit d'un signal de type PWM (Pulse Width Modulation) dont l'amplitude est limitée à 6V maximum.

Il n'est pas nécessaire de programmer la période et le rapport cyclique, Micropilot effectuant ces réglages de manière autonome. Pour modifier le PWM, il suffit d'attribuer aux variables fServo une valeur comprise entre -32767 et 32767. La valeur -32767 correspond au réglage minimum, et donc au réglage moteur éteint. Il est

important de noter que les variateurs utilisés requièrent une commande à zéro lors de l'initialisation des moteurs. Le code devra donc toujours envoyer -32767 aux variateurs au démarrage. Pour cette même raison, les variateurs ne seront connectés au servoboard qu'une fois la carte initialisée. En effet, il n'y a aucun moyen de contrôler les valeurs envoyées au servoboard lors de l'initialisation.



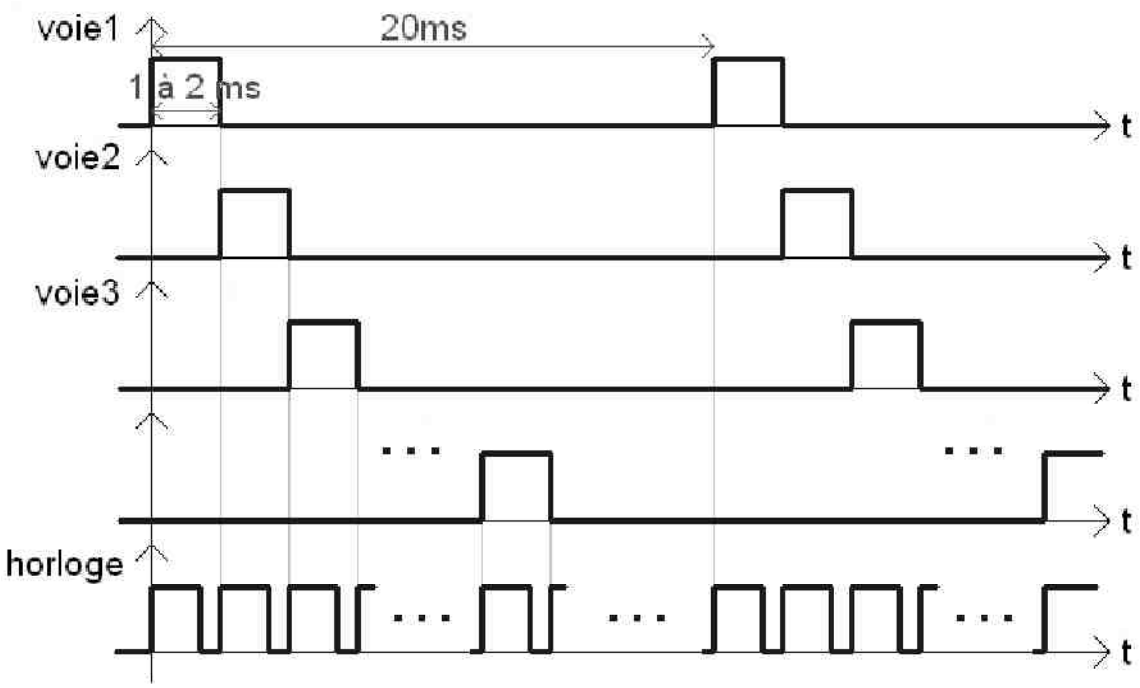
Moteur et variateur

Les autres données seront donc mises en forme de manière à être comprises entre -32767 et 32767. De cette façon, il sera facile d'additionner des données de natures différentes.

MISE EN FORME DU SIGNAL DE LA TÉLÉCOMMANDE

La télécommande envoie un signal modulé numériquement au récepteur. Ce signal contient l'état de toutes les voies de la télécommande et est échantillonné à une période T. Le récepteur reçoit donc tous les T l'état de toutes les voies. Cette période T est découpée en autant de morceaux n que de voies (ici n=9, mais nous ne nous servons que de quatre voies). Chaque voie est encodée sous la forme d'un PWM de période t de sorte que n x t = T. Le rapport cyclique donne la valeur de la voie.

Le récepteur découpe ensuite le signal transmis et restitue sur chacune de ses sorties le signal correspondant à une voie.



Forme du signal restitué par le récepteur

De cette manière, la valeur de chacune des voies est proportionnelle au TON (durée pendant laquelle le signal est au niveau haut) du signal correspondant.

C'est ce TON que Micropilot restitue à l'utilisateur lorsqu'il lit les valeurs tpitch, troll, tyaw et tthrust.

Afin de mettre en forme le signal, il nous faut donc connaître les bornes de ce signal pour chaque voie.

Nous relions donc la sortie d'une voie du récepteur à un oscilloscope sur lequel nous pouvons mesurer le TON d'un signal de type PWM, puis nous faisons varier la commande correspondante de la butée haute à la butée basse de la télécommande. L'opération est renouvelée pour chaque voie.

Voie	Thrust	Roll	Pitch	Yaw
TON mesuré	De 1ms à 1,92 ms	De 1,31ms à 1,72ms	De 1,30ms à 1,72ms	De 1,1ms à 1,94ms

Il apparaît que les données de lacet et poussée varient entre 1 et 2ms et les données de roulis et tangage entre 1,2 et 1,8ms.

Dans le code, Micropilot renvoie la valeur des TON en millisecondes multipliée par 2000. Afin d'obtenir une valeur comprise entre -32767 et 32767, il faut donc multiplier les données reçues par 32767 et diviser par l'écart de TON.

Ainsi tthrust et tyaw sont multipliés par 32767/1000. Troll et tpitch sont multipliés par 32767/600.

Enfin ces données sont centrées, de sorte qu'une position relâchée des commandes de roulis, tangage et lacet renvoie 0 et qu'une position gaz en bas renvoie -32767.

Pour cela, l'utilisateur doit placer sa télécommande au démarrage gaz en bas et relâcher les autres axes. Une mesure est alors faite de la position des voies puis est utilisée pour centrer les axes.

MISE EN FORME DES SIGNAUX D'ATTITUDE

La carte Micropilot permet de mesurer les positions angulaires en roulis, tangage et lacet ainsi que leurs dérivées (vitesses angulaire). Le drone est asservi en position angulaire et la correction utilisée est de type PDD². Il ne manque donc que l'accélération angulaire pour effectuer la correction. Cet état sera reconstitué par dérivation numérique à partir des vitesses angulaires.

Deux types de mesures sont disponibles pour la position : la instantaneous value et la current value. La current value est obtenue en faisant la moyenne sur 1/20 s des données mesurées. Cette valeur subit de plus un filtrage propre à la carte. La instantaneous value se contente de donner en permanence la position angulaire mesurée. Elle est donc sujette à un bruit de mesure important. Puisque le code n'est exécuté que 30 fois par seconde, et que de toutes manières, c'est amplement suffisant, nous utilisons la current value, de manière à avoir le signal le plus propre possible.

Les angles de type current sont exprimés par Micropilot en radians, et sont multipliés par 1024. Le quadrirotor est supposé ne pas dépasser des angles de plus de 30 degrés en tangage et roulis. La carte renvoie donc des angles compris entre -535,9 et 535,9 (car $1024*30*\pi/180 = 535,9$). Pour faire simple, nous arrondirons à 540.

Les angles de roulis et tangage sont donc multipliés par 32767/540.

Le lacet a un comportement particulier. En effet, cet axe ne possède pas de référence fixe (nous ne l'alignons pas sur un angle magnétique). Le comportement choisi est celui utilisé par la carte : la référence en lacet (angle 0°) est définie par l'axe longitudinal (axe de roulis). Lorsque le drone avance en ligne droite, on considère que le lacet est de 0°. Si le drone pivote, son lacet change. S'il se stabilise à un nouvel angle, le lacet redevient 0°, la référence étant cette nouvelle position. En pratique la mesure de lacet renvoie toujours 0 lorsque le drone ne possède pas de mouvement de lacet. Lorsqu'il pivote, Micropilot renvoie la variation d'angle par rapport à la position initiale puis renvoie à nouveau 0 lorsque le drone est stabilisé sur l'axe de lacet. Il s'agit en quelque sorte d'une intégration de la vitesse angulaire avec remise à zéro périodique.

Nous considérons que le drone n'excédera pas de variations sur l'axe de lacet de plus de 45° d'un coup. Micropilot renvoie donc au maximum 810.

L'angle de lacet est multiplié par 32767/810.

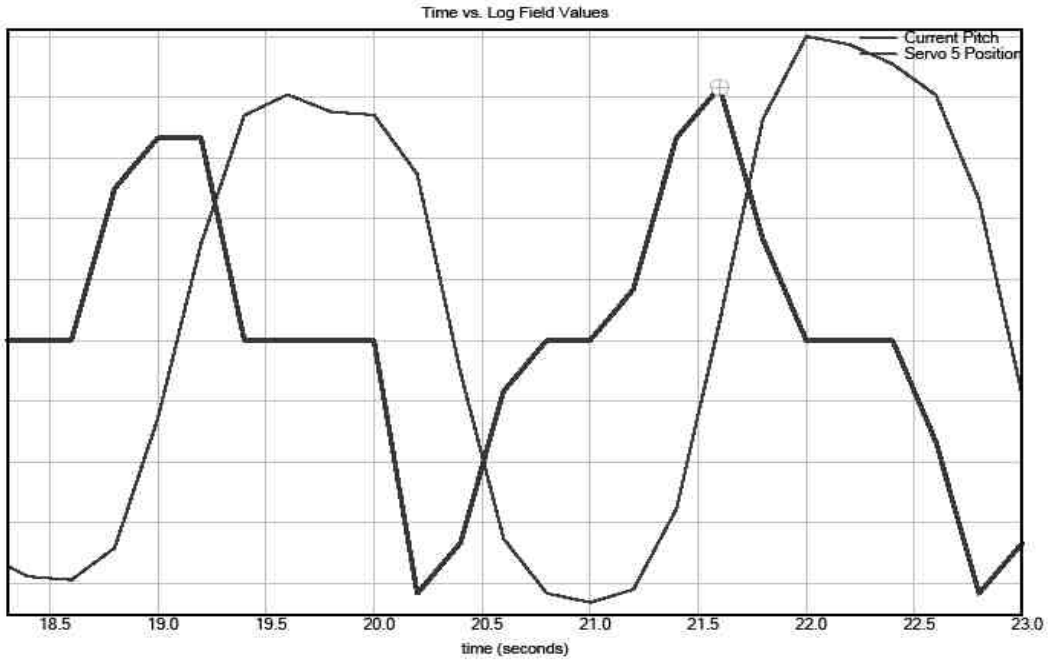
L'écart en vitesse et accélération ne peut être obtenu par un raisonnement aussi simple, car nous n'avons a priori aucune idée des vitesses et accélérations angulaires maximales. Il faut donc les mesurer. Nous utilisons un programme de test qui enregistre les données de vitesse et accélération dans le datalog. Il suffira de faire manuellement pivoter le drone pour mesurer ces grandeurs. Les données enregistrées sont donc composées des valeurs de vitesse et accélération angulaires de tangage et de lacet. Le roulis n'est pas mesuré, car il a le même comportement que le lacet. L'accélération est reconstituée à partir de la vitesse. Comme le datalog

n’enregistre que certaines valeurs système et qu’il n’est pas éditable, nous écrivons simplement les données qui nous intéressent dans les fServo (qui ne sont reliés à rien pour ce test).

Code utilisé :

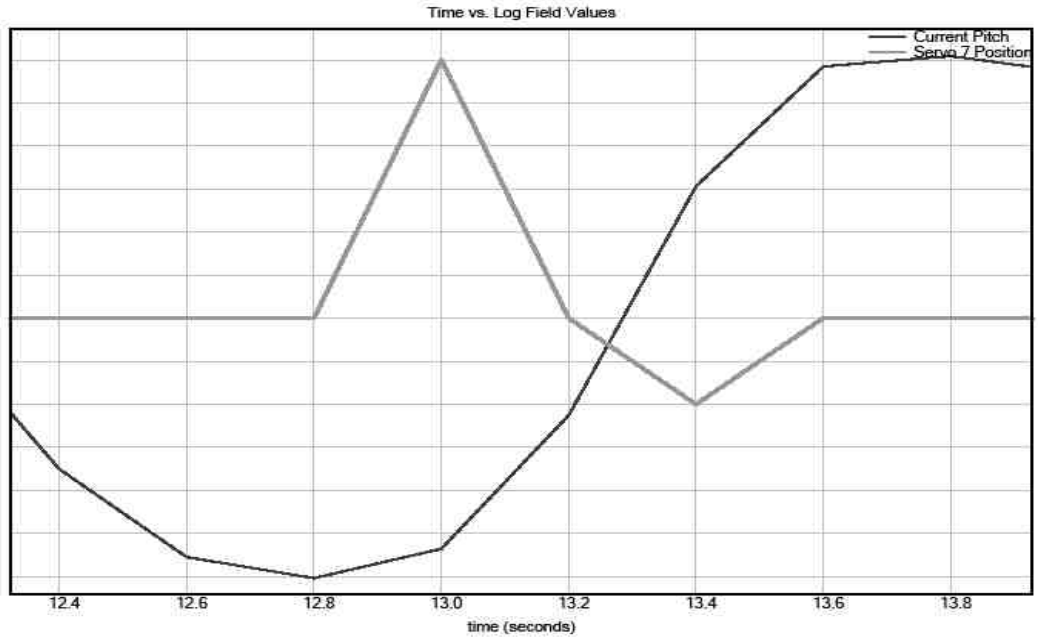
```
//calcul de l'accélération   le 0.24 est une échelle de temps

p1pitch=p2pitch;

p2pitch=MPVAR(ppitch);

pppitch=(p2pitch - p1pitch)*100/24;

//calcul de l’accélération en lacet

p1yaw=p2yaw;

p2yaw=MPVAR(pyaw);

ppyaw=(p2yaw-p1yaw)*100/24;

//ecriture des resultants sur des variables enregistrées dans le datalog

MPVAR(fServo5)=MPVAR(ppitch);

MPVAR(fServo6)=MPVAR(pyaw);

MPVAR(fServo7)=pppitch;

MPVAR(fServo8)=ppyaw;
```

Nous faisons exécuter au drone une série de mouvements caractéristiques de la dynamique souhaitée. Nous lisons ensuite les valeurs extrêmes de variation des vitesses et accélérations :



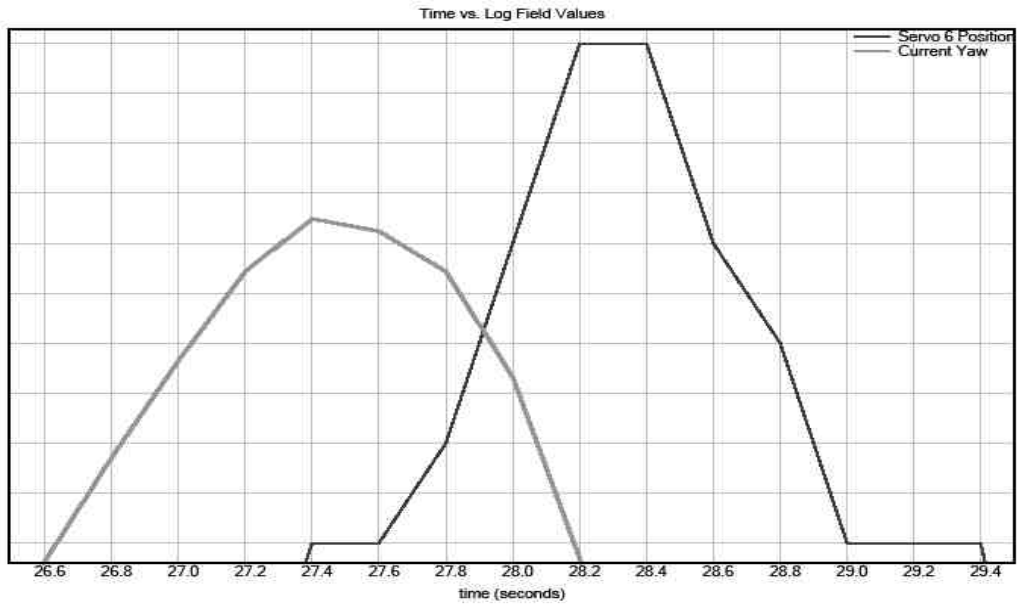
Mesure sur la vitesse de tangage

Pour la vitesse de tangage, nous mesurons une valeur maximale de 1536.



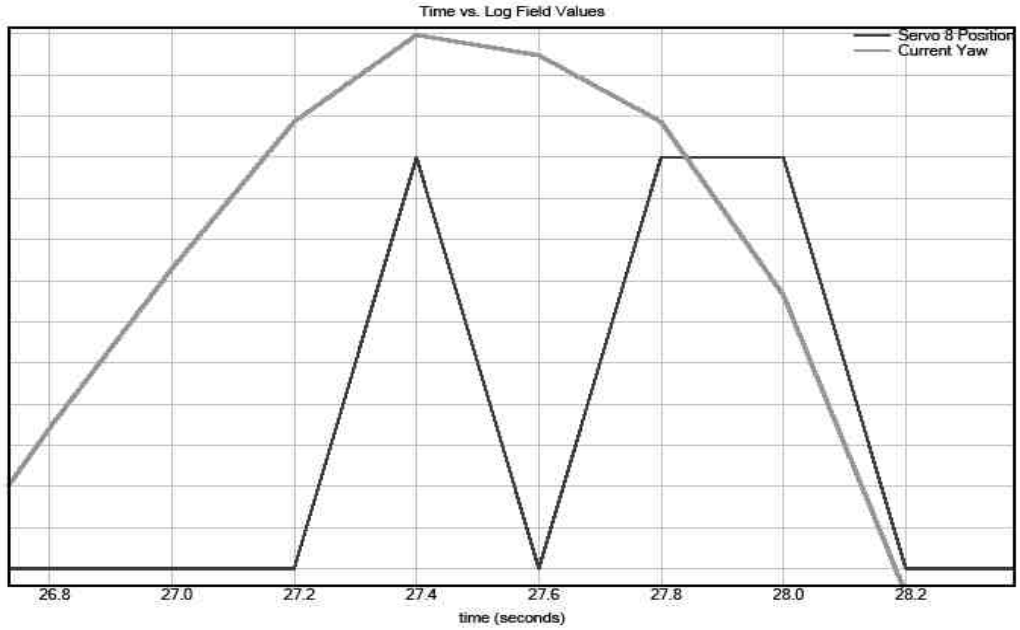
Mesure sur l'accélération en tangage

Pour l'accélération en tangage, la valeur maximale mesurée est 1280.



Mesure sur la vitesse en lacet

Pour la vitesse en lacet, la valeur maximale mesurée est de 1280.



Mesure sur l'accélération en lacet

Pour l'accélération en lacet, la valeur maximale mesurée est de 768.

Compte tenu du côté approximatif de ces mesures, et afin de se garder une marge vis-à-vis de la dynamique réelle, nous arrondissons ces valeurs au millier supérieur. Même si les valeurs d'attitude n'atteindront sans doute jamais 32767, il ne faut pas oublier qu'il ne s'agit que d'ordres de grandeur, et que les vraies valeurs seront modifiées par les coefficients du PDD².

Dans le code, les valeurs mesurées pour la vitesse angulaire en tangage, lacet et roulis seront multipliées par 32767/2000.

Les valeurs mesurées pour l'accélération en tangage et roulis seront multipliées par 32767/2000.

Les valeurs mesurées pour l'accélération en lacet seront multipliées par 32767/1000.

Nous devons maintenant connaître le sens de variation de ces angles et vitesses par rapport au sens conventionnel de variation du tangage, lacet et roulis.

Nous reprenons donc notre code et faisons effectuer au drone une séquence de mouvements précis :

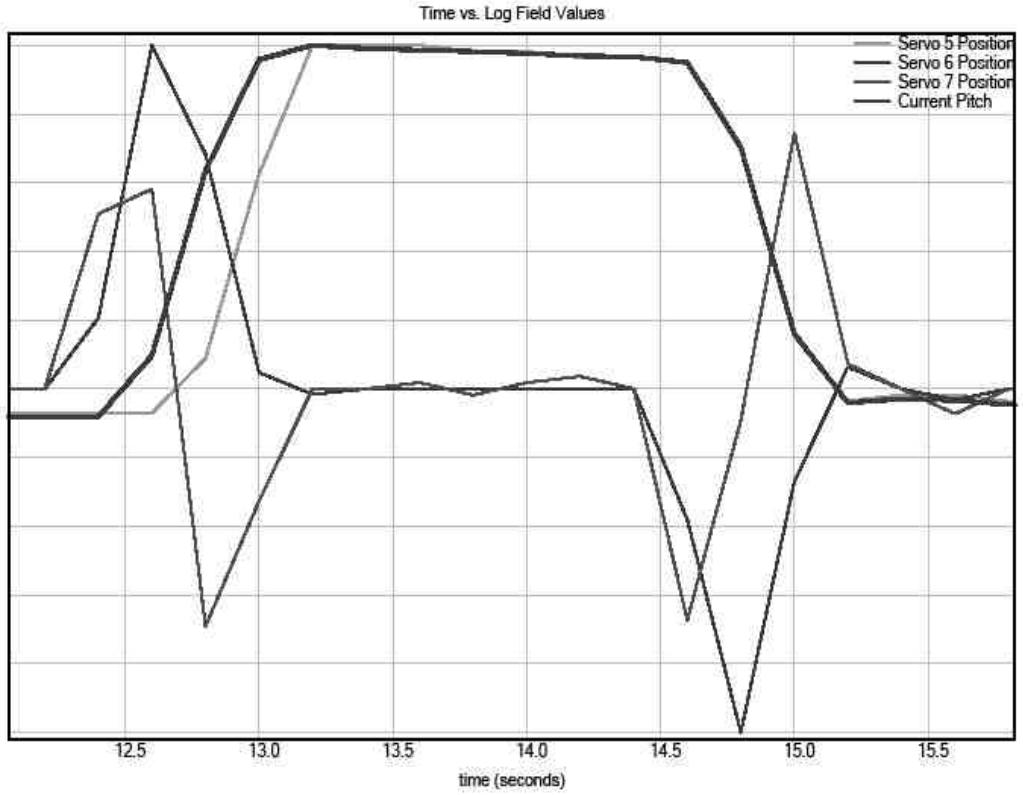
1. Un coup à piquer
2. Un coup à cabrer
3. Un coup à bâbord
4. Un coup à tribord

Code utilisé:

```
//calcul de l'accélération  le 0.24 est une echelle de temps
```

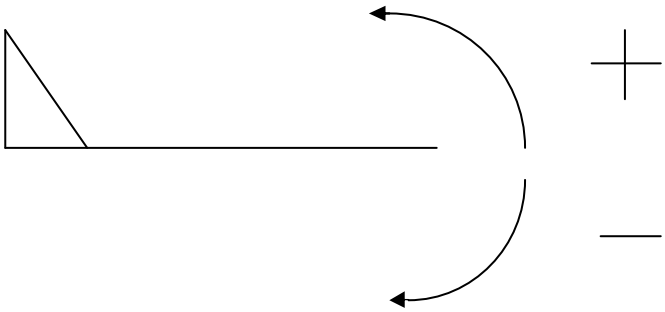
```
p1pitch=p2pitch;
p2pitch=MPVAR(ppitch);
pppitch=(p2pitch - p1pitch)*100/24;

MPVAR(fServo5)=MPVAR(mpitch)*32767/810;
MPVAR(fServo6)=MPVAR(ppitch)*32767/2000;
MPVAR(fServo7)=pppitch*32767/2000;
MPVAR(fServo8)=MPVAR(mroll)*32767/810;
```

Observons l'allure du tangage, de la vitesse angulaire en tangage et de l'accélération en tangage pour le mouvement « à cabrer » :

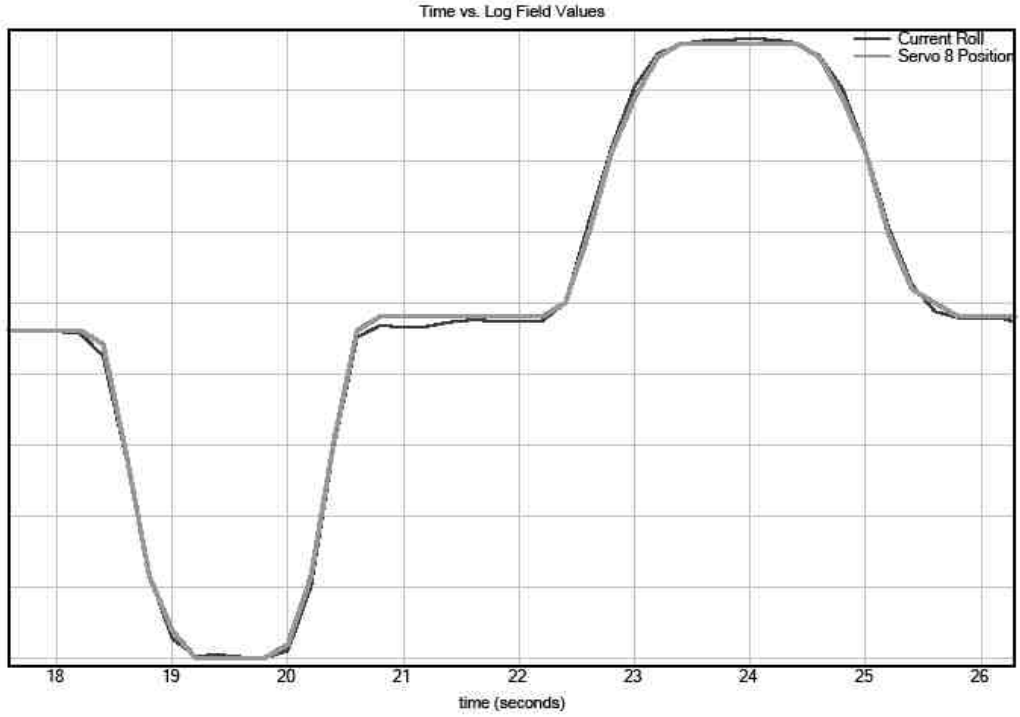


Mouvement à cabrer

Le signe de la position angulaire suit bien le sens conventionnel d'un mouvement à cabrer, c'est-à-dire qu'il est positif. La vitesse suit elle aussi le sens conventionnel, lors de l'initiation du mouvement cabreur, elle est positive puis redescend à zéro au palier. L'accélération suit cette tendance, pour la même phase elle est successivement positive (accélération à cabrer) puis négative (décélération avant le palier).

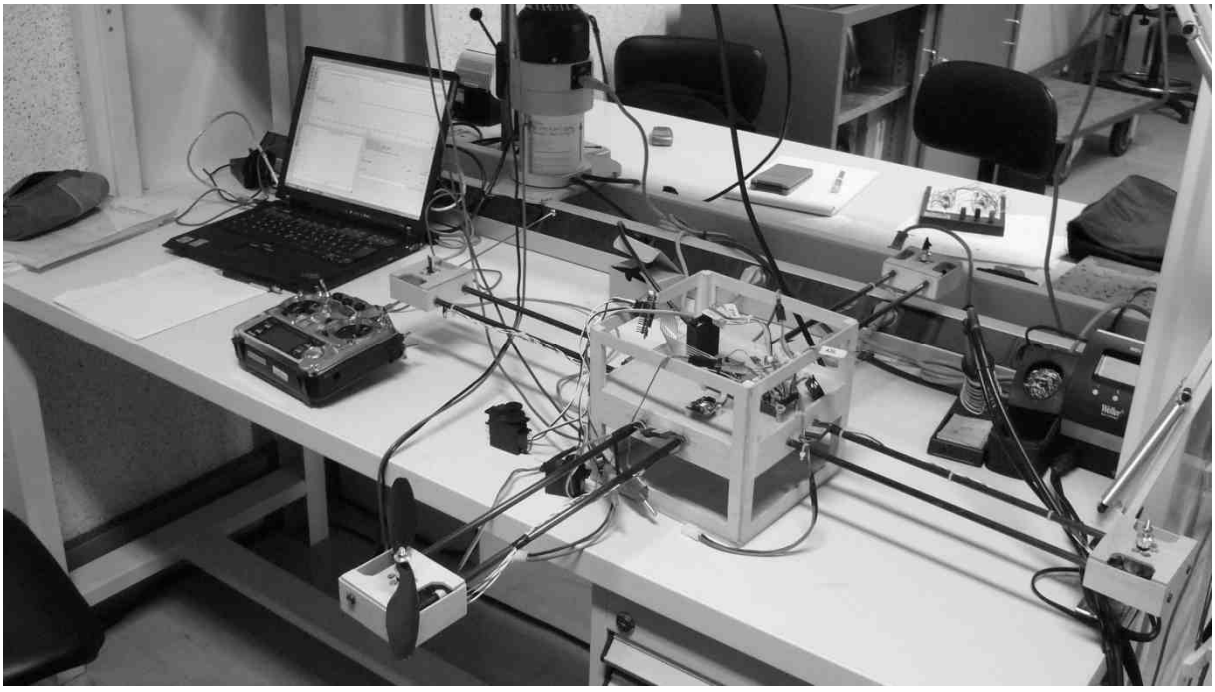


En ce qui concerne le mouvement en roulis :



Mouvement en roulis à bâbord puis à tribord

Ici aussi Micropilot suit les conventions de signes habituelles, une rotation à tribord étant notée positive.



CODE DE COMMANDE

Le fonctionnement de la carte Micropilot est désormais acquis, il ne reste plus qu'à écrire le code de stabilisation.

INITIALISATION

Le programme doit commencer son exécution par l'acquisition du zéro de position des commandes. Cette manœuvre est effectuée grâce à une variable booléenne qui permettra de n'effectuer ce centrage qu'à la première exécution du code. La variable en question est ici nommée « init ».

```
//initialisation de la carte, on récupère les valeurs zéro de la télécommande
//ce code n'est exécuté qu'une fois au démarrage de la carte
if(init==1){

        zerothrust=MPVAR(tthrust);
        zeroyaw=MPVAR(tyaw);
        zeroroll=MPVAR(troll);
        zeropitch=MPVAR(tpitch);
        c2pitch=zeropitch;
        c2roll=zeroroll;
        c2yaw=zeroyaw;
        init=0;
}
```

Les variables de type « c2machin » sont utilisées dans le rate limiter, dont le fonctionnement est expliqué quelques pages plus loin. Lors de l'initialisation du code, l'utilisateur doit lâcher les commandes et s'assurer que la manette des gaz est bien au minimum.

Une fois cette vérification effectuée, il faut forcer une variable de la carte. Précisément, il faut spécifier à Micropilot que le drone n'a pas décollé. En effet, le fait que le drone ait ou non décollé n'est pas utilisé dans notre code, mais cette simple variable perturbe l'interface de contrôle Horizon. Car le code de base de Micropilot verrouille l'accès aux variables systèmes lorsque l'appareil est en vol, et l'on ne peut alors plus rien modifier (comme les réglages du PDD²). Cela est d'autant plus gênant que Micropilot détecte très mal si le système est ou non en vol, d'où des erreurs intempestives.

```
MPVAR(offground)=0;
```

Cette petite modification évitera par la suite bien des problèmes dus à cette variable.

MISE EN FORME DES SIGNAUX DE LA TELECOMMANDE

La prochaine étape consiste à mettre en forme les signaux de la télécommande. Ces signaux sont centrés et mis à l'échelle par rapport aux ordres de grandeur des servocommandes. Pour les axes de roulis, lacet et tangage, le point nul est au milieu de l'axe. Pour la commande des gaz, le réglage à zéro correspond à la position basse de la manette.

```
//mise en forme des signaux envoyés par la télécommande en signaux de grandeurs comparables aux fServo (variant entre -32767 et +32767)

croll = (MPVAR(troll) - zeroroll)*32767/600;

cpitch = (MPVAR(tpitch) - zeropitch)*32767/600;

cyaw = (MPVAR(tyaw) - zeroyaw)*32767/1000;

cthrust = -32767+(MPVAR(tthrust)-zerothrust)*32767/1000;//la position 0 pour les gaz correspond à la manette en bas, donc à -32767
```

Pour rappel, le préfixe t signifie télécommande et le préfixe c signifie commande. La différence provenant de la mise en forme des signaux en t sous la forme c.

CALCUL DES ACCÉLÉRATIONS

La carte Micropilot ne permet pas de lire les états d'accélération, il faut donc les reconstituer à partir des données de vitesse, par dérivation numérique. Cette dérivation s'effectue par soustraction de deux vitesses et division par le pas de temps entre ces deux vitesses. D'après l'étude d'Andreas Görmer sur la dynamique de vol d'un drone, le pas de temps entre la mesure de deux vitesses est de 0.24 secondes. Cette valeur est simplement donnée à titre d'indication, car l'accélération ainsi calculée sera de toute manière multipliée par un coefficient de correction.

```
//calcul de l'accélération   le 0.24 est une echelle de temps

p1pitch=p2pitch;

p2pitch=MPVAR(mppitch);

pppitch=(p2pitch - p1pitch)*100/24;


p1roll=p2roll;

p2roll=MPVAR(mproll);

pproll=(p2roll-p1roll)*100/24;


p1yaw=p2yaw;

p2yaw=MPVAR(mpyaw);
```

```
ppyaw=(p2yaw-p1yaw)*100/24;
```

Les variables en pimachin sont des variables de stockage temporaire des vitesses.

MISE EN PLACE D'UN RATE LIMITER

Une stabilisation efficace demande de faibles variations sur la commande. En effet, de fortes variations provoqueraient le calcul d'une correction inappropriée, voire divergente. Il faut donc limiter la vitesse de variation de la commande. Cela permettra aussi de rejeter des perturbations rapides dues au bruit ou à des mouvements saccadés et rapides sur les manettes de la télécommande. Ce problème est résolu par un rate limiter.

Il s'agit d'une portion de code qui analyse l'entrée et la compare à un seuil. Le rate limiter filtre ensuite l'entrée de manière à ne pas dépasser le seuil. Ainsi toute variation brusque sur la commande est évitée. Une stabilisation efficace est alors possible.

Tout le problème du rate limiter réside dans le calcul du seuil. En pratique, le seuil a simplement été défini de manière empirique sur le banc de test. Une valeur arbitraire a d'abord été testée, puis affinée de manière à obtenir un bon compromis entre la stabilité du drone et la réactivité de la commande.

```
//calcul du rate limiter

c1pitch = c2pitch;

c1roll = c2roll;

c1yaw = c2yaw;


c2pitch = cpitch;

c2roll = croll;

c2yaw = cyaw;


dpitch=(c2pitch-c1pitch);

droll=(c2roll-c1roll);

dyaw=(c2yaw-c1yaw);


if(dpitch>400){dpitch=400;}//valeur 400 retenue apres essai

if(dpitch<-400){dpitch=-400;}

if(droll>400){droll=400;}

if(droll<-400){droll=-400;}
```

```
if(dyaw>400){dyaw=400;}
if(dyaw<-400){dyaw=-400;}


cpitch = c1pitch + dpitch;
croll = c1roll + droll;
cyaw = c1yaw + dyaw;
```

La commande retenue pour le calcul de la correction est donc obtenue en sommant la commande précédente et sa variation. Cette variation est filtrée pour ne pas excéder le seuil.

CALCUL DE L'ERREUR DE POSITION

L'erreur utilisée pour la commande est obtenue par soustraction de l'attitude mise en forme à la commande.

```
//calcul de l'erreur de position
//on supposera que le drone ne peut pas excéder des angles de 30 degrés et 45 degrés en lacet
pitch = cpitch-MPVAR(mpitch)*32767/540; // 540 = 30° en radians * 1024
roll = croll-MPVAR(mroll)*32767/540;
yaw = cyaw-MPVAR(myaw)*32767/810; //810=45° en rad *1024
```

Cette formule représente bien l'intérêt de mettre en forme les signaux: on soustrait deux signaux de même ordre de grandeur.

CALCUL DE L'EFFET INTEGRAL (OPTIONNEL)

Pour éliminer l'erreur statique, il est possible d'utiliser un effet intégral. Le code contient donc un segment d'intégration de la position. Pour éviter que cette intégration ne diverge, un filtrage est ensuite réalisé afin de ne pas produire une valeur qui aurait des effets trop importants sur la commande. Idéalement, l'effet intégral ne doit intervenir sur la commande que lorsque le drone est en position fixe. En effet, l'effet intégral est déstabilisateur, son seul rôle étant d'éliminer l'erreur. De plus amples commentaires sur l'effet intégral sont fournis dans la partie stabilisation du drone.

```
//calcul de l'effet integral
ipitch = ipitch + pitch;
iroll = iroll + roll;
iyaw = iyaw + yaw;
if(ipitch>10000){ipitch=10000;}
```

```
if(ipitch<-10000){ipitch=-10000;}

if(iroll>10000){iroll=10000;}

if(iroll<-10000){iroll=-10000;}

if(iyaw>10000){iyaw=10000;}

if(iyaw<-10000){iyaw=-10000;}
```

MISE À L'ÉCHELLE DES VITESSES ET ACCÉLÉRATIONS

Conformément aux réglages obtenus dans la partie acquisition des données, les données de vitesse et accélération sont mises à l'échelle.

```
//mise à l'échelle des vitesses et accelerations(sur une échelle de -32767 à 32767)

ppitch = MPVAR(mppitch)*32767/2000;

proll = MPVAR(mproll)*32767/2000;

pyaw = MPVAR(mpyaw)*32767/2000;


pppitch = pppitch*32767/2000;

pproll = pproll*32767/2000;

ppyaw = ppyaw*32767/1000;
```

CALCUL DU CORRECTEUR

La correction proposée est de type PIDD². Dans la section stabilisation, nous montrerons que l'effet intégral n'a finalement pas été utilisé. Il est livré ici pour information.

La commande est constituée de l'erreur à laquelle sont soustraites les données de vitesse et d'accélération. Ces données dérivées ont un effet stabilisateur. En effet, on comprend que si le drone est écarté de la position voulue, pour obéir à la commande, il doit initier un mouvement de retour. La commande doit donc être importante mais décroître à mesure que l'on s'approche de la position demandée. Comme vitesse et accélération augmentent dans le cas d'un asservissement purement proportionnel, le fait de les soustraire permet de réaliser une approche de la position demandée en douceur, et ainsi éviter le pompage. Vitesse et accélération sont donc soustraites à l'erreur. De même, l'effet intégral doit être ajouté à l'erreur. Ainsi, si le drone se stabilise à une position légèrement écartée de la position demandée (cet écart apparaît naturellement du fait des différences de vitesse de rotation des moteurs pour une même commande), le système intègre cet erreur ce qui crée une commande de correction croissante. A partir d'un certain temps, l'intégration est suffisamment grande pour provoquer le déplacement effectif du drone, et l'annulation de l'erreur statique. On remarque que l'intégration a bien un effet déstabilisant, puisqu'il fait diverger la commande.

Les axes de roulis et tangage sont corrigés de la même manière, car ils sont symétriques et ont donc a priori la même dynamique. Ce choix a été confirmé par la pratique. L'axe de lacet possède lui d'autres coefficients.

```
//calcul du correcteur

//le /100 permet d'utiliser des coeffs 100 fois plus grands (plus lisible)

pitchmix = (MPVAR(K1roll)*pitch - MPVAR(K2roll)*ppitch - MPVAR(K3roll)*pppitch + MPVAR(K4roll)*ipitch)/100;


rollmix = (MPVAR(K1roll)*roll - MPVAR(K2roll)*proll - MPVAR(K3roll)*pproll + MPVAR(K4roll)*iroll)/100;


yawmix = (MPVAR(K1yaw)*yaw - MPVAR(K2yaw)*pyaw - MPVAR(K3yaw)*ppyaw + MPVAR(K4yaw)*iyaw)/100;
```

Pour stabiliser le drone, l'utilisateur doit donc régler huit coefficients. Pour un réglage facile, ces coefficients sont liés à des variables systèmes que l'on peut facilement modifier dans Horizon. Il aurait été plus propre d'utiliser des variables locales et un contrôle direct par le port série, mais par manque de documentation concernant le contrôle direct de Micropilot, nous avons dû nous résigner à utiliser Horizon. De plus amples renseignements sont détaillés dans l'annexe « utilisation de la carte Micropilot ».

RÉPARTITION DE LA CORRECTION SUR LES MOTEURS

La commande est maintenant construite, elle est alors répartie sur les moteurs, suivant les conventions de signe définies précédemment et les principes de vol du quadrirotor détaillés dans la présentation du projet.

```
//répartition de la correction sur chaque moteur et mise en forme


mot1 = (-pitchmix-yawmix)*0.3;

mot2 = (+rollmix+yawmix)*0.3;

mot3 = (+pitchmix-yawmix)*0.3;

mot4 = (-rollmix+yawmix)*0.3;
```

Le 0.3 est une mise en forme rapide qui permet de dire que grosso modo 30% de la poussée des moteurs sera dédiée à la correction d'attitude. Le reste est consacré à la sustentation.

MISE EN FORME DE LA POUSSÉE

Reste à définir le terme de poussée. Nous commençons par lui appliquer un facteur de forme similaire au 0.3 précédent. Ce facteur doit être modifié par l'utilisateur de manière à assurer une puissance suffisante pour faire voler le drone tout en évitant la saturation d'un moteur lorsque les gaz sont à fond et que la correction est maximale. Logiquement, un choix de 0.7 évite toute saturation (car 0.3(correction) + 0.7(poussée) = 1). Cependant ce chiffre peut s'avérer insuffisant pour assurer la puissance nécessaire au vol. Charge à l'utilisateur de l'ajuster au mieux.

```
//mise en forme de la poussée, thrust représente 70% max de la poussée du moteur


thrust=(cthrust+32767)*0.7 - 32767;


        rolltrim=4000;
        pitchtrim=3500;
        if(thrust>-10000){
                rolltrim=4200;
                pitchtrim=4200;
        }
        if(thrust>4000){
                rolltrim=5000;
                pitchtrim=5000;
        }
```

Les dernières lignes consistent en une compensation manuelle des différences de vitesse moteur en fonction du régime. Il s'agit de la fameuse erreur statique : une même consigne sur chaque moteur provoque des vitesses de rotation différentes du fait des différences mécaniques des moteurs. Il n'y a aucun moyen avec Micropilot de mesurer la vitesse de rotation des moteurs pour les asservir. La solution de l'automaticien est donc l'ajout d'un effet intégral. On peut aussi compenser manuellement (d'où le suffixe « trim » des variables concernées) : c'est ce qui est fait ici. Cependant, cette compensation dépend des plages de régime moteur (d'où les boucles en if), et dépend bien sûr des moteurs. Ces valeurs sont donc obtenues après de nombreux essais sur le drone, et assurent une correction suivant trois plages de vitesses. Cette solution est loin d'être élégante, et est fastidieuse à mettre en place, mais elle marche.

COMMANDE DU DRONE

La dernière chose à faire est simplement de générer le signal envoyé aux servocommandes (donc aux moteurs). Ce signal est composé de l'addition de la correction, de la poussée, et de la compensation.

Le signal envoyé dépend également de la position de la télécommande. En effet, nous avons vu dans la partie acquisition des données que les variateurs n'acceptent de s'initialiser que s'ils démarrent en recevant une commande nulle (-32767). La manette des gaz dispose donc d'une zone basse dans laquelle les moteurs sont coupés. Cela est nécessaire au fonctionnement avec ces variateurs, et permet à coup sûr de couper les moteurs en plaçant les gaz à zéro. Il s'agit donc également d'un organe de sécurité.

```
//le if qui suit permet une initialisation correcte des moteurs et leur arrêt manette des gaz en bas
//en effet les variateurs doivent recevoir un zero (c.-à-d. -32767) de la carte pour s'initialiser

if(thrust<-30000){
        MPVAR(fServo5)=-32767;
        MPVAR(fServo6)=-32767;
        MPVAR(fServo7)=-32767;
        MPVAR(fServo8)=-32767;
}
else{

        //Une vérification est faite sur la commande pour éviter de saturer un moteur

        if((thrust+mot2)<32700){
                MPVAR(fServo5) = thrust+mot2+rolltrim;
        }
        else{MPVAR(fServo5)=32700;}

        if((thrust+mot3)<32700){
                MPVAR(fServo6) = thrust+mot3+pitchtrim;
        }
        else{MPVAR(fServo6)=32700;}

        if((thrust+mot4)<32700){
                MPVAR(fServo7) = thrust+mot4-rolltrim;
        }
        else{MPVAR(fServo7)=32700;}
```

```
                    if((thrust+mot1)<32700){
                        MPVAR(fServo8) = thrust+mot1-pitchtrim;
                    }
                    else{MPVAR(fServo8)=32700;}

                }
```

Il faut également noter qu'une vérification est effectuée avant l'écriture du signal pour s'assurer qu'aucun moteur ne reçoit un signal supérieur à 32767 (en fait 32700, on garde une marge de sécurité). Cela peut éviter l'emballement de certains moteurs.

CODE FINAL

Pour information, voici l'intégralité du code de commande. Il reprend bien sûr les éléments précédents, mais contient aussi les déclarations de variable, les attributions de variables système et la structure des méthodes du code Micropilot. C'est le code tel qu'il est exécuté par la carte.

```
#include "usercode.h"
#include "simdll.h"
#include "mpfields.h"
#include "mptypes.h"
#include "math.h"

#define MPVAR(p2var)     (*p2var)

int (*savedGetVarPointer)( void **result, int id);

int getMPVarPointer( void **result, int id) {   return (*savedGetVarPointer)( result, id);  }

int mpUserInit( int (*function)( void **result, int id))
{
  savedGetVarPointer = function;
  return USER_RETURN_INIT_OK;
}

/*
 * mpUserEvent - This is the function that is called when the user code is implimented
 */
int mpUserEvent( long *result, int userEventIdentifier)
{
//-- MP Vars ------------------------------------------------------------

            //variables des moteurs
static long* fServo5;
static long* fServo6;
static long* fServo7;
static long* fServo8;
```

```
//variables d'attitude mesurées
static long* mpitch;
static long* myaw;
static long* mroll;
//vars d'attitude dérivée (vitesse) mesurées
static long* mppitch;
static long* mproll;
static long* mpyaw;
//vars d'attitude dérivées (vitesse) mises à l'échelle
static long ppitch;
static long proll;
static long pyaw;
//vars d'attitude dérivée deux fois (accélération)
static long pppitch;
static long pproll;
static long ppyaw;

//variables des données de la télécommande
static long* tpitch;
static long* troll;
static long* tyaw;
static long* tthrust;
//variables des commandes de position (var de la telecommande apres mise en forme)
static long cpitch;
static long croll;
static long cyaw;
static long cthrust;
//variables des commandes de correction de position
static long pitch=0;
static long roll=0;
static long yaw=0;
static long thrust;
//variables des commandes moteur
static long pitchmix;
static long rollmix;
static long yawmix;
static long mot1;
static long mot2;
static long mot3;
static long mot4;

//vars utilisées pour la mise en forme des signaux

static long zerothrust;
static long zeroyaw;
static long zeroroll;
static long zeropitch;

//vars utilsées pour le calcul de l'accélération
static long p1pitch=0;
static long p1roll=0;
static long p1yaw=0;
static long p2pitch=0;
static long p2roll=0;
static long p2yaw=0;

//valeurs utilisées pour le rate limiter
```

```
static long c1pitch=0;
static long c2pitch=0;
static long c1roll=0;
static long c1yaw=0;
static long c2roll=0;
static long c2yaw=0;
static long dpitch;
static long droll;
static long dyaw;


//coefficients des correcteurs PID
static long* K1roll;
static long* K2roll;
static long* K3roll;
static long* K1yaw;
static long* K2yaw;
static long* K3yaw;

//variables diverses
static int init=1;
static long* offground;
static long* radioon;
static long rolltrim;
static long pitchtrim;

//variables integrales
static long* K4roll;
static long* K4yaw;
static long ipitch=0;
static long iroll=0;
static long iyaw=0;




//------------------------------------------------------------------------
	switch( userEventIdentifier)
	{
	//------------------------------------------------------------------------
	//
	case USER_HARDWARE_INITIALIZED :
		getMPVarPointer( &fServo5, MPFID_FINE_SERVO_BASE + SERVO_FIVE );
		getMPVarPointer( &fServo6, MPFID_FINE_SERVO_BASE + SERVO_SIX );
		getMPVarPointer( &fServo7, MPFID_FINE_SERVO_BASE + SERVO_SEVEN );
		getMPVarPointer( &fServo8, MPFID_FINE_SERVO_BASE + SERVO_EIGHT );

		getMPVarPointer( &mpitch, MPFID_CURRENT_PITCH );
		getMPVarPointer( &myaw, MPFID_CURRENT_YAW );
		getMPVarPointer( &mroll, MPFID_CURRENT_ROLL );

		getMPVarPointer( &mppitch, MPFID_INST_PITCHDOT );
		getMPVarPointer( &mproll, MPFID_INST_ROLLDOT );
		getMPVarPointer( &mpyaw, MPFID_INST_YAWDOT );
```

```
		getMPVarPointer( &tpitch, MPFID_ELEVATOR_PULSE );
		getMPVarPointer( &troll, MPFID_AILERON_PULSE );
		getMPVarPointer( &tyaw, MPFID_RUDDER_PULSE );
		getMPVarPointer( &tthrust, MPFID_THROTTLE_PULSE );

		getMPVarPointer( &K1roll, MPFID_ROTATION_SPEED );
		getMPVarPointer( &K2roll, MPFID_CLIMB_SPEED );
		getMPVarPointer( &K3roll, MPFID_APPROACH_SPEED );
		getMPVarPointer( &K1yaw, MPFID_CLIMB_MARGIN );
		getMPVarPointer( &K2yaw, MPFID_WAYPOINT_DIA );
		getMPVarPointer( &K3yaw, MPFID_CCT_ALTITUDE );

		getMPVarPointer( &offground, MPFID_OFF_GROUND );
		getMPVarPointer( &radioon, MPFID_PIC_CIC );

		getMPVarPointer( &K4roll, MPFID_FLARE_ALTITUDE );
		getMPVarPointer( &K4yaw, MPFID_DESCENT_RATE );

		MPVAR(fServo5)=-32767;
		MPVAR(fServo6)=-32767;
		MPVAR(fServo7)=-32767;
		MPVAR(fServo8)=-32767;


		return USER_RETURN_IGNORE;
	//-------------------------------------------------------------------------
	// PID loop works with 5/30 Hz
	case USER_PID1 :

		//initialisation de la carte, on récupère les valeurs zero de la télécommande
		//ce code n'est exécuté qu'une fois au démarrage de la carte
		if(init==1){

			zerothrust=MPVAR(tthrust);
			zeroyaw=MPVAR(tyaw);
			zeroroll=MPVAR(troll);
			zeropitch=MPVAR(tpitch);
			c2pitch=zeropitch;
			c2roll=zeroroll;
			c2yaw=zeroyaw;
			init=0;
		}

		MPVAR(offground)=0;

		//mise en forme des signaux envoyés par la télécommande en signaux de grandeurs
comparable aux fServo (variant entre -32767 et +32767)

		croll = (MPVAR(troll) - zeroroll)*32767/600;
		cpitch = (MPVAR(tpitch) - zeropitch)*32767/600;
		cyaw = (MPVAR(tyaw) - zeroyaw)*32767/1000;
		cthrust = -32767+(MPVAR(tthrust)-zerothrust)*32767/1000;//la position 0 pour les gazs
correspond à la manette en bas, donc à -32767
```

```
//calcul de l'accélération   le 0.24 est une échelle de temps
p1pitch=p2pitch;
p2pitch=MPVAR(mppitch);
pppitch=(p2pitch - p1pitch)*100/24;

p1roll=p2roll;
p2roll=MPVAR(mproll);
pproll=(p2roll-p1roll)*100/24;

p1yaw=p2yaw;
p2yaw=MPVAR(mpyaw);
ppyaw=(p2yaw-p1yaw)*100/24;

//calcul du rate limiter
c1pitch = c2pitch;
c1roll = c2roll;
c1yaw = c2yaw;

c2pitch = cpitch;
c2roll = croll;
c2yaw = cyaw;

dpitch=(c2pitch-c1pitch);
droll=(c2roll-c1roll);
dyaw=(c2yaw-c1yaw);

if(dpitch>400){dpitch=400;}//le rate limiter filtre à partir d'une commande de plus de 11° par
seconde
if(dpitch<-400){dpitch=-400;}
if(droll>400){droll=400;}
if(droll<-400){droll=-400;}
if(dyaw>400){dyaw=400;}
if(dyaw<-400){dyaw=-400;}

cpitch = c1pitch + dpitch;
croll = c1roll + droll;
cyaw = c1yaw + dyaw;



//calcul de l'erreur de position
//on supposera que le drone ne peut pas excéder des angles de 30 degrés et 45 degrés en
lacet
pitch = cpitch-MPVAR(mpitch)*32767/540; // 540 = 30° en radians * 1024
roll = croll-MPVAR(mroll)*32767/540;
yaw = cyaw-MPVAR(myaw)*32767/810; //810=45° en rad *1024

//calcul de l'effet integral
ipitch = ipitch + pitch;
iroll = iroll + roll;
iyaw = iyaw + yaw;
if(ipitch>10000){ipitch=10000;}
if(ipitch<-10000){ipitch=-10000;}
if(iroll>10000){iroll=10000;}
if(iroll<-10000){iroll=-10000;}
if(iyaw>10000){iyaw=10000;}
if(iyaw<-10000){iyaw=-10000;}
```

```
		//mise à l'échelle des vitesses et accélérations (sur une échelle de -32767 à 32767)
		ppitch = MPVAR(mppitch)*32767/2000;
		proll = MPVAR(mproll)*32767/2000;
		pyaw = MPVAR(mpyaw)*32767/2000;

		pppitch = pppitch*32767/2000;
		pproll = pproll*32767/2000;
		ppyaw = ppyaw*32767/1000;




		//calcul du correcteur
			//le /100 permet d'utiliser des coeffs 100 fois plus grands (plus lisible)
		pitchmix =  (MPVAR(K1roll)*pitch - MPVAR(K2roll)*ppitch - MPVAR(K3roll)*pppitch +
MPVAR(K4roll)*ipitch)/100;

		rollmix =  (MPVAR(K1roll)*roll - MPVAR(K2roll)*proll - MPVAR(K3roll)*pproll +
MPVAR(K4roll)*iroll)/100;

		yawmix =  (MPVAR(K1yaw)*yaw - MPVAR(K2yaw)*pyaw - MPVAR(K3yaw)*ppyaw +
MPVAR(K4yaw)*iyaw)/100;



		//répartition de la correction sur chaque moteur et mise en forme

		mot1 = (-pitchmix-yawmix)*0.3;
		mot2 = (+rollmix+yawmix)*0.3;
		mot3 = (+pitchmix-yawmix)*0.3;
		mot4 = (-rollmix+yawmix)*0.3;

		//mise en forme de la poussée, thrust représente 80% max de la poussée du moteur

		thrust=(cthrust+32767)*0.8 - 32767;


			rolltrim=4000;
			pitchtrim=3500;
			if(thrust>-10000){
				rolltrim=4200;
				pitchtrim=4200;
			}
			if(thrust>4000){
				rolltrim=5000;
				pitchtrim=5000;
			}


		//le if qui suit permet une initialisation correcte des moteurs et leur arrêt manette des gazs en
bas
		//en effet les variateurs doivent recevoir un zero (c-à-d -32767) de la carte pour s'initialiser

		if(thrust<-30000){
			MPVAR(fServo5)=-32767;
			MPVAR(fServo6)=-32767;
```

```
                        MPVAR(fServo7)=-32767;
                        MPVAR(fServo8)=-32767;
                }
                else{

                        //Une vérification est faite sur la commande pour éviter de saturer un moteur

                        if((thrust+mot2)<32700){
                                MPVAR(fServo5) = thrust+mot2+rolltrim;
                        }
                        else{MPVAR(fServo5)=32700;}

                        if((thrust+mot3)<32700){
                                MPVAR(fServo6) = thrust+mot3+pitchtrim;
                        }
                        else{MPVAR(fServo6)=32700;}

                        if((thrust+mot4)<32700){
                                MPVAR(fServo7) = thrust+mot4-rolltrim;
                        }
                        else{MPVAR(fServo7)=32700;}

                        if((thrust+mot1)<32700){
                                MPVAR(fServo8) = thrust+mot1-pitchtrim;
                        }
                        else{MPVAR(fServo8)=32700;}


                }

                return USER_RETURN_REPLACE;
        //---------------------------------------------------------------------------
        // PID loop works with 5/30 Hz
        case USER_PID2 :
                return USER_RETURN_REPLACE;
        //---------------------------------------------------------------------------
        // Event not handled
        default:
                return USER_RETURN_UNUSED;
        }
}
```

STABILISATION DU DRONE

FABRICATION D'UNE PLATEFORME D'ESSAIS

Nous savions que lorsque les PDD^2(proportionnel, dérivée, dérivée seconde) contrôlant la stabilité autour des différents axes auraient été codés dans la carte MicroPilot, il nous faudrait déterminer leurs coefficients.

Comme nous n'avions pas le modèle du système et qu'il aurait été trop difficile de l'identifier, ces coefficients devaient être trouvés de manière empirique, grâce à des tests. Pour cela, nous avons fabriqué une plateforme d'essais qui nous permette de tester la stabilité autour des axes de roulis et de tangage. Ainsi, nous avons pu observer les effets d'une large gamme de coefficients pour les PDD^2 contrôlant la stabilité autour de ces axes et décider de ceux qui étaient les plus efficaces.

Cette plateforme se présente sous la forme d'un support d'une hauteur de 1m20 environ, sur lequel on fixe l'un des axes (de roulis ou de tangage) du drone. Le quadrirotor ne possède alors plus qu'un degré de liberté : la rotation autour de l'axe fixé. On peut ainsi tester sans risque la stabilité du drone autour de cet axe.

La plateforme est constituée de deux parties principales : le support en bois et les pièces métalliques qui permettent de fixer le drone sur le support.

SUPPORT EN BOIS

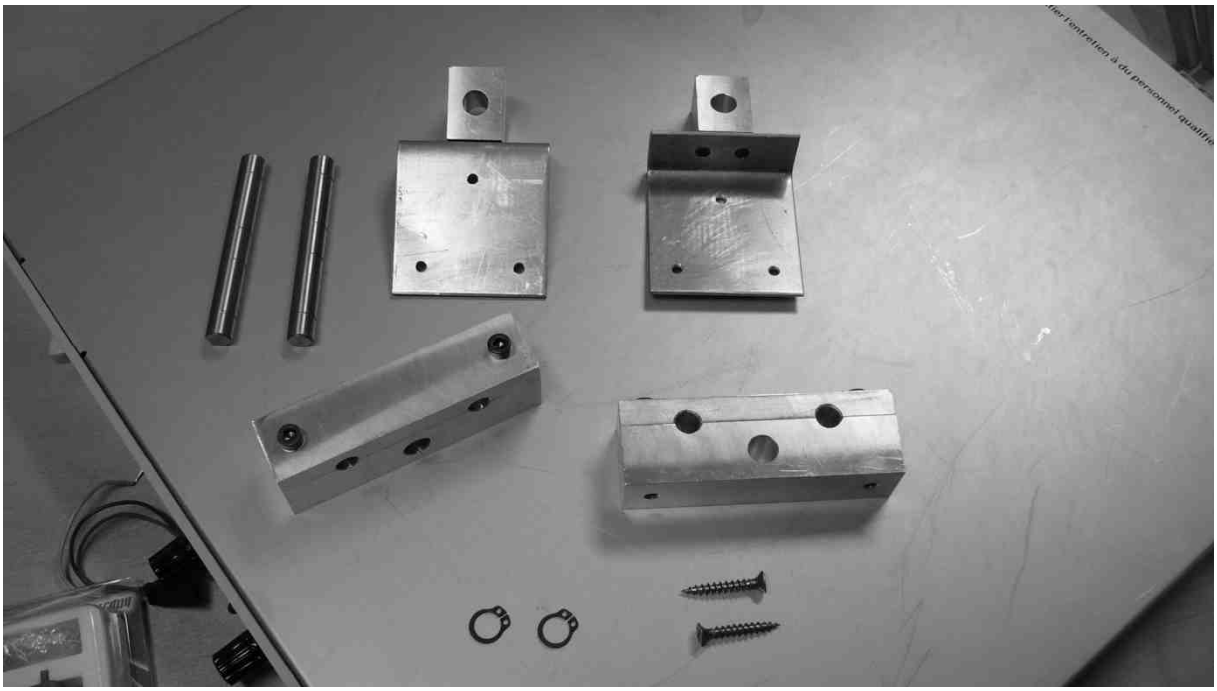
Il est constitué de deux triangles équilatéraux, face à face, tenus à 80 cm l'un de l'autre par de solides planches de contreplaqué (placées suffisamment bas pour ne pas gêner les rotations du quadrirotor). Le drone est fixé au niveau des sommets des deux triangles, grâce à des pièces métalliques décrites ci-dessous. Nous avons choisi d'utiliser des triangles équilatéraux car un angle de 60° est suffisamment important pour éviter un basculement du support sur un des côtés, et nous a permis d'avoir un support assez haut sans toutefois avoir un encombrement au sol trop important. Nous avons pu, grâce à la hauteur du support, travailler de manière confortable sur le drone sans avoir à l'ôter de la plateforme d'essais.

Un des triangles du support

Plateforme d'essais

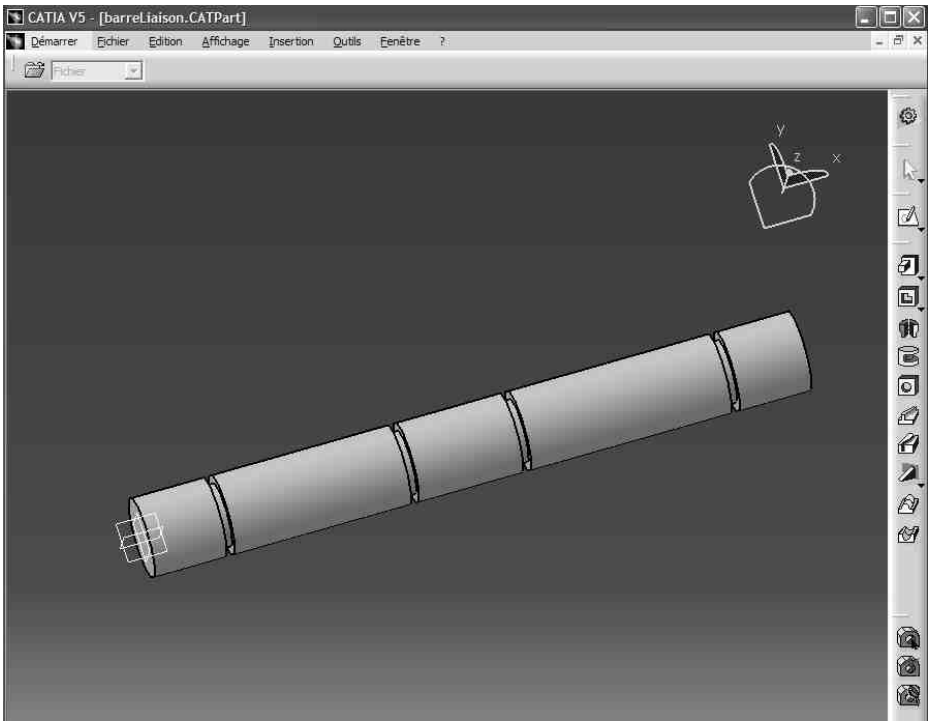
PIECES METALLIQUES DE LIAISON ENTRE LE SUPPORT EN BOIS ET LE DRONE

Ces pièces permettent de créer une liaison pivot entre le support, qui est fixe, et l'axe de roulis ou de tangage du quadrirotor. L'une des pièces est fixée à l'aide de deux vis au support, une autre est fixée sur l'axe du drone, il s'agit d'une sorte de mors dans lequel sont tenues les deux barres de carbone. Ces deux pièces en aluminium sont liées par un axe en acier autour duquel elles peuvent tourner. Pour empêcher les mouvements de translation des pièces le long de l'axe, des circlips ont été placés, bien que ceux-ci créent des frottements et gênent la rotation des pièces autour de l'axe. Toutes ces pièces ont été fabriquées en double car le drone est relié en deux points au support.



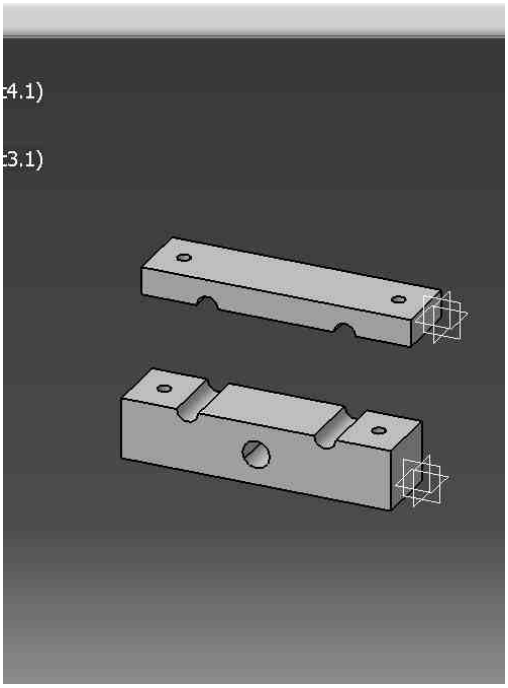
Pièces de liaison entre le support et le quadrirotor

Les axes en aciers ont été usinés pour faciliter la mise en place de circlips : des raies correspondant au diamètre des circlips ont été creusées aux endroits adéquats.

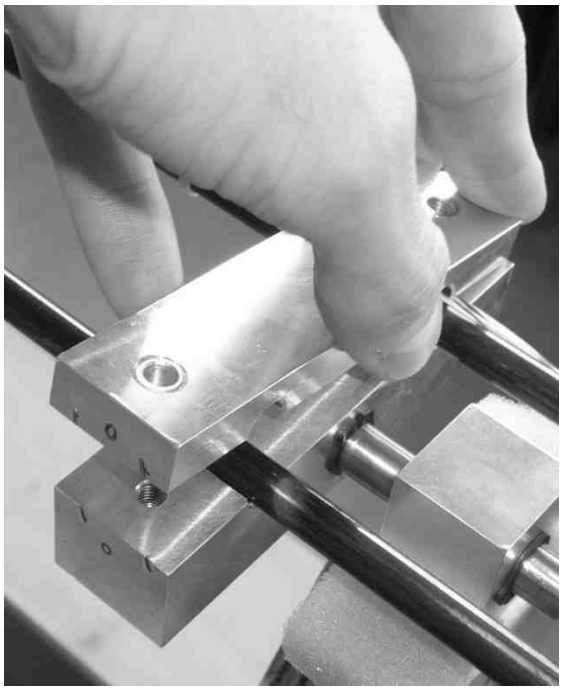


Conception CATIA : Axe en acier

Les mors sont constitués de deux pièces que l'on peut fixer de part et d'autres des deux barres de carbone. Au centre de la pièce inférieure, on a percé un trou dans lequel vient s'insérer l'axe en acier.



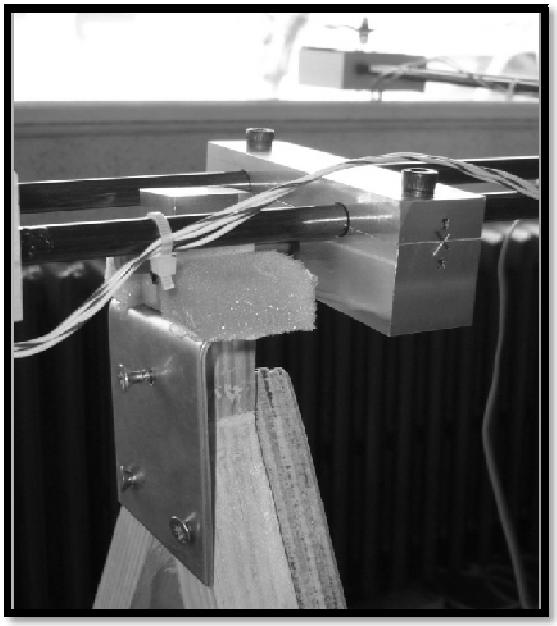
Conception CATIA : Mors.



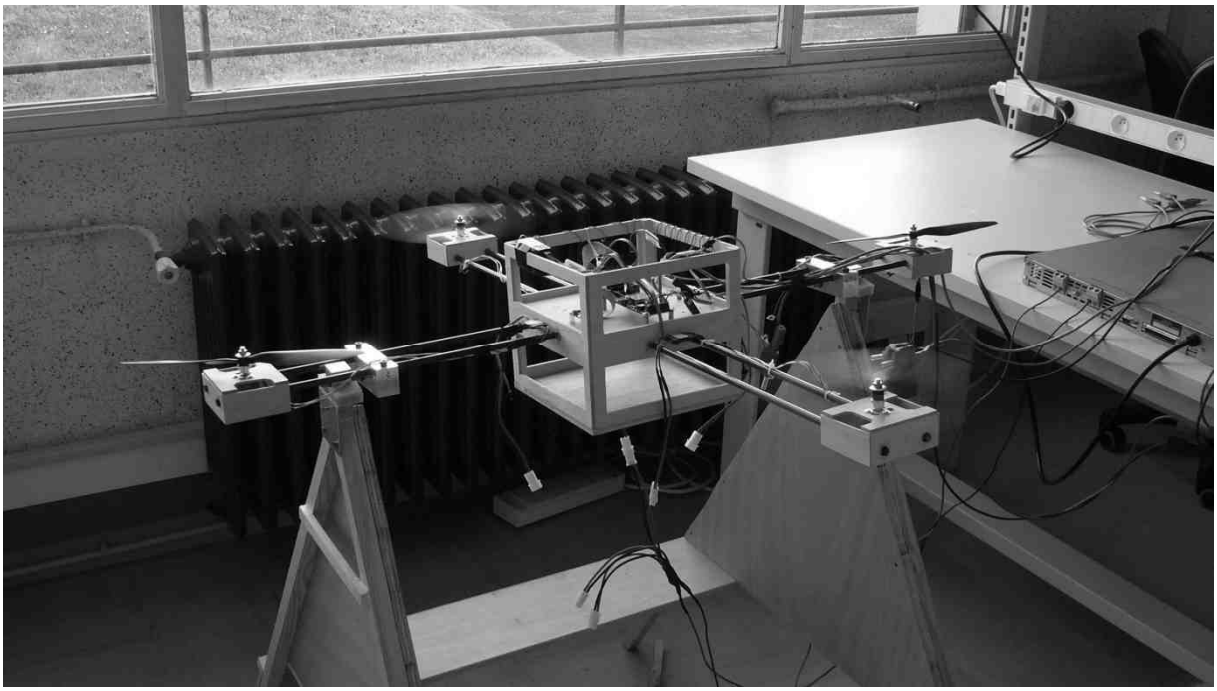
Mise en place du mors sur les barres de carbone

Cette plateforme nous a permis de réaliser les tests nécessaires pour choisir les coefficients des . Cependant, à cause des frottements s'opposant à la rotation du drone autour de son axe, elle ne nous a pas permis de travailler avec une grande précision. L'une des causes principales de ces frottements provenait du fait qu'au cours des tests, seuls les deux moteurs n'appartenant pas à l'axe fixé fonctionnaient. Comme les moteurs face à face tournent dans le même sens, un moment de lacet était créé. Ce moment crée un effort latéral indésirable entre les mors et les axes en acier.

Enfin, les pièces métalliques fixées au support empêchent une rotation complète du drone autour de son axe. Pour éviter que les tiges de carbone ne viennent heurter violemment cette pièce et s'abiment, nous avons placé de part et d'autre de la pièce deux petits cubes de mousse.



Photographie des pièces de liaison montées sur le support

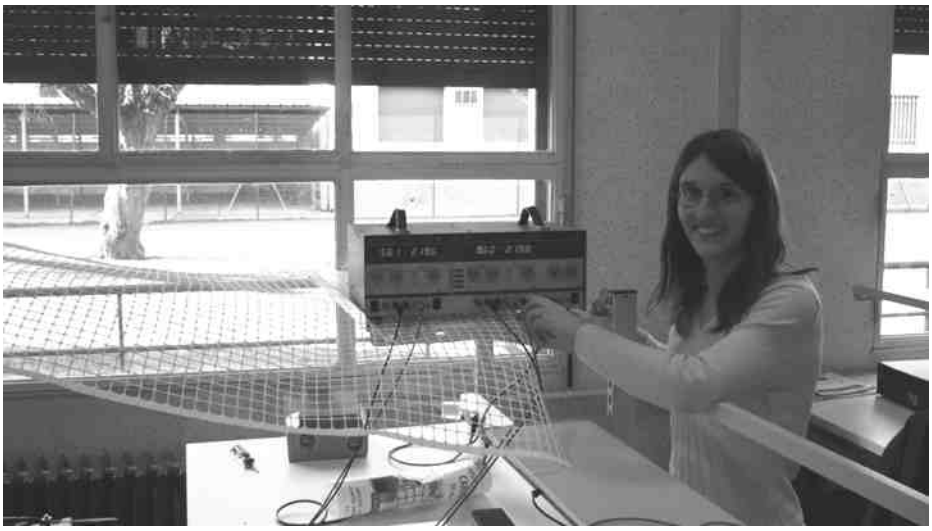


Quadrirotor sur la plateforme d'essais lors d'un test

INSTALLATION DU DRONE POUR LES TESTS

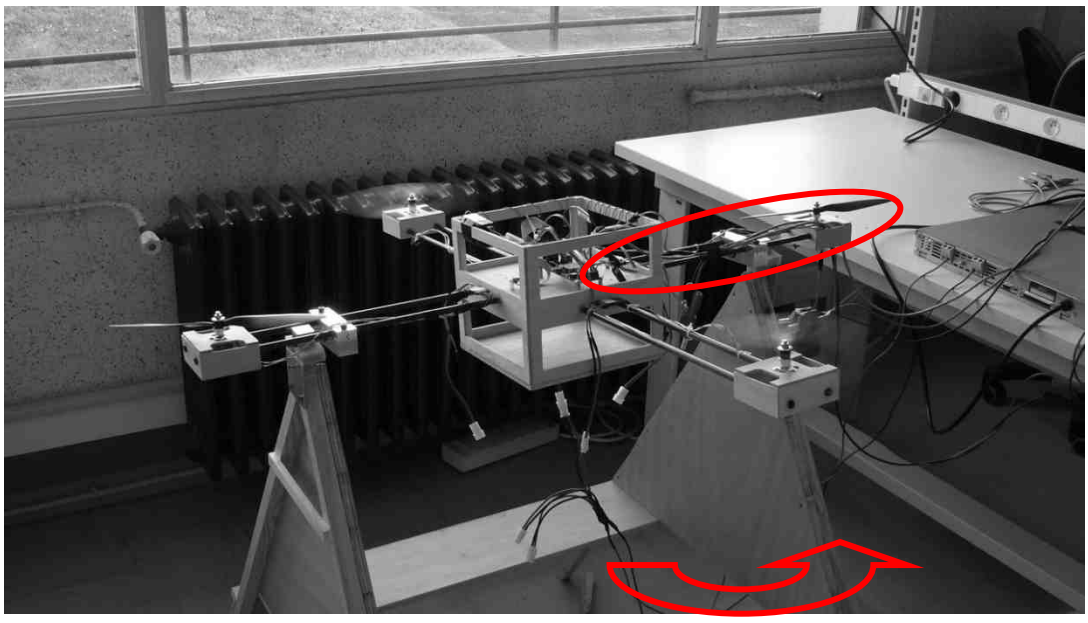
Pour pouvoir travailler en toute sécurité, le support et le drone ont été installés au fond de la salle B10, deux tables ont été placées de manière à éviter que des élèves peu éveillés ne viennent approcher de trop près les hélices en rotation, puis on a tendu un filet au dessus pour contrer toute projection. Le matériel nécessaire aux tests a été placé sur les tables : alimentations, ordinateur portable, télécommande.

Branchements : Les tests ont été réalisés sans batteries, nous avons donc utilisé des alimentations pour alimenter la carte MicroPilot, le récepteur, le servos Board et les moteurs. (Voir la partie branchements et alimentation dans les annexes.)

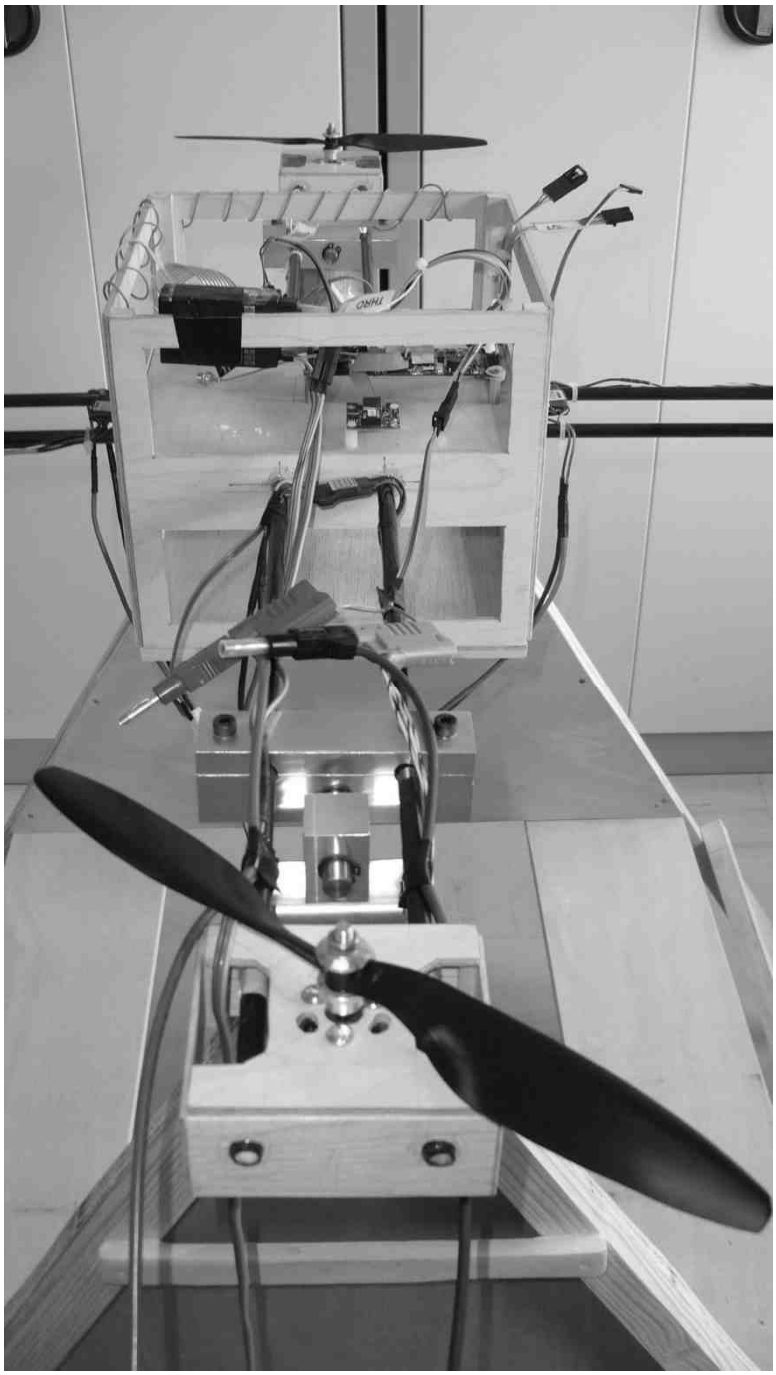


Alimentation de la carte MicroPilot, du récepteur et du servos Board

Cependant, il a fallu faire attention aux fils reliant les différents éléments aux alimentations, il ne fallait surtout pas qu'ils puissent entrer en contact avec les hélices lors de la rotation du drone autour d'un de ses axes. Les câbles d'alimentation de la carte, du servos Board et du récepteur ont été fixés avec du scotch sur les barres de carbones fixés sur le support, jusqu'à l'alimentation. Les câbles d'alimentation des moteurs ont été passés par la partie inférieure du triangle du support le plus proche de l'alimentation.



Branchements



Fixation des câbles d'alimentation de la carte MicroPilot et des éléments associés
Sur les barres de carbone

PROTOCOLE DE TEST ET STABILISATION EN TANGAGE ET ROULIS

PROTOCOLE DE TEST

A l'aide de cette plateforme de test et de la carte MicroPilot, nous avons pu tester différents types de correcteurs en partant du simple proportionnel jusqu'à un correcteur $PIDD^2$ sur la position angulaire. Le but de ces essais était de trouver le correcteur qui assure au quadrirotor la meilleure stabilité possible autour de ses axes de roulis et de tangage.

Nous avons commencé par tester la stabilité autour de l'axe de tangage. Cet axe du drone a donc été fixé sur le support et seuls les moteurs 1 et 3 ont été branchés à l'alimentation.

Le protocole réalisé pour chaque test a été le suivant :

1. Chargement des nouveaux coefficients
2. Mise en marche du quadrirotor
3. Impulsion appliquée selon le mouvement étudié (ici autour de l'axe tangage) grâce à la télécommande
4. Décompte des oscillations permettant de retrouver la position initiale

Aucun des correcteurs P (proportionnels) et PD (proportionnel, dérivée) que nous avons testé n'a donné de résultats satisfaisants. Le système était toujours instable ou dans le meilleur des cas très oscillant. Si nous avions pu tracer le lieu des racines du système, il aurait sans doute été très éloigné de l'axe des réels (amortissement faible).

Nous avons donc décidé d'utiliser unPDD^2. La donnée d'accélération permet d'obtenir une information plus rapide sur la position du drone. Cette technique nous permet d'anticiper sur la modification de vitesse angulaire, on dépasse ainsi moins violemment la position souhaitée : les oscillations sont réduites.

CHOIX DES COEFFICIENTS

Nous avons effectué un raisonnement par expérience pour déterminer les meilleurs coefficients pour lePDD^2. 70 est une valeur moyenne pour les coefficients.

Chargement de nouveaux coefficients

Les premiers essais ont été réalisés avec une **puissance moyenne des moteurs assez faible**.

Proportionnel	vitesse	Accélération	Résultat du test
85	85	85	5 oscillations
85	100	85	3 oscillations
85	120	85	2 oscillations
85	150	85	0 oscillation, mais plutôt lent
85	200	85	Nombreuses oscillations

Nous en avons donc conclu qu'un PDD^2 suffirait pour stabiliser le système et que le coefficient de vitesse devait probablement être supérieur aux deux autres. Cependant nous avons alors voulu tester ces résultats pour une **puissance moyenne des moteurs élevée**.

Proportionnel	vitesse	Accélération	Résultat du test
85	85	85	5 oscillations
85	100	85	Infinité d'oscillations

Les coefficients qui donnent de bons résultats à faible puissance ne donnent pas forcément les mêmes résultats à puissance élevée. **Les tests suivants ont donc tous été réalisés à puissance élevée**.

Proportionnel	vitesse	Accélération	Résultat du test
70	70	70	5 oscillations
70	85	70	Stable mais très oscillant
70	100	70	3 oscillations puis pompage
70	120	70	Régime critique
100	140	100	Complètement instable

Nous avons déduit de ces tests que les coefficients utilisés jusqu'à présent étaient trop élevés.

Proportionnel	vitesse	Accélération	Résultat du test
60	80	65	Satisfaisant pour toutes puissances
60	90	65	Instable à grande vitesse
50	80	55	Satisfaisant
50	**75**	**55**	**Très bien pour toutes les puissances**

Nous avons donc conservé ces valeurs pour les coefficients. Nous avons alors testé ces coefficients pour la stabilité autour de l'axe de roulis, ils ont été aussi efficaces. Cela est tout à fait logique étant donné la symétrie du quadrirotor.

Cependant au cours des tests nous avons remarqué une erreur statique : le drone ne revenait pas tout à fait en position horizontale. Il y avait généralement un angle d'environ 5° entre l'axe libre du drone et l'horizontale. Cela est dû au fait que tous les moteurs ne sont pas parfaitement identiques. L'un des moteurs est forcément plus puissant que l'autre.

ANNULATION DE L'ERREUR STATIQUE

Nous avons tout d'abord essayé d'éliminer cette erreur statique en ajoutant un terme intégrateur à nos correcteurs. (voir la partie Code de commande.) Cependant un terme intégrateur ne fait pas qu'annuler l'erreur statique, il déstabilise également le système. Nous avons testé de très petites valeurs pour ce coefficient, pour aucune de ces valeurs, la stabilité du système n'a été satisfaisante.

Nous avons donc décidé d'éliminer cette erreur par correction directe sur la commande des moteurs : on a ajouté un terme sur le moteur le moins puissant et retranché ce même terme sur l'autre moteur.

Cependant l'erreur statique était proportionnelle à la puissance appliquée aux moteurs : plus les moteurs tournaient vite, plus l'erreur était importante. Nous avons donc appliqué des termes différents selon la puissance commandée. Plus la puissance est importante, plus le terme est grand.

Rolltrim est le terme ajouté ou retranché sur les moteurs en dehors de l'axe de roulis.

Pitchtrim est le terme ajouté ou retranché sur les moteurs en dehors de l'axe de tangage.

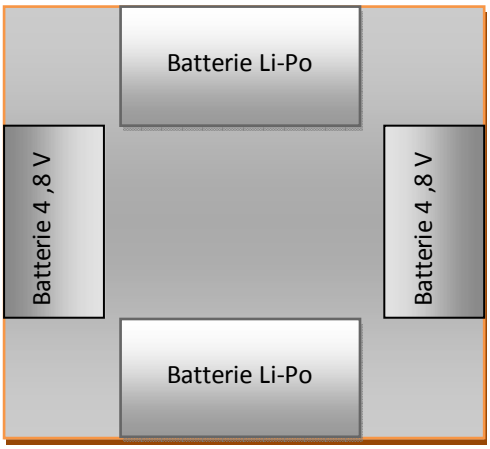
thrust	rolltrim	pitchtrim
-32767 à -10000	4000	3500
-10000 à +4000	4200	4200
+4000 à +32767	5000	5000

ESSAIS EN VOL

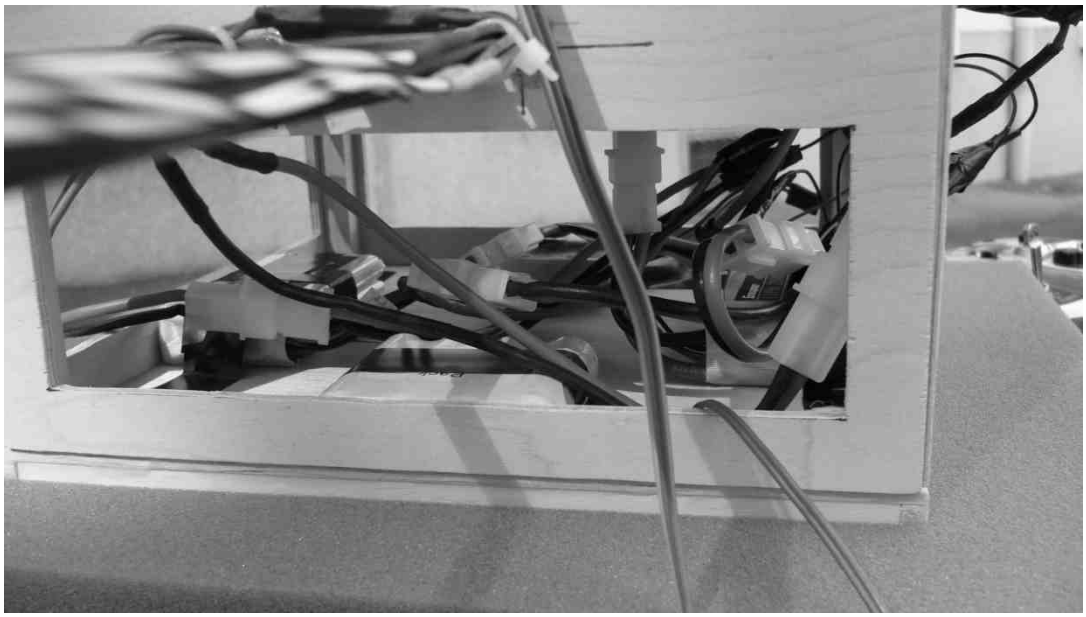
La stabilisation autour des axes de roulis et de tangage nous paraissant satisfaisante, nous avons décidé de la tester en vol. Ces essais en vol devaient, de plus, nous permettre de tester la stabilité autour de l'axe de lacet.

PRÉPARATION DU DRONE

Pour la première fois, nous devions utiliser des batteries à la place des alimentations. Nous avons donc placé et fixé les batteries sur le quadrirotor, dans le compartiment inférieur de la partie centrale. Les batteries ont été placées de manière symétrique pour ne pas perturber la stabilité du drone.

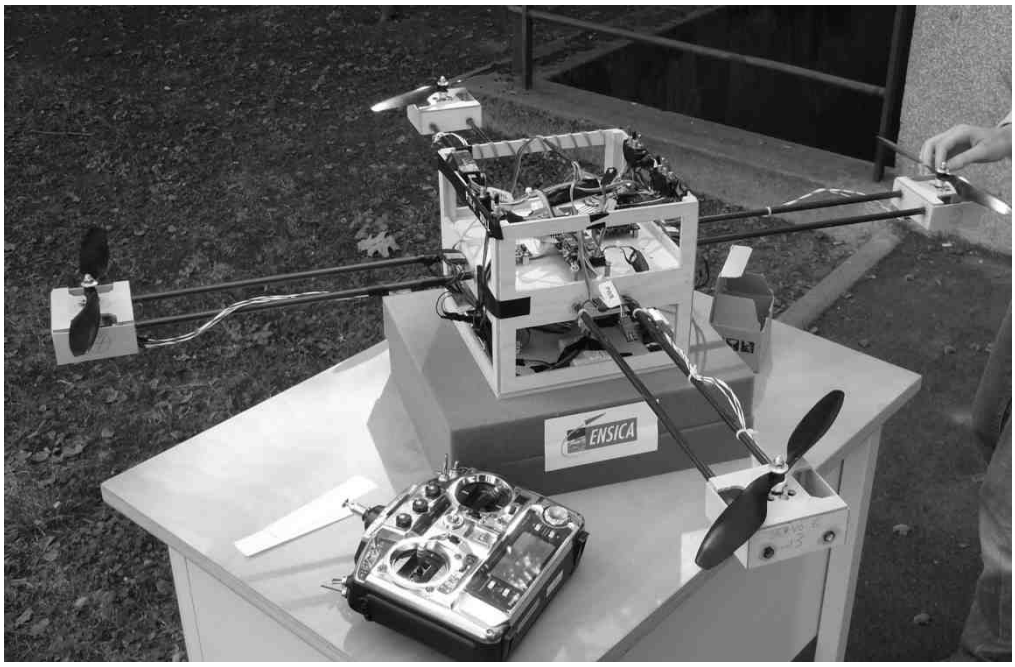


Répartition des batteries



Photographie du compartiment inférieur

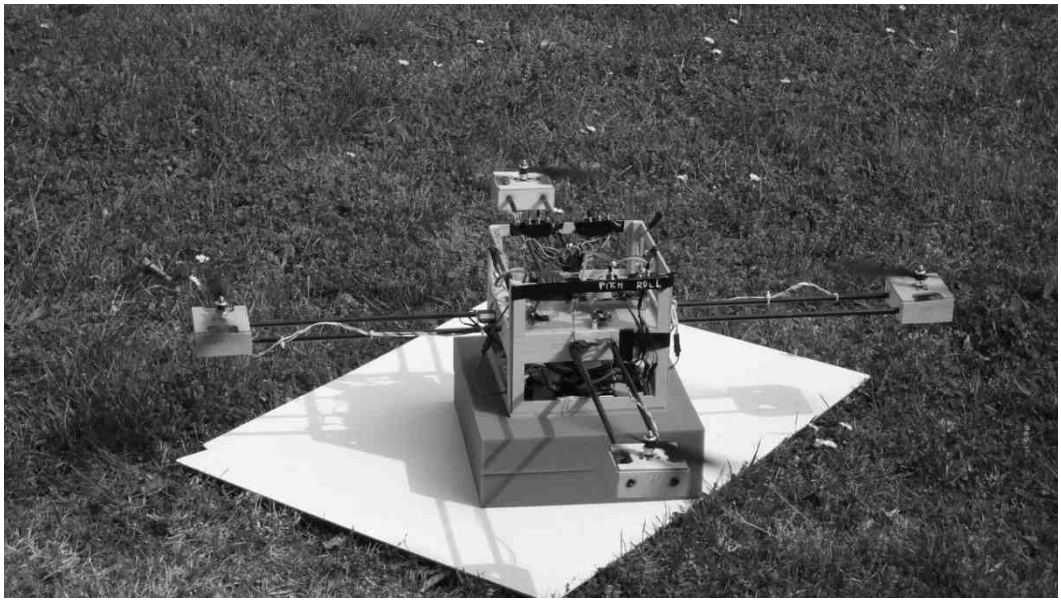
Nous avons placé de la mousse en dessous du drone pour amortir une éventuelle chute brutale. Nous avons tout d'abord placé deux pains de mousses superposés, fixés au quadrirotor grâce à du scotch double face. Enfin pour pouvoir piloter le drone, nous avons collé un autocollant ENSICA sur la face avant du drone.



Photographie du drone avant son premier essai de vol

LES ESSAIS DE VOL

Au cours des premiers essais, nous nous sommes rendus compte que le quadrirotor n'était pas suffisamment puissant pour se sustenter, et ce bien qu'il ne pèse que 1,6 Kg, masse bien inférieure à celle donnée comme maximale par le groupe ayant dimensionné le drone (2,8 Kg).



Drone en phase de décollage

Nous avons alors réfléchi aux meilleurs moyens de réduire la masse du quadrirotor.

Nous avons commencé par diminuer la quantité de mousse. Nous n'avons gardé qu'un seul pain de mousse et nous l'avons évidé au maximum.

Photographie de la mousse « amortisseur »

Le gain en masse étant très faible, le drone n'a pas davantage décollé. Nous avons alors réfléchi à la possibilité d'enlever le compartiment inférieur de la partie centrale du quadrirotor. Les batteries auraient alors été fixées entre le support en bois du compartiment supérieur et les barres de carbone. Cependant nous n'étions pas sûrs que le gain en masse soit suffisant pour permettre au drone de décoller.

De plus, il nous semblait qu'avec les batteries, les hélices tournaient moins vite qu'avec l'alimentation. Nous avons alors branché les moteurs sur l'alimentation. Une fois encore, le drone n'a pas quitté le sol.

Pour essayer d'évaluer la masse à gagner, nous avons ôté les deux batteries Li-Po, qui pèsent chacune 165 g. Malheureusement, même avec ces 330 g en moins, le drone n'a pas réussi à se sustenter. Nous sommes arrivés à la conclusion qu'un redimensionnement complet était nécessaire. (voir partie dimensionnement du drone).

Ces tests nous ont cependant montré que le quadrirotor est relativement stable autour de son axe de lacet. En effet, lorsqu'un côté du drone se soulevait, le quadrirotor ne tournait sur lui-même que très faiblement. Ce léger mouvement de lacet est très facilement pilotable grâce à la télécommande.

CALCULS DE DIMENSIONNEMENT DU DRONE ACTUEL

Au cours de nos essais en vol, nous ne sommes pas parvenus à faire décoller le drone. Manifestement, les moteurs ne pouvaient fournir la poussée nécessaire. Au cours de ces essais, le drone avait une masse de 1,6 kg.

L'équipe ayant fabriqué le drone affirme que chaque moteur est capable de porter au maximum 700 grammes, d'après la documentation du constructeur (qui explique qu'un de ces moteurs associé aux hélices choisies peut porter entre 200 et 700 grammes pour un avion de voltige effectuant du vol vertical). Ils en ont conclu que les quatre moteurs pourraient porter au maximum 2,8 kg.

Nous allons vérifier ce dimensionnement. Pour cela, nous utilisons la méthode du bilan de puissance, issue de la théorie linéarisée de Froude.

Pour un hélicoptère, la phase de vol consommant le plus d'énergie est le vol stationnaire. Si les moteurs sont capables de fournir l'énergie suffisante à ce vol, alors le drone pourra aborder toutes les phases de vol.

La puissance induite en vol stationnaire vaut :

$$P_{i0} = \frac{1}{\eta_i} \cdot \frac{F_n^{3/2}}{\sqrt{2\rho S}}$$

Avec :

$\frac{1}{\eta_i}$ Qualité induite

$\eta_i = \frac{B}{K}$ Où K est un coefficient tenant compte de l'irrégularité de vitesse induite le long de la pâle. Il dépend de l'adaptation de la corde et du vrillage. Pour un rotor vrillé comme ici, il vaut environ 1,07.

B est un coefficient tenant compte de l'irrégularité de portance, notamment des pertes en extrémité de pâle. En pratique, il est voisin de 0,96.

Fn = K M g le poids de l'appareil. K vaut environ 1,05 (K dépend de la forme du fuselage).

P est la masse volumique de l'air. A 25°, elle vaut environ 1,17 kg/m^3.

S est la surface du rotor

Les hélices retenues ont un diamètre de 25 cm, et une zone morte au milieu de 3 cm de diamètre. Le rotor a donc une surface de 0,048m². Les 4 hélices ont une surface de 0,194m².

La qualité induite est de 0,89.

Pour porter 2,8 kg, Pio doit donc être de 258 Watts, soit 65 Watts par moteur. Pour porter 1,6 kg, Pio doit être de 112 Watts, soit 28 Watts par moteur.

A cette puissance induite s'ajoute la puissance de profil, puissance due à la trainée des hélices.

$$P_{po}=\rho/8\ S\sigma\ Cx\ U^3$$

Sσ est la plénitude du rotor multipliée par la surface du rotor, c'est-à-dire la surface des pâles. Ici, il y a deux pâles de 0,12m de rayon, et d'environ 1,5dm d'épaisseur.

TYPHOON MICRO 6/2310 PROMODEL									
Diamètre 29 mm	Longueur 27,5 mm	Poids 41,15 g	Arbre 3,17 mm / Fixation : 3 vis M3			10 pôles	APPLICATIONS		
Courant Io 0,33 A	Résistance 0,18 Ω	Kv 1360 tr/V	P nominale 90 W	Courant Max 12 A	Rendement 84,6 %	23 spires	Indoors / ParkFlyers 3D jusqu'à 700 g		
HELICE	APC 8 x 6			APC 9 x 4,7			APC 10 x 4,7		
TENSION	Courant	Puissance	Régime	Courant	Puissance	Régime	Courant	Puissance	Régime
6 V	8,2 A	49,2 W	5450 Tr/Min	7,3 A	43,8 W	5700 Tr/Min	9,6 A	57,6 W	4800 Tr/Min
7 V	10,3 A	72,1 W	6050 Tr/Min	9,0 A	63,0 W	6350 Tr/Min	11,3 A	79,1 W	5150 Tr/Min
8 V				10,6 A	84,8 W	6900 Tr/Min			

U est la vitesse périphérique. Pour le moteur utilisé (Typhoon Micro 6/23), la vitesse de rotation est d'environ 5000tr/min lorsque les gaz sont à fond, sous une tension d'alimentation d'au plus 7V. Car les hélices utilisées ont un APC de 10x4,5.

Cx est le coefficient de traînée. Pour un nombre de Mach faible (<0,6) on pourra supposer que :

$$Cx = 0{,}008 + 0{,}009\ Cz^2$$

Cz est usuellement compris entre 0,4 et 0,6. Prenons 0,4, ce qui donne Cx=0,009.

Une pâle ayant un rayon de 12,5 cm, on trouve U=52m/s. Soit 187 km/h, ce qui semble un ordre de grandeur correct.

La puissance de profil est alors de Ppo=1,8W par moteur.

La puissance que devra développer chaque moteur est la somme de la puissance induite et de la puissance de profil. Pour lever 1,6 kg, un moteur doit donc développer 29,8W. Pour lever 2,8kg, il doit développer 66,8W.

Ces calculs théoriques ne tiennent pas compte de l'efficacité sustentatrice. Cette efficacité qualifie la capacité des pâles à soulever la charge. Cette efficacité, aussi appelée figure de mérite, est en quelque sorte le rendement du rotor.

$$\eta_s = P_{io}/P_r$$

La figure de mérite maximale des rotors actuels vaut environ 0,6. La puissance réelle Pr est donc la puissance théorique divisée par 0,6.

Pour lever 1,6kg, chaque moteur doit donc développer 50 W. Pour lever 2,8kg, chaque moteur doit développer 111 W.

Au cours de nos tests, nous n'avons guère pu dispenser plus de 30 Ampères pour tous les moteurs, dans la gamme de tension des variateurs (7-12V). Ainsi, la puissance électrique maximale dispensée aux moteurs a été d'environ 330W. Soit 82,5W par moteur. Les moteurs ayant un rendement de 80%, ils n'ont donc fourni aux rotors que 66W.

Théoriquement, il aurait donc été possible de soulever le drone à 1,6kg, puisque nous aurions disposé de 16W de marge. Cependant, il ne faut pas oublier que de nombreuses valeurs utilisées dans ce calcul sont issues de

méthodes de pré-dimensionnement assez simple. Par exemple, la figure de mérite qualifie la moyenne des rotors utilisés sur hélicoptère. Ces rotors sont sans doute plus optimisés que nos hélices d'avion de modélisme.

La marge de 16W est donc à nuancer, elle est probablement moindre dans la réalité. D'autant que nous n'avons pas pris en compte la consommation électrique des variateurs, pourtant non négligeable.

Enfin, on remarque que pour une masse de 1,6kg, la sustentation du drone est limite, et que les moteurs doivent tourner à leur maximum pour lever l'engin. Hors la manœuvrabilité requiert un surplus de puissance (certains moteurs tournent plus vite).

Enfin la puissance électrique n'a pu atteindre 330W qu'avec une alimentation externe stabilisée. Avec la marge de tension des variateurs utilisés, nous ne pouvons utiliser que les accus Li-Po détaillés en annexe, qui ne peuvent dégager une telle puissance.

La sustentation à 1,6kg est donc limite, en pratique, force est de constater que le drone ne vole pas. Quand au dimensionnement qui fixait la masse embarquée à 2,8kg, il va sans dire qu'elle a été donnée par un constructeur « oubliant » malencontreusement l'efficacité sustentatrice pour évaluer les performances de son moteur.

Pour faire voler le drone en l'état actuel il faut donc l'alléger, et disposer de plus de puissance pour l'alimentation. Il faut donc changer les variateurs. On pourrait alors utiliser les batteries du drone dragonflyer, qui fourniraient une puissance suffisante. Mais ces batteries sont bien plus lourdes… Hors les gains de masse structuraux que l'on peut effectuer sont minimes. De l'ordre de 300g grand maximum…

Dans l'état actuel, le drone est fabriqué à la limite de ses performances, et la pratique montre qu'il ne peut voler.

PRÉSENTATION D'UNE SOLUTION FUTURE

Dans l'état actuel, le drone présente un système de commande qui nous l'avons vu, permet d'assurer simplement la stabilité. Le code est adaptable à un autre type de structure. Il suffit pour cela de mesurer à nouveau les coefficients et ordres de grandeur utilisés, pour s'adapter à une autre architecture de quadrirotor.

Nous pensons que la pérennité de ce projet passe par un travail de construction réfléchi d'une plate-forme de quadrirotor performante et de bonne finition. Il sera alors aisé de travailler sur une base saine et évolutive. Le travail qui suit constitue un guide à la réalisation de cette architecture.

CONCLUSION DE L'ÉTUDE PRÉCÉDENTE

Le bilan de puissance du quadrirotor montre diverses choses :

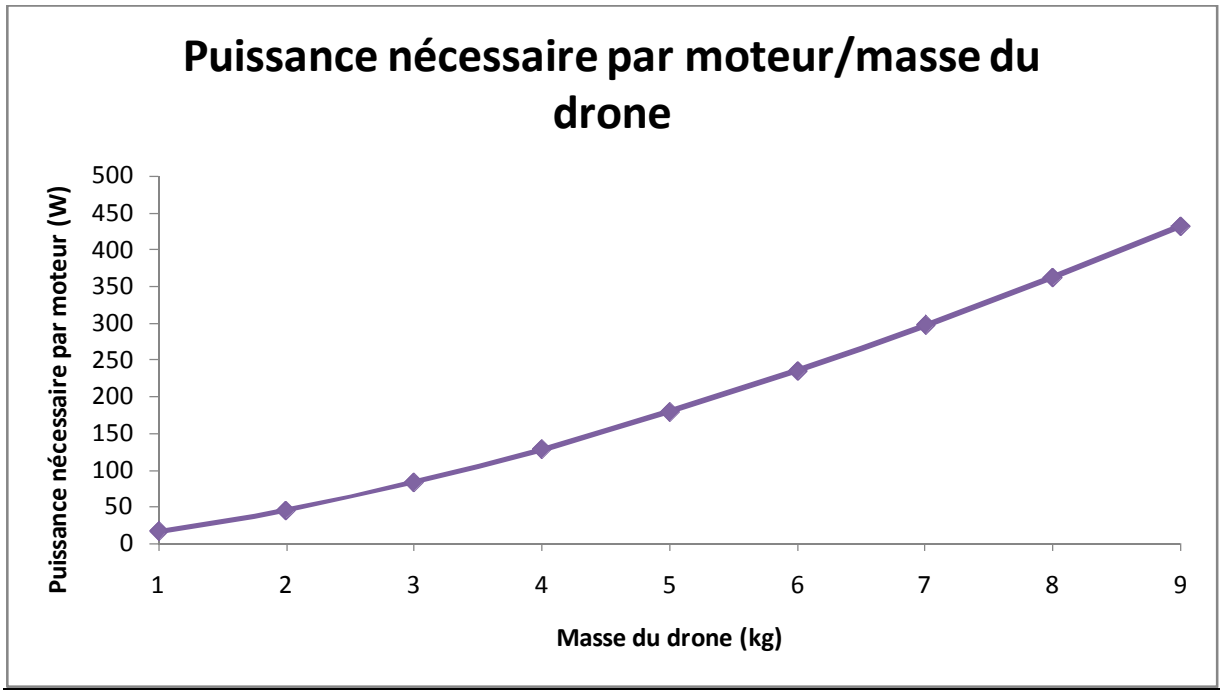
Le drone actuel nécessite beaucoup d'énergie pour se sustenter. La masse nécessaire à l'emport d'une telle quantité d'énergie impose le changement des variateurs, et probablement des moteurs (si l'on veut lever 2,8kg). Plutôt que d'emprunter ce cercle vicieux, regardons ce que l'on peut optimiser dans la puissance induite. S'il semble difficile de modifier la densité de l'air, en revanche on doit pouvoir augmenter la surface du rotor.

C'est donc là que réside le principal problème du drone actuel : les hélices sont beaucoup trop petites. Si l'on veut vraiment obtenir un dimensionnement efficace, il faut utiliser les hélices les plus grandes possibles.

PROPOSITION D'ARCHITECTURE

Pour la nouvelle solution, nous décidons de conserver un engin de taille similaire au drone actuel. L'engin futur devra tenir dans un carré d'environ un mètre de côté (à une dizaine de centimètre près). Les hélices auront donc un diamètre de 50 cm environ. En considérant une zone morte de 4cm de diamètre, on peut imaginer une surface S totale de 0,78m². Une telle hélice est introuvable sur le marché, car il ne faut pas oublier que deux des hélices doivent être à pas inverse. La meilleure façon d'obtenir ce que nous cherchons semble être de concevoir cette hélice à l'Ensica. Un tel modèle à déjà été conçu pour le drone Casper. Il doit être possible au prix de quelques modifications de fabriquer une hélice similaire, à la taille qui nous convient.

Reprenons alors le calcul de la puissance induite, pour déterminer la puissance nécessaire en fonction de la masse. Nous utiliserons pour le calcul de la puissance une figure de mérite de 0,6 et une marge de 40%. Ainsi, le drone atteindra la puissance nécessaire au vol stationnaire lorsque les moteurs développeront 60% de leur puissance max. Ce qui laisse une marge confortable.

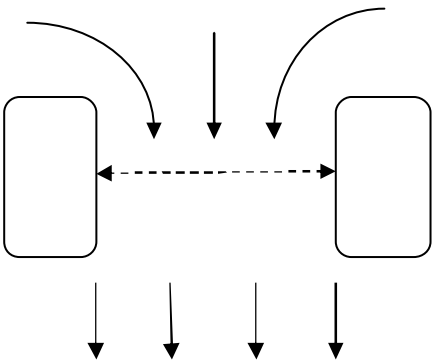


Nous constatons que les 1,6kgs sont soulevés avec 32W par moteur. Il semble cependant intéressant d'utiliser de telles hélices au transport d'une masse importante. Par exemple il faut 300W pour lever 7kg et 235 pour en soulever 6. Et nous disposons pour ces valeurs de 40% de marge !

Nous pouvons maintenant conclure sur cette architecture :

Un grand drone implique de grandes hélices, donc implique une capacité d'emport importante.

Avec 6 kg et 4 moteurs de 300 W (le moteur de Casper, aux très grandes hélices, fait 450 W et lève 2kg), on peut voir venir. En effet, le drone actuel fait environ 1,2 kg sans ses batteries. La masse restante peut être largement utilisée pour placer des batteries de très forte capacité, capables d'alimenter ces moteurs. Elles ne devraient pas peser plus de deux kilos. Avec le reste, on peut améliorer la structure et garder une réserve confortable en charge utile pour implémenter des caméras, des transmetteurs…



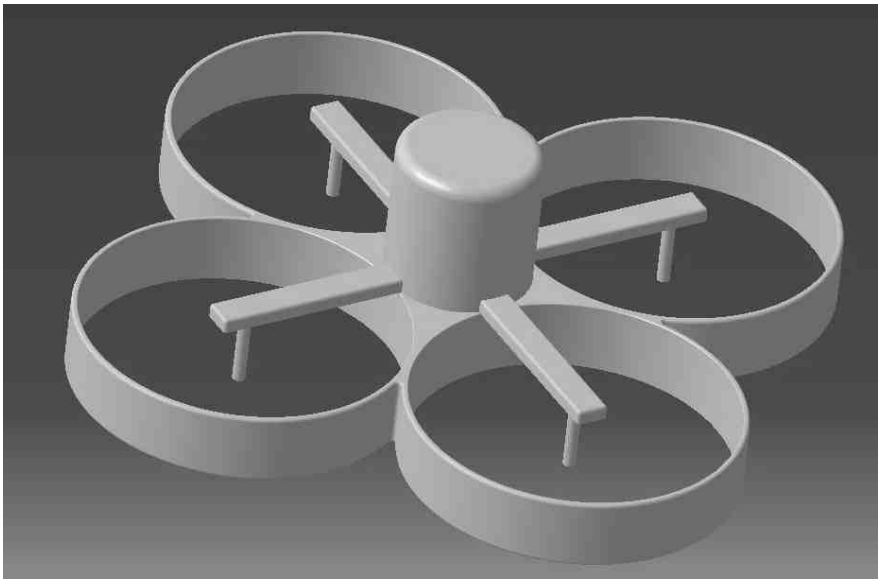
Principe du fenestron

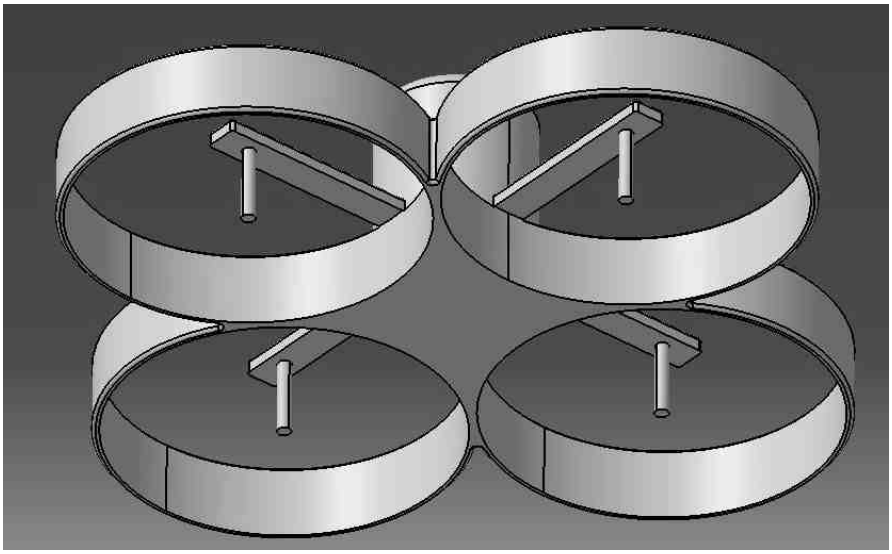
Par exemple, la masse disponible permet de caréner les hélices. Ce carénage protège les personnes, peut servir de structure d'atterrissage et peut jouer un rôle de fenestron, ce qui améliore de 1,4 la surface utile, et apporte un gain de portance appréciable !

Le fenestron permet d'optimiser la direction du flux d'air en sortie de rotor. La surface de rotor nécessaire pour obtenir la même portance qu'avec un rotor classique est divisée par √(2). L'ajout de matière nécessaire à sa réalisation peut donc être compensé par un gain de portance substantiel.

Toujours dans l'optique d'améliorer le flux, il semble plus intéressant d'utiliser des moteurs en configuration « push ». En effet, les hélices actuelles fonctionnent en « pull », elles tirent le drone, et le flux généré par l'hélice est détérioré dans le champ de la nacelle du moteur. Sur une voilure tournante, le flux situé dans la partie amont des hélices n'a pratiquement pas d'importance, et il n'y a presque aucun mouvement d'air dans cette partie. En revanche, à l'aval des hélices, le flux est concentré dans un mince pinceau sous le rotor, et la présence d'un obstacle en dégrade l'écoulement. Les hélices devraient donc être placées sous les moteurs, et non dessus.

Comme nous cherchons à augmenter au maximum la taille des hélices, le caisson central ne peut plus être placé dans le même plan. Il doit donc être placé sous les hélices, comme sur un hélicoptère (ce qui dégrade le flux), et jouer ainsi un rôle de balancier stabilisateur. On peut aussi le placer au dessus du plan des hélices. De cette manière, le flux d'air n'est pas perturbé, mais la position horizontale devient une position d'équilibre instable. Compte-tenu des bonnes performances du système de stabilisation, cela ne semble pas être un inconvénient. Cette position favorise de plus la réactivité du drone.





Proposition d'architecture drone quadrirotor

A contrario, si l'objectif visé est le gain de masse, le chemin à parcourir est exactement l'inverse : la taille de l'engin doit être réduite, de manière à maximiser la taille des hélices relativement à la structure. Cependant, le gain de masse par réduction de la structure serait faible, puisqu'il serait principalement dû à une réduction de la longueur des barres de carbone. Il ne faut pas oublier qu'il faudrait toujours utiliser de nouveaux variateurs, et de nouvelles batteries plus lourds car plus puissantes. Compte tenu de l'état actuel des technologies de batteries, la meilleure solution semble donc de construire un drone de grandes dimensions, sur lequel on pourrait emporter des batteries conséquentes.

Mécaniquement, la solution proposée requiert donc des hélices adaptées, qui possèderont probablement un bras souple, pour éviter la rupture de pâles due aux moments de flexion. Il s'agit donc d'une technologie autrement plus compliquée que les simples hélices plastiques actuelles.

Le reste de la structure peut être construit en carbone, car il offre un très bon rapport rigidité/poids, et est presque insensible à la fatigue. Ainsi la solution qui consiste à relier les moteurs au bâti par deux barres de carbone devrait être conservée. De même, le carénage des hélices peut être en carbone, voire en fibre de verre.

En ce qui concerne l'avionique, la carte MicroPilot offre un bon rapport puissance/poids. Son principal défaut réside dans son hermétisme. Ainsi l'erreur statique est due principalement à un différentiel de vitesse des moteurs pour la même commande. Il n'est pas possible avec MicroPilot de rajouter une entrée, comme un capteur relié à une roue codeuse sur l'arbre moteur, pour mesurer cette vitesse. On peut éventuellement imaginer une carte fille qui intercepte les commandes MicroPilot, mesure la vitesse des moteurs et l'asservit à la commande. Cependant, cette carte doit travailler sur des signaux de type PWM, et être suffisamment rapide. La complexité est donc au rendez-vous.

Une autre voie se dégage, celle d'une carte à microprocesseur. Le DMI travaille sur l'adaptation d'une machine virtuelle java pour processeur embarqué. Une telle solution offre la facilité de codage en java, l'ouverture d'un système non propriétaire et la puissance d'un processeur moderne. L'implémentation de cette carte est le principal obstacle. Il faut pouvoir lui adjoindre une carte d'entrée-sortie, pour traiter les signaux d'attitude, de vitesse des moteurs, et de commande des moteurs. Il lui faut également une carte d'alimentation.

CONCLUSION

Nous constatons que les solutions ne manquent pas. A l'heure actuelle, le système de commande est performant, et adaptable. La priorité semble donc de construire une nouvelle structure. L'actuelle n'a en effet pas subi de calculs de dimensionnement, et elle s'avère sous-motorisée. La construction d'un support solide et capable de multiples évolutions est donc la prochaine étape du projet. Il suffira ensuite de lui adjoindre la commande à base de carte MicroPilot et de reproduire nos protocoles de stabilisation pour disposer d'une machine opérationnelle. Le maître mot de cette solution nouvelle est l'emploi d'hélices plus grandes. En sus de tous les avantage cités précédemment, de grandes pâles feraient également profiter de l'effet de sol (existant pour une hauteur inférieur à la longueur des hélices).

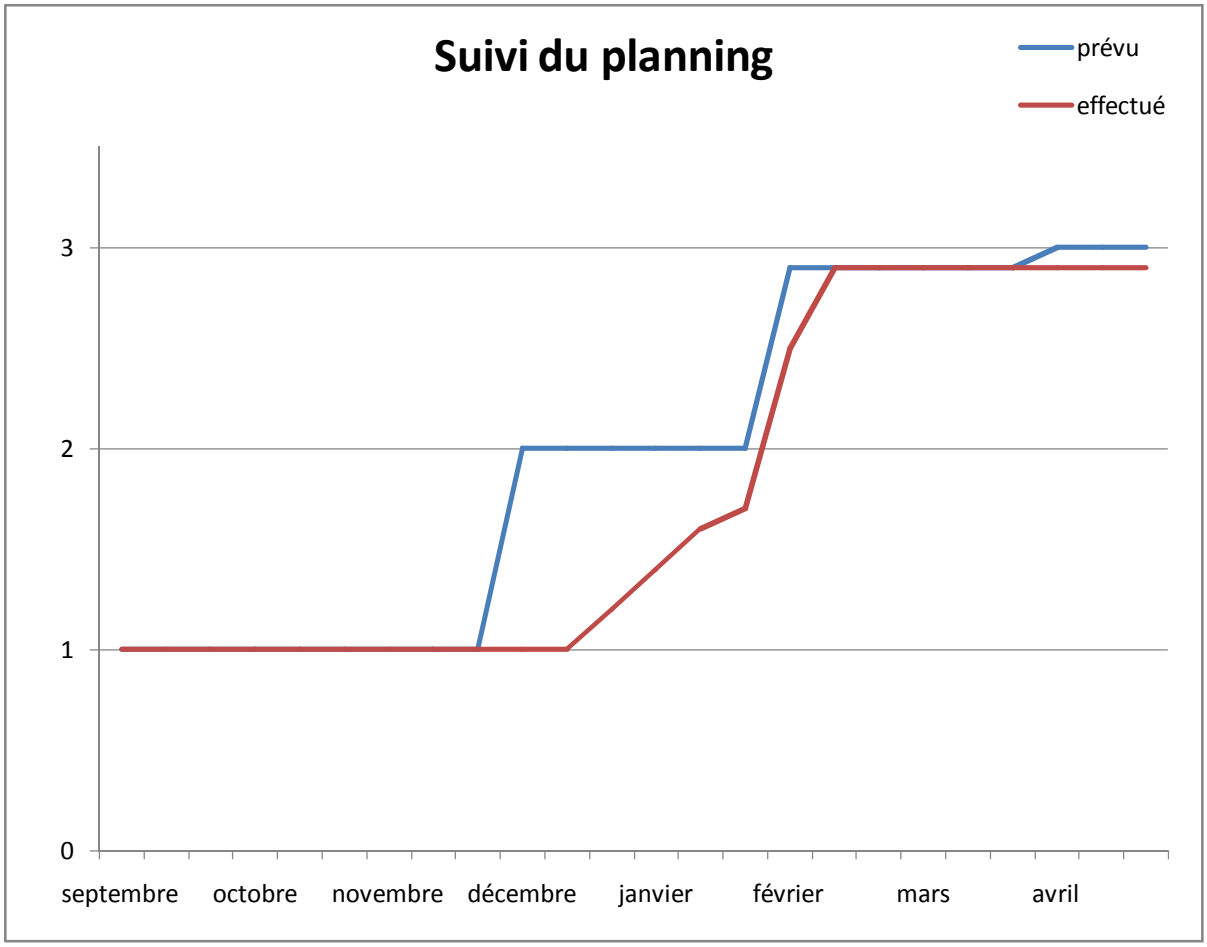
RETROSPECTIVE DU PROJET

PLANNING

Le planning initial comportait 3 étapes :

1. Prise en main de la carte Micropilot. Nous devions être capables à la fin de cette phase de programmer un code simple. Échéance prévue : fin novembre.
2. Fabrication d'un support de test et modification des éléments du drone en vue des tests. Nous devions disposer à la fin de cette phase d'un support permettant d'effectuer facilement les tests de stabilisation. Les branchements du drone devaient par ailleurs être simples. Pour rappel, le drone ne disposait dans son état initial d'aucune facilité de branchement particulière relative à MicroPilot. Échéance prévue : fin janvier.
3. Stabilisation du drone. Le drone devait, à la fin de cette phase, pouvoir effectuer un vol stabilisé télécommandé. Échéance prévue : avril.

Ce planning a été partiellement respecté. Le principal obstacle a été la difficulté de prise en main de MicroPilot. En effet, le bon fonctionnement de la carte n'est possible que lorsqu'aucune erreur n'apparaît. La majorité des erreurs possibles ne concernent cependant pas un drone quadrirotor, qui plus est alimenté par un moyen externe. Compte tenu du manque de documentation, l'élimination progressive de tous les problèmes, et la compréhension du fonctionnement ont été particulièrement laborieux. Pour ne pas prendre trop de retard, nous avons anticipé sur la suite du planning en fabriquant le support de test en parallèle aux essais de programmation. Nous avons ainsi pu rattraper notre retard début février.



Le deuxième problème majeur est survenu sur la fin de nos essais de vol. Le drone n'a en effet pas pu décoller. Cette défaillance est due à un sous-dimensionnement initial des moteurs. Nous n'avons pas remis en cause les choix initiaux, et considéré que le drone pouvait voler. Au moment où cet incident est survenu, il était trop tard pour envisager de commander ou fabriquer les pièces nécessaires au vol. Il n'y a donc pas eu d'essais en vol, ce qui explique le retard planning final.

IMPRÉVUS, PROBLÈMES RENCONTRÉS

Comme expliqué précédemment, deux problèmes majeurs sont survenus.

Le premier problème venait de la difficulté de prise en main de MicroPilot, que nous avions sous-estimée. À force de persévérance, et avec l'aide de Laurent Alloza, nous avons pu surmonter cette difficulté. Les conclusions à en tirer sont disséminées tout au long de ce rapport, elles concernent la mise en œuvre de MicroPilot.

Le deuxième problème est survenu sur la fin du projet, le drone ne pouvant se soulever.

Croyant au départ que ce problème était uniquement dû à des hélices inadaptées, nous avons envisagé plusieurs solutions qui n'ont finalement pas pu être mises en œuvre :

- Superposition de deux hélices par moteur. Nous disposons de chaque hélice en double, cette modification aurait permis d'améliorer la sustentation. Le problème venait de la longueur de l'axe de support hélice : elle n'était pas suffisante pour superposer deux hélices. Ce type de pièce n'existe pas pour des longueurs supérieures, et sa fabrication est complexe (filetage, usinage de cônes concentriques, grande précision requise…). Compte tenu de la surcharge de l'atelier à ce moment, nous avons dû abandonner.
- Augmentation de la vitesse de rotation des pâles. Via un réducteur, il est possible d'accélérer les pâles pour un même régime moteur. Le problème réside dans l'implémentation de ce réducteur. Il nécessite un axe de plus pour déporter l'axe des hélices de l'axe du moteur. Comme les hélices sont fixées dessus, cet axe doit supporter toutes les contraintes de sustentation. Il n'y a actuellement pas la place sur les nacelles moteur pour fixer un tel axe.

Ces solutions ne sont valables que si les moteurs sont suffisamment puissants. Le calcul de dimensionnement démontre que ça n'est pas le cas. Pour faire voler le drone, il faut donc remplacer les moteurs, les variateurs, les batteries, et proposer une modification sur l'hélice. Enfin, les coefficients de stabilisation doivent être réajustés. Nous n'avions pas le temps de procéder à de telles modifications, et la quantité de changements remet en cause l'architecture initiale. Plutôt que de proposer une telle réparation, pour n'obtenir qu'un engin aux performances limitées, nous préférons livrer des pistes quant à une nouvelle architecture de quadrirotor.

Nous considérons que l'architecture du drone doit être repensée, en ayant toujours en tête les performances attendues. Ces performances doivent être prouvées dès le début du projet, par des calculs de dimensionnement. Les principes de calculs sont livrés ici, et disponibles dans le cours d'hélicoptère de troisième année, en accès libre au CDI.

ATTEINTE DES OBJECTIFS, CONCLUSION

Le drone a obtenu de bonnes performances de stabilisation sur le banc de test, qui n'est certes pas représentatif de la réalité, mais permet d'être optimiste pour un essai en vol. Ces tests ont permis de stabiliser le drone sur ses deux axes longitudinaux. Le lacet ne peut être stabilisé qu'en vol. Les essais au sol ont montré qu'il existe naturellement un mouvement de lacet, ce qui n'est pas surprenant pour un hélicoptère. Ce mouvement est facilement corrigé avec la télécommande, malgré des coefficients de stabilisation non optimisés.

On peut donc conclure qu'à l'issue de ce travail, le drone est stabilisé et commandable en roulis et tangage. La stabilité semble assurée en lacet, mais la commandabilité reste à démontrer sur cet axe.

D'autre part, nous disposons d'un code de commande performant et souple, ainsi que d'un protocole de test rodé. Il sera donc facile, une fois le drone prêt à voler, d'ajuster la stabilisation.

Nous nous sommes attachés à travers ce rapport à expliquer le plus simplement possible les moyens que nous avons mis en œuvre. Ainsi, il sera facile pour l'équipe qui nous succédera de régler de nouveaux coefficients sur une plate-forme différente.

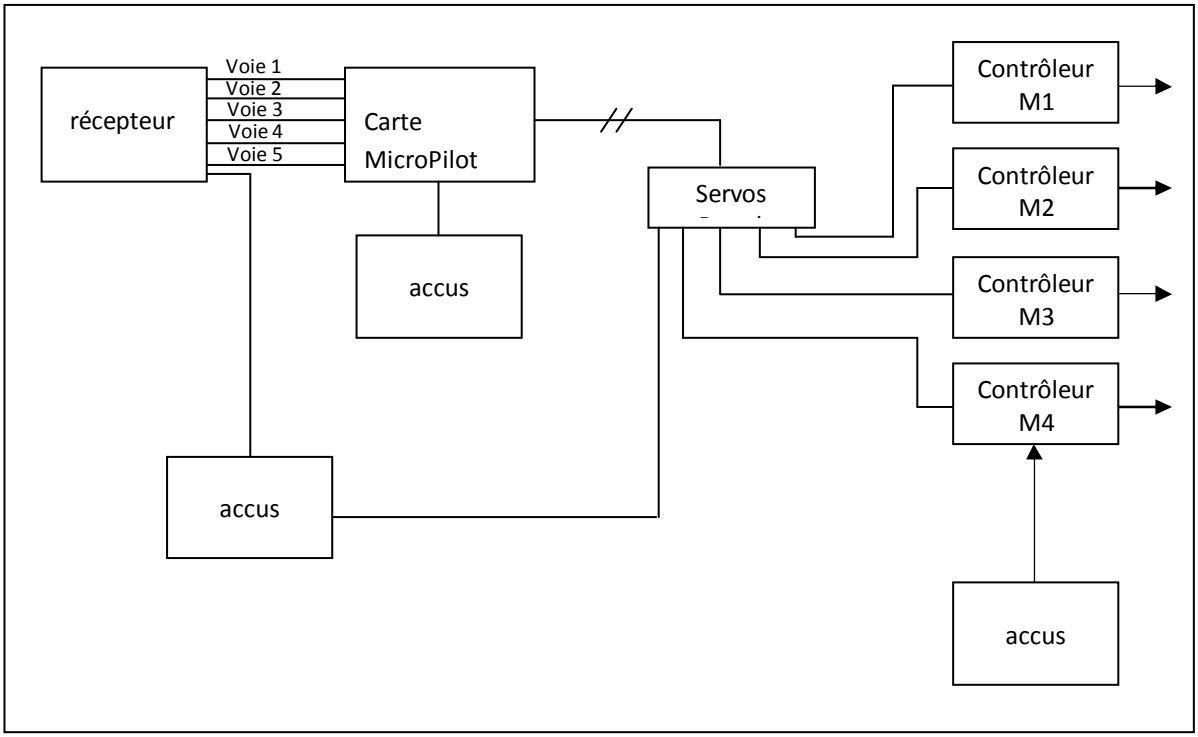
Nous pensons que la partie automatisme de ce projet est terminée. Un groupe peut reprendre ce travail et s'attacher à travailler la structure, par exemple en suivant les pistes détaillées dans la partie dimensionnement. Il n'aura alors qu'à suivre notre méthode de réglage des coefficients pour assurer la stabilité du drone.

ANNEXE

BRANCHEMENTS ET ALIMENTATION

BRANCHEMENTS DE LA CARTE MICROPILOT ET DE SES ELEMENTS ADDITIONNELS

La figure suivante est une représentation simplifiée des branchements à réaliser.



Branchements de la carte MicroPilot

BRANCHEMENTS RÉCEPTEUR / CARTE

Voie 1 : AILE (fil orange)

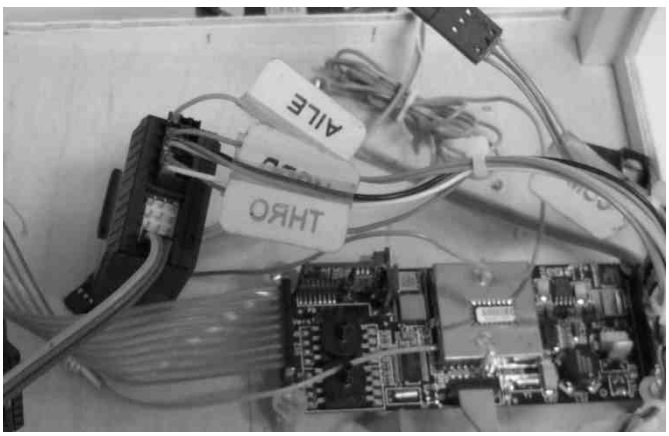
Voie 2 : ELEV (fil vert)

Voie 3 : THRO (fil gris, rouge, noir)

Voie 4 : RUDD (fil bleu)

Voie 5 : fil jaune

Pour chaque voie, le fil signal doit être branché vers la partie avant du récepteur. Pour la voie 3, le fil signal est le fil gris.



Branchement récepteur carte

Broche 5 : Moteur 2

Broche 6 : Moteur 3

Broche 7 : Moteur 4

Broche 8 : Moteur 1

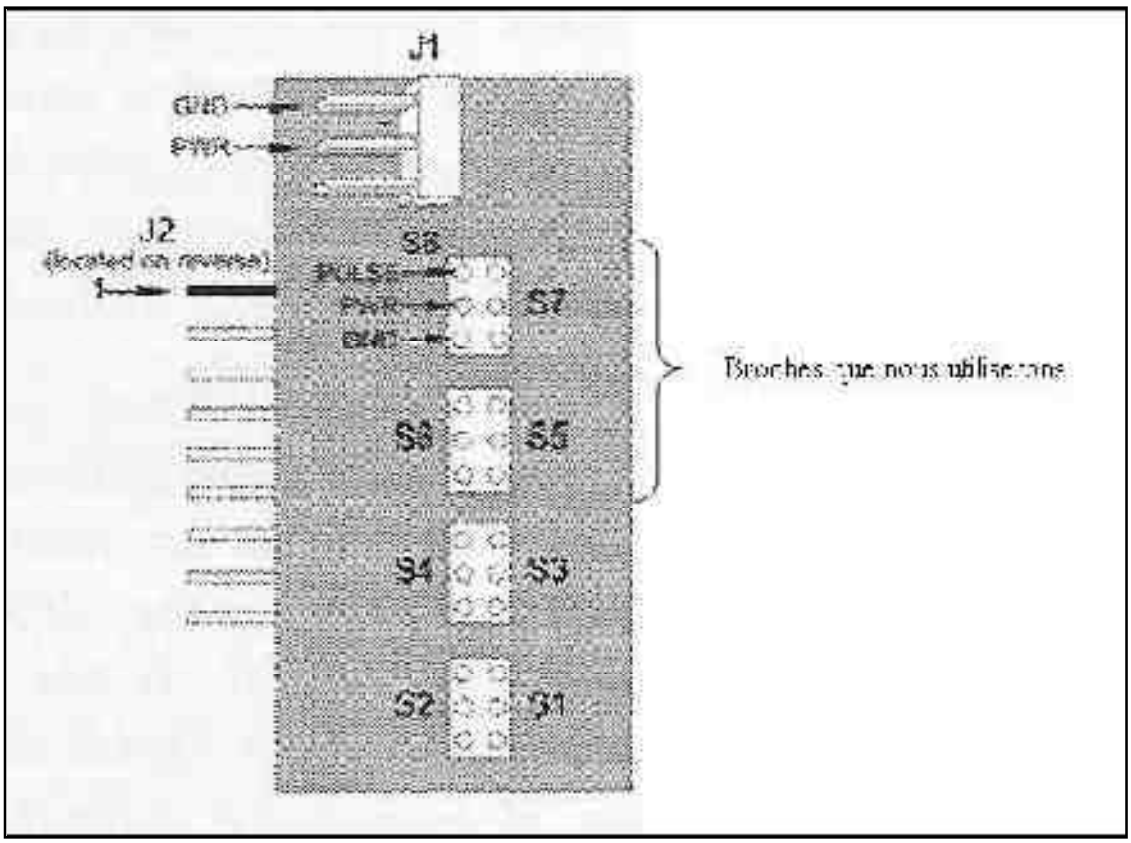


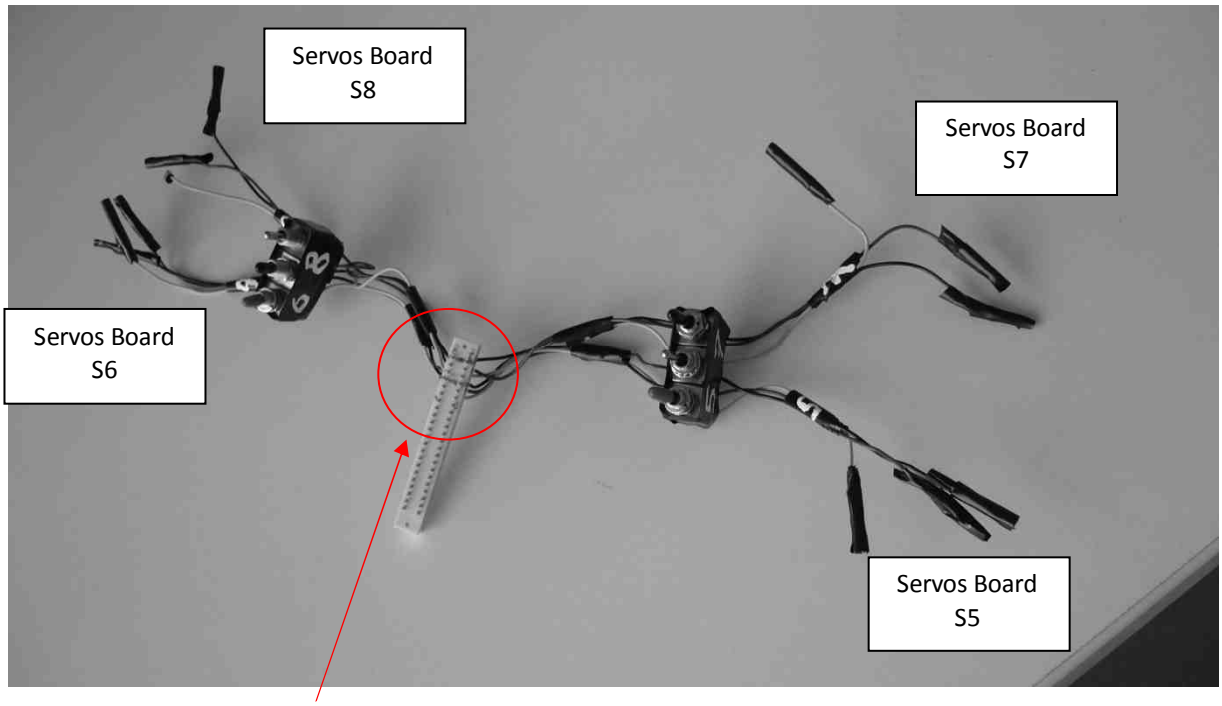
Schéma du bloc servos Board

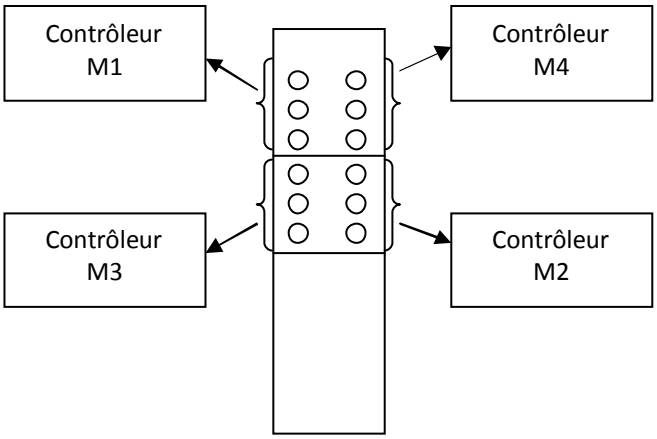
Les broches 1 à 4 ne sont pas utilisées dans notre cas car la carte envoie sur elles des signaux spécifiques correspondant aux réglages initiaux de la MP2028 pour le vol d'un avion. Ainsi par exemple, si l'AGL (altimètre par ultrasons) n'est pas branché, ce qui est notre cas, étant donné qu'il ne mous est pas utile dans le cas d'un hélicoptère, la carte signale cette erreur en envoyant des consignes périodiques sur les broches 1 à 4 (concrètement la carte fait battre les gouvernes de l'avion).

Attention : Il est très important de faire attention au sens de branchement sur le servos board, car celui-ci n'étant pas muni de détrompeurs, une erreur entrainerait la destruction du contrôleur concerné.

Branchement avec interrupteurs

Pour faciliter la mise en marche du drone nous avons placé des interrupteurs entre le servos board et les contrôleurs. En effet, les contrôleurs ne doivent recevoir aucune information avant que la carte MicroPilot soit complètement initialisée. Les interrupteurs nous ont permis de ne pas avoir à brancher et débrancher les contrôleurs à chaque nouvel essai.

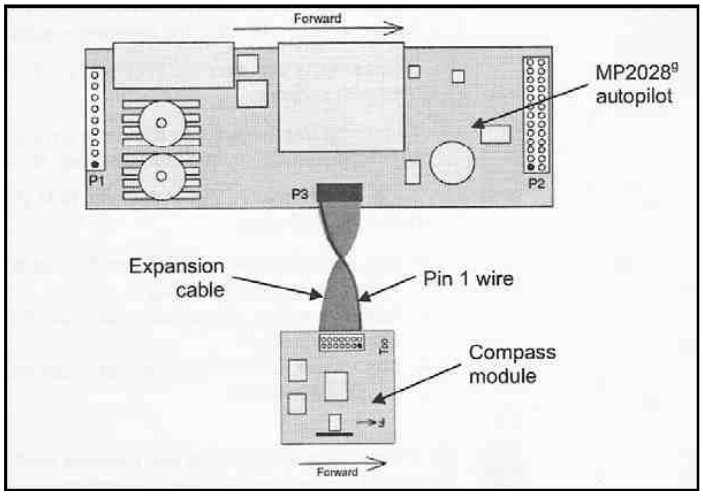




Branchement avec interrupteurs

BRANCHEMENT CARTE / COMPAS MAGNETIQUE

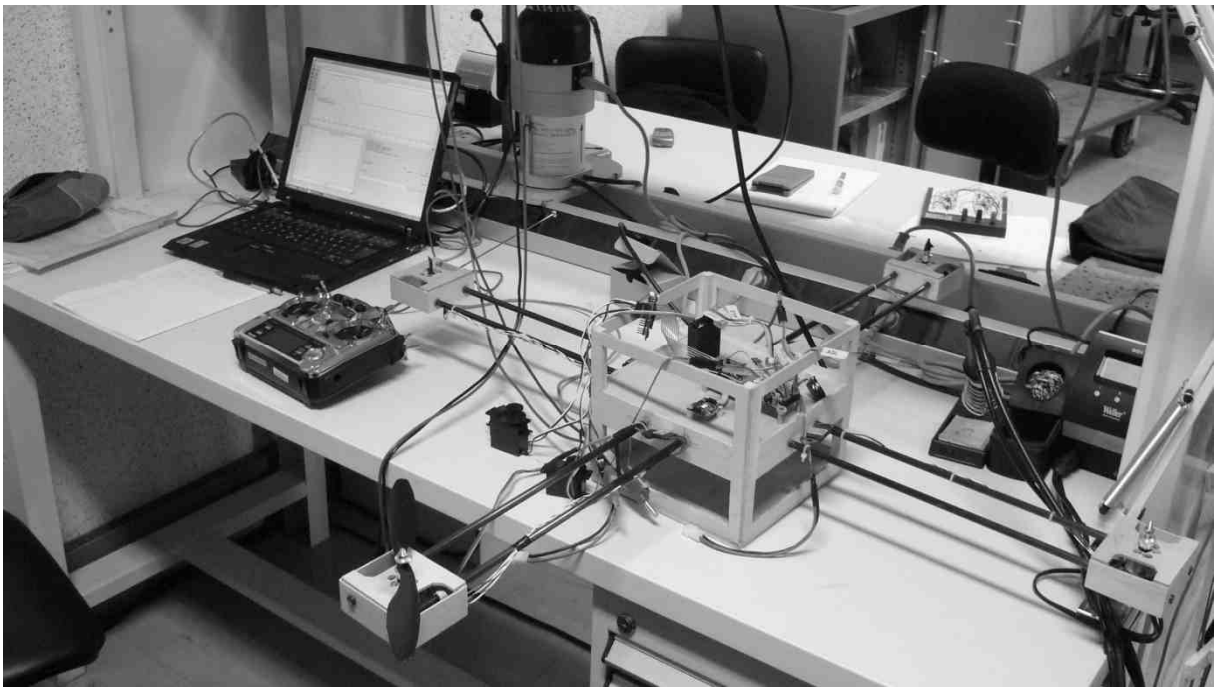
Le branchement doit se faire en accord avec le schéma suivant extrait de la documentation MicroPilot MP2028.



Branchement de la carte MicroPilot et du compas magnétique

BRANCHEMENT DE LA LIAISON CARTE / ORDINATEUR

Ce branchement se fait simplement par l'intermédiaire du fil COM de la carte qu'il faut relier à un câble série (avec éventuellement l'adaptation d'un câble série/USB) ou à un radio modem embarqué pour les communications en vol.



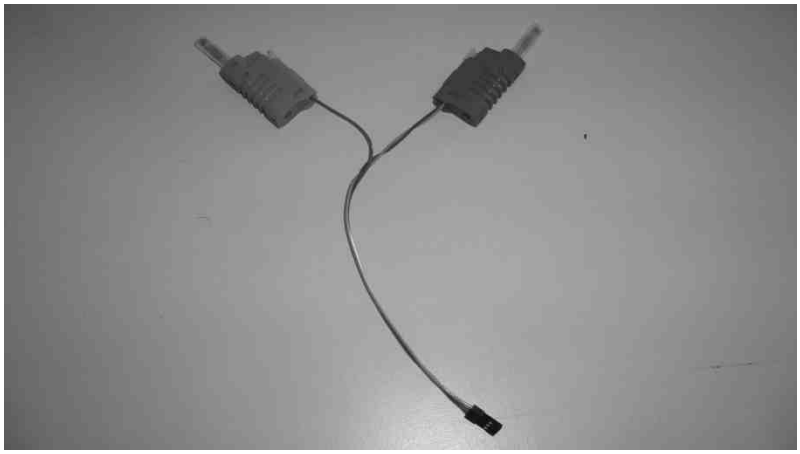
Drone relié au PC portable

ALIMENTATION DE LA CARTE ET DE SES ELEMENTS ASSOCIÉS

ALIMENTATION DE LA CARTE MICROPILOT

La carte doit être alimentée par une tension supérieure à 4,2V. L'alimentation se fait par le câble PWR, en faisant une nouvelle fois très attention au sens de branchement.

Pour les essais au sol, nous avons utilisé une alimentation réglée sur 5 V environ. Pour relier le câble PWR à l'alimentation nous avons utilisé un câble particulier. (Voir photo ci-dessous). Dans la fiche positive, on a inséré les fils rouge et orange, dans la négative, on a placé le fil marron.



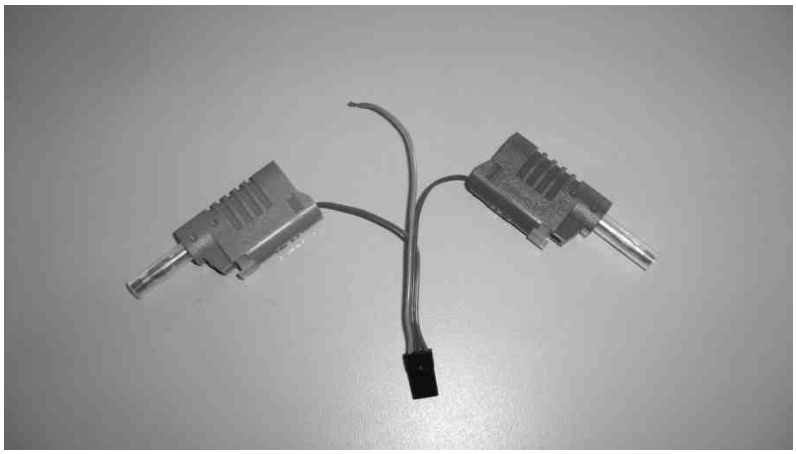
Câble d'alimentation de la carte MicroPilot

Pour les essais en vol, nous avons utilisé une batterie classique 4,8V, 800 mAh.

ALIMENTATION DU SERVOS BOARD ET DU RÉCEPTEUR

Le servos Board est alimenté en 4,8 V, ce qui peut être réalisé en parallèle avec l'alimentation du récepteur lui aussi en 4,8 V. Cela permet un gain de masse sans pour autant réduire l'autonomie de l'accumulateur, la puissance consommée par ces deux éléments étant relativement faible. Cette double alimentation est réalisée à l'aide d'un câble en câble en Y, possédant une entrée (accumulateur) et deux sorties (récepteur, servos board). L'alimentation du récepteur se fait par la dernière voie du récepteur.

Pour les essais au sol nous avons utilisé une alimentation réglée sur une tension de 4,8 V. Pour relier le câble en Y à l'alimentation nous avons utilisé un câble particulier (figure suivante). Ici, contrairement à l'alimentation de la carte, dans la fiche positive on n'a inséré que le fil rouge. Dans la fiche négative, on insère le fil marron.



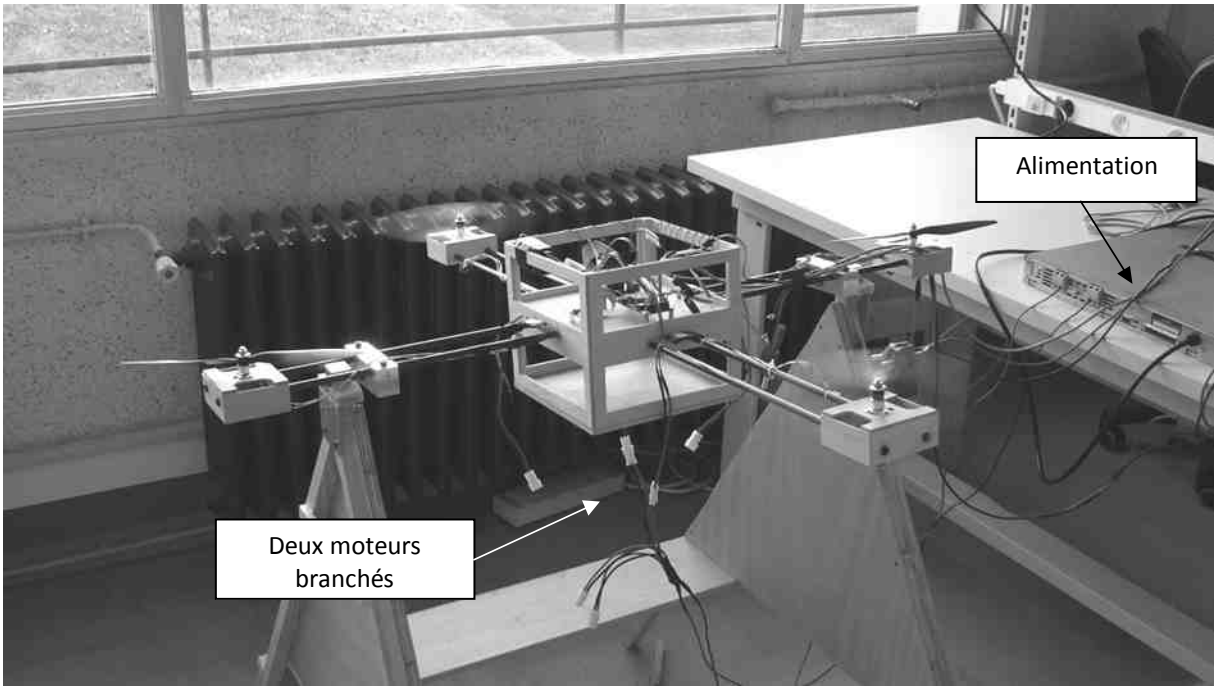
Câble d'alimentation du récepteur et du servos Board

Pour les essais en vol nous avons utilisé une batterie identique à celle que nous utilisons pour la carte MicroPilot.

ALIMENTATION DES MOTEURS

L'alimentation des moteurs de fait par l'intermédiaire des contrôleurs (ceux-ci sont reliés aux moteurs).

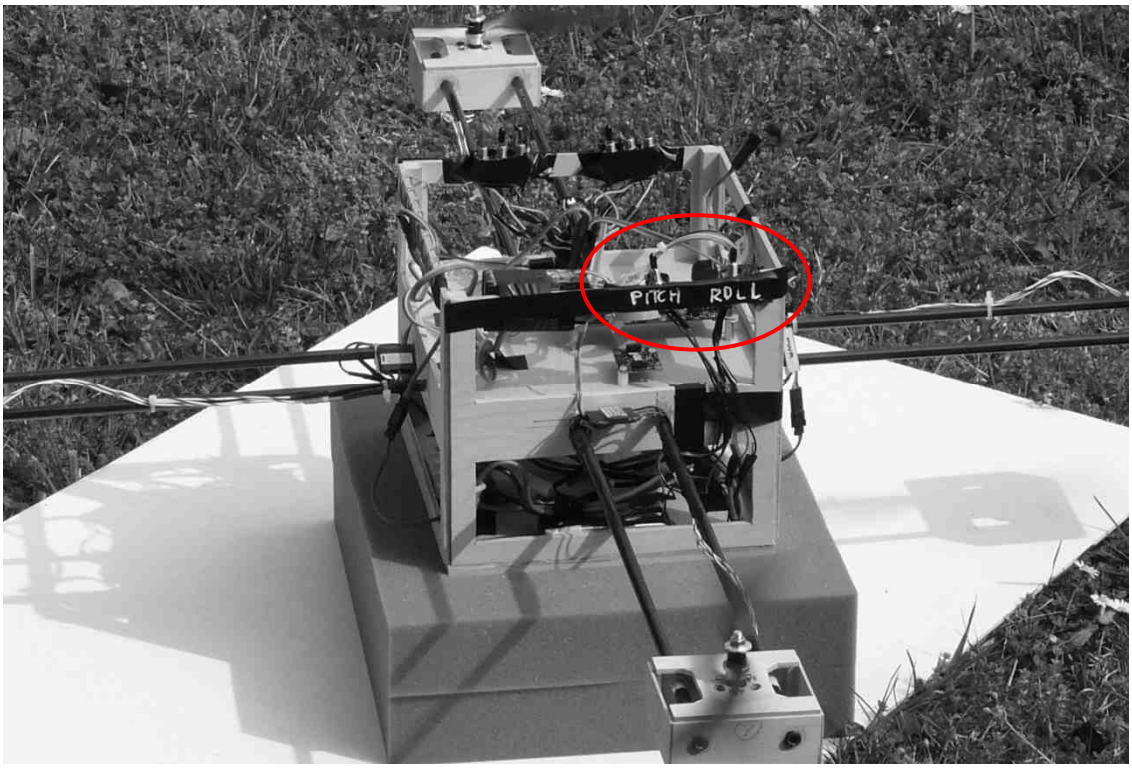
Pour les essais au sol, nous avons utilisé une alimentation puissante réglée sur 10 V, ceci grâce à un câble à une entrée (alimentation) et quatre sorties (contrôleurs). Cependant, lors des essais nous ne faisions généralement fonctionner que deux moteurs à la fois.



Branchement des moteurs lors des essais sur le support

Pour les essais en vol, nous avons utilisé deux batteries Li-Po à deux éléments. Chaque batterie alimente deux moteurs.

A cause des contrôleurs, les moteurs doivent être mis sous tension à un instant bien précis. C'est pour cela que nous avons installé sur le drone deux interrupteurs permettant de couper le courant entre les contrôleurs et les batteries. Un des interrupteurs permet de mettre sous tension les moteurs sur l'axe de tangage (pitch) et l'autre ceux sur l'axe de roulis (roll).



Interrupteurs : alimentation des moteurs

UTILISATION DE LA CARTE MICROPILOT ET DES LOGICIELS ASSOCIÉS

Trois logiciels sont utilisés pour écrire un code dans la carte Micropilot:

- Horizon, qui permet de lire et écrire des variables dans la mémoire
- Xtender, qui permet de charger un programme dans la carte
- LogViewer, qui permet de lire les données enregistrées par la carte
- Enfin, on peut également utiliser HyperTerminal pour remplacer Horizon

L'utilisation d'HyperTerminal est le meilleur moyen de comprendre le fonctionnement de la carte. Ce moyen sera donc décrit en premier.

UTILISATION DE HYPERTERMINAL

La carte communique avec l'ordinateur via un port série. Il est possible, via un logiciel tel que HyperTerminal ou RealTerm, de lire et d'écrire des données sur la carte.

Ce moyen permettra d'analyser l'état des variables systèmes. Il s'agit des variables décrites dans l'annexe du manuel Micropilot.

Pour ce faire, il faut configurer le logiciel pour lire les données du port série à un débit de 9600 bauds. Sur l'ordinateur IBM « microdrone », le port série étant absent, on branchera l'adaptateur sur le port USB du haut. Il s'agit du port COM1 pour lequel RealTerm est déjà configuré (raccourci sur le bureau).

Il suffit ensuite de brancher la carte et d'attendre la fin de son initialisation qui dure environ 20 secondes. Cette phase peut être plus longue si le GPS n'est pas désactivé et que le drone est situé dans un bâtiment. La récupération des éphémérides est alors très longue. Pour éviter ce désagrément, on peut exécuter un cold start du GPS en tapant sur « ffff ».

La configuration du système se fait via le menu, accessible en tapant « qqqq ». Une fois dans le menu, différentes commandes permettent d'agir sur le système, par exemple :

h	Lister les commandes disponibles
a	Ajuster manuellement un servo. Il faut ensuite taper le numéro du servo, puis taper – ou + pour l'ajuster
x	Lire un paramètre
p	Écrire un paramètre
l	Lister les paramètres

Les paramètres à lire ou écrire sont appelés par leur numéro, détaillés dans l'annexe du manuel Micropilot. Voici une liste des numéros utiles :

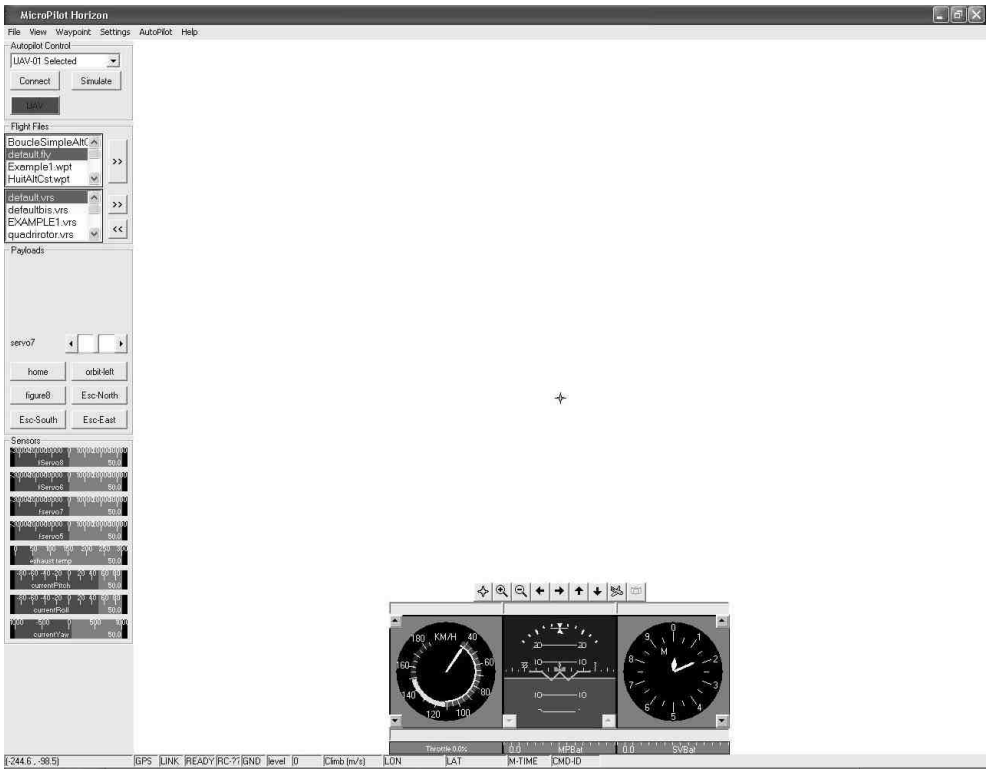
1231	fServo5 : commande du moteur 2
1232	fServo6 : commande du moteur 3
1233	fServo7 : commande du moteur 4
1234	fServo8 : commande du moteur 1
1153	Commande de roulis de la télécommande
1152	Commande de tangage de la télécommande
1154	Commande de lacet de la télécommande
1155	Commande de gaz de la télécommande
5451	Spécifie si le UserPid 1 est actif (=3 si actif)
117	Activer ou désactiver le capteur à ultrasons
157	Définit le niveau critique de la batterie de la carte Micropilot

155	Définit le niveau critique de la batterie du ServoBoard
1057	Mesure de tangage
1060	Mesure de lacet
1059	Mesure de roulis
1403	Mesure de vitesse en tangage
1404	Mesure de vitesse en roulis
1405	Mesure de vitesse en lacet
116	Activer la télémétrie
880x	Variables laissées libres pour l'utilisateur. Se référer au travail d'Andreas Görmer pour plus de détails.

On peut ainsi configurer complètement le drone, voire le contrôler puisque l'on peut faire bouger les servos. On pourrait ainsi imaginer un programme simple en java de configuration du drone, qui enverrait ces données au port série. Ainsi, on peut utiliser les variables 880x pour régler les 8 coefficients du PDD². Cependant, cette interface possède un défaut majeur : le programme de la carte est stoppé lorsque l'on rentre dans le menu de configuration. Il faut donc pouvoir configurer, puis quitter le menu pour reprendre le fonctionnement du programme. Et c'est là que réside le problème : il n'y a pas de commande pour quitter le menu ! La seule solution est donc d'éteindre la carte et de la rallumer, ce qui est loin d'être élégant. C'est pourquoi nous nous sommes résignés à utiliser Horizon.

UTILISATION DE HORIZON

Horizon est un logiciel fournissant une interface agréable de configuration et de contrôle des états de MicroPilot. On n'accède cependant qu'aux paramètres disponibles dans l'interface. Le contrôle sur la carte est donc limité.



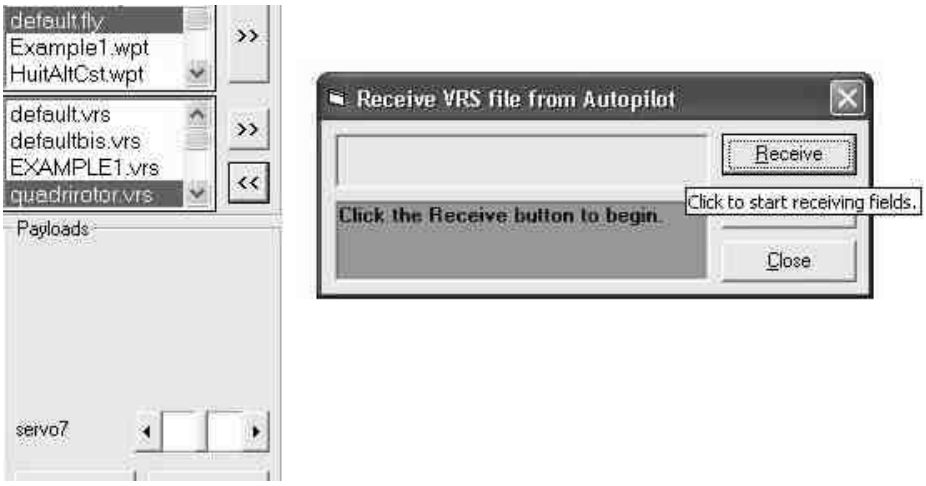
L'interface Horizon

On peut séparer les possibilités d'Horizon en deux catégories :

- La configuration du système, similaire à ce que l'on pourrait faire avec HyperTerminal.
- Le contrôle de paramètres au cours de l'exécution du programme.

CONFIGURATION DU SYSTÈME

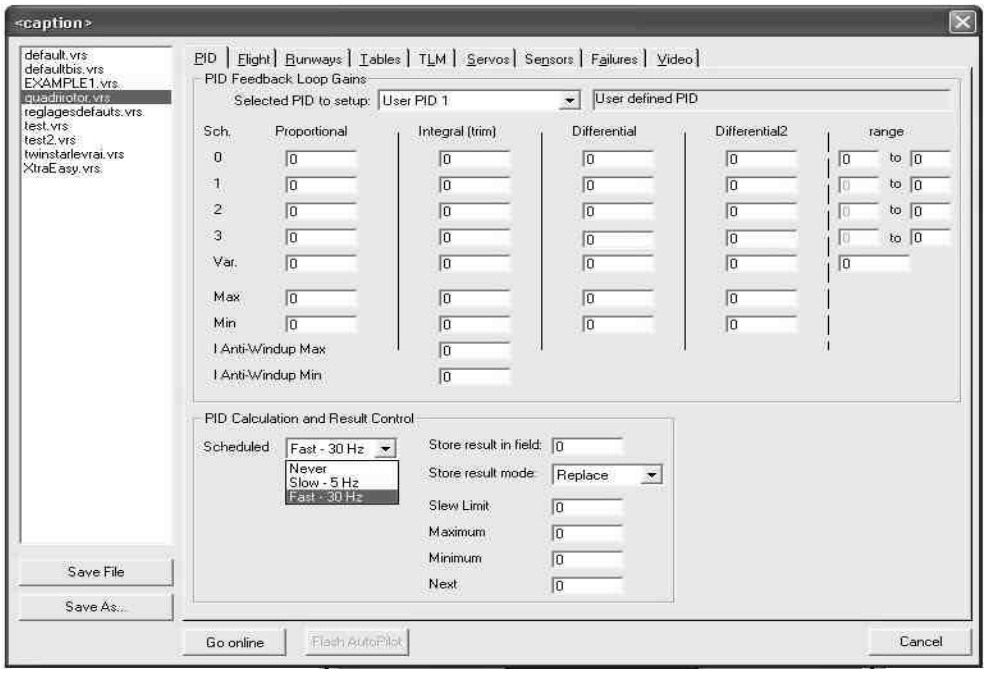
Les données de configuration sont sauvegardées dans des fichiers portant l'extension « .vrs ». La liste des fichiers de configuration est disponible dans la liste déroulante à gauche de l'interface. Un double clic sur quadrirotor.vrs ouvre par exemple la configuration sauvegardée du drone. Les deux flèches à droite de la liste permettent respectivement de charger les données sauvegardées sur le pc vers le drone, ou de sauvegarder les données du drone sur le pc.



Sauvegarde des paramètres du drone

Ces fichiers permettent donc d'enregistrer des configurations différentes.

Jetons un œil sur le contenu d'un fichier vrs. Lorsqu'on double clique sur quadrirotor.vrs, on obtient cette fenêtre :



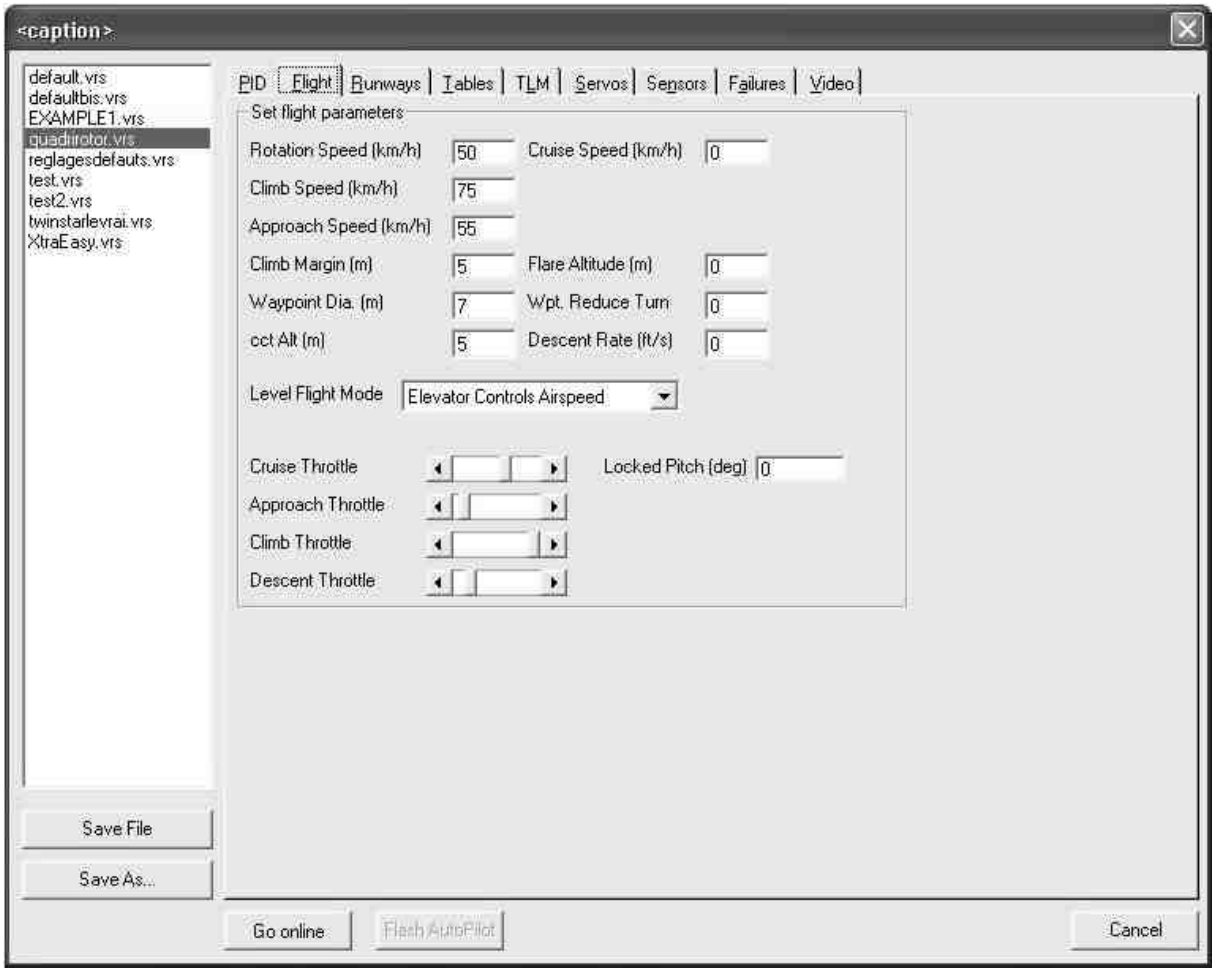
Cet écran regroupe la liste des paramètres à modifier, regroupés en onglets. Le premier onglet, « PID » contient notamment les paramètres de configuration des user PID. On peut ainsi définir le User PID 1 à 30Hz (cf. écran). Ainsi le code contenu dans User PID 1 sera exécuté 30 fois par seconde. L'action effectuée est la même que mettre à 3 la variable 5451 sous HyperTerminal. C'est la seule chose qui nous intéresse dans cet onglet. On s'assurera également que les autres User PID sont désactivés.

L'onglet Sensors permettra de désactiver le GPS, la sonde à ultrasons (appelée AGL), et de définir les niveaux critiques des batteries MicroPilot et servoboard. Lorsque les batteries alimentant ces éléments auront atteint un voltage inférieur à celui proposé, la carte générera un code d'erreur. On ne pourra alors plus exécuter le code utilisateur dans ces conditions. Lors des tests, on devra donc vérifier d'alimenter les éléments avec une tension supérieure à ces seuils. Il faut noter que les seuils sont exprimés en centivolts (400=4 volts).

L'onglet Flight est particulièrement intéressant. Nous remarquons en effet qu'il est impossible d'accéder sous Horizon aux variables 880x, qui sont les variables utilisateurs. Ces variables auraient été bien pratiques pour stocker les coefficients de réglage du correcteur. En revanche, l'onglet Flight regroupe un certain nombre de variables (Rotation Speed, Climb Speed…) qui ne nous sont d'aucune utilité. Nous allons donc les détourner de leur usage premier pour régler les coefficients du correcteur.

Nom sous Horizon	Nom dans le code	Fonction
Rotation Speed	K1roll	Correction proportionnelle roulis et tangage
Climb Speed	K2roll	Correction dérivée roulis et tangage
Approach Speed	K3roll	Correction dérivée deux fois roulis et tangage
Climb Margin	K1yaw	Correction proportionnelle lacet
Waypoint Dia.	K2yaw	Correction dérivée lacet
Cct Alt.	K3yaw	Correction dérivée deux fois lacet
Flare Altitude	K4roll	Correction intégrale roulis et tangage
Descent Rate	K4yaw	Correction intégrale lacet

Pour modifier ces coefficients et ainsi régler la stabilisation, il suffit de brancher le drone, d'ouvrir l'écran quadrirotor.vrs dans Horizon, puis dans l'onglet Flight de cliquer sur « Go Online » (le bouton devient vert si tout se passe bien). De cette manière, les données proposées dans les champs sont celles chargées dans le drone. Il faut alors modifier les champs correspondant aux coefficients correcteurs, puis de cliquer sur « Flash Autopilot » pour les charger dans le drone.



Écran "Flight" de quadrirotor.vrs

CONTRÔLE DES PARAMÈTRES

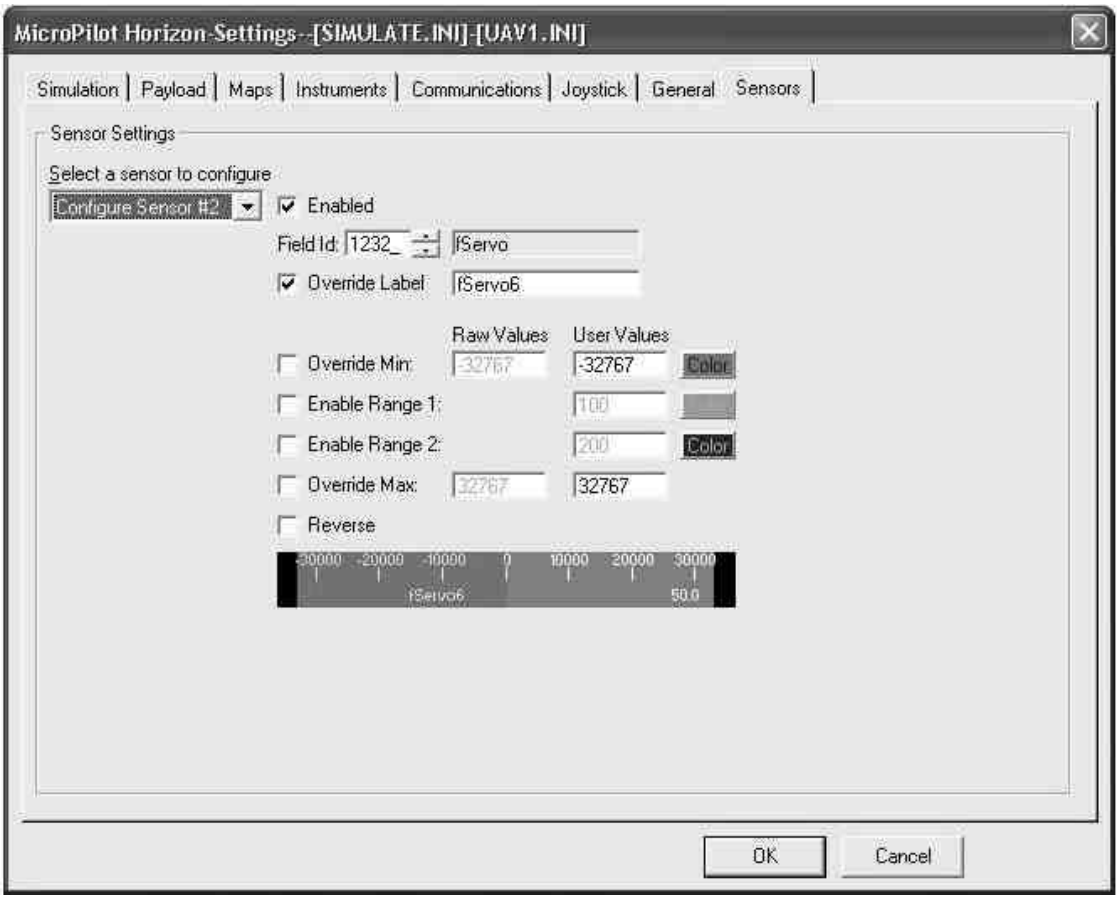
L'autre fonctionnalité d'Horizon est le contrôle de l'exécution du code. Pour cela, alimenter le drone suivant la procédure décrite dans l'annexe « branchements du drone », brancher le port COM à l'adaptateur série du PC. S'assurer que la carte MicroPilot est alimentée et lancer Horizon. Cliquer alors sur « Connect ».

La barre d'état d'Horizon contient diverses inscriptions qui sont vertes ou rouges suivant l'état du drone. Si tout est bien configuré, seule l'inscription « READY » doit être rouge, sinon, passer la souris sur l'inscription pour en savoir plus sur l'erreur. Il peut s'agir par exemple d'un niveau de batteries trop bas. Il suffit de reprendre la configuration de la carte à l'aide de la documentation MicroPilot et d'un peu de patience pour résoudre le problème.

L'inscription « READY » indique lorsque la carte est opérationnelle. Il faut compter environ deux minutes après l'alimentation de la carte pour que le système soit prêt. « READY » devient alors vert et MicroPilot commence à exécuter le code du User PID.

Ce n'est que lorsque MicroPilot est « READY » que l'on peut brancher les variateurs (basculer les 6 interrupteurs d'allumage des variateurs). Toute tentative de branchement avant cette phase conduit à une erreur des moteurs (série de bips des moteurs). Lorsque le système est branché, chaque moteur émet un bip. Il faut alors brancher l'alimentation des moteurs (basculer les deux interrupteurs d'alimentation). Les interrupteurs situés sur le drone sont fermés lorsqu'ils sont basculés vers l'extérieur du drone. Ils sont ouverts lorsqu'ils sont basculés vers l'intérieur. Il faut également vérifier que la télécommande est bien branchée avant que la carte ne s'initialise, sinon le code ne peut pas faire le centrage des commandes, et les moteurs ne démarreront pas.

On remarquera qu'à partir de cet instant, Horizon affiche les paramètres d'attitude dans l'horizon artificiel et met à jour les données des senseurs à gauche. Il est possible de contrôler ainsi le bon fonctionnement des commandes. Pour modifier les senseurs à afficher, aller dans « Settings > Edit Horizon Settings ». Choisir l'onglet « Sensors ». On peut ainsi configurer 8 capteurs. Le type de donnée à afficher est identifié par le numéro MicroPilot, par exemple 1234 pour le servo 8.



On remarquera qu'il n'est pas possible de vérifier à la fois l'état des commandes et de configurer les paramètres. Pour quitter l'état d'observation, il faut cliquer sur « TERMINATE ».

UTILISATION DU LOGVIEWER

LogViewer est un utilitaire qui permet de lire et tracer les données enregistrées au cours d'un vol. Son icône est sur le bureau, et les logs sont enregistrés dans le dossier log, également sur le bureau.

Tout d'abord, il faut s'assurer qu'aucun programme ne monopolise le port série (HyperTerminal, Horizon…). Puis dans LogViewer, faire « File > Read Log File from Autopilot ». Le fichier correspondant au vol précédent est chargé. Dans l'onglet « Log Data », il suffit alors de double cliquer sur les titres des données à afficher (l'indicatif « visible » apparaît alors) pour tracer l'évolution de ce paramètre dans « Log Plots ». On peut alors zoomer, puis imprimer ou imprimer en pdf. Les fichiers log sont sauvegardés dans le répertoire correspondant en fonction de l'heure et de la date, mais on peut changer leur titre pour les retrouver facilement.

UTILISATION DE XTENDER

Xtender permet d'écrire du code dans MicroPilot. Le code est écrit en C . Pour plus de facilité, nous avons crée deux répertoires, accessibles sur le bureau « Raccourci vers MicroPilot > XTENDER3 > examples ». Le répertoire quadrirotor contient le code utilisé pour la stabilisation. Le répertoire test contient un code modifiable pour expérimenter. Dans chaque répertoire, le code doit être appelé « test.c » ou « quadrirotor_userCode.c » (suivant leurs répertoires respectifs). Si l'on veut garder des copies du code, il est possible de faire des copiés collés, par exemple dans le répertoire « sauvegardes de test » dans « test ». On pourra ainsi sauvegarder différents codes, puis les replacer dans le répertoire racine en les renommant « test.c » pour les enregistrer sur la carte (la procédure est la même pour quadrirotor).

Une fois le code écrit, il faut le compiler, puis le charger dans le drone.

La compilation se fait en exécutant le fichier « make-usercode.bat ». Il faut alors prendre garde aux erreurs de compilation, signalés par « Warning ». Seules les erreurs de variable non utilisées ou relatives à l'utilisation de GetMPVarPointer sont autorisées. Les autres doivent être corrigées, et le code recompilé avant de continuer.

Il faut ensuite copier les fichiers de compilation dans le répertoire approprié, pour cela, exécuter « appendUser.bat » qui s'en charge tout seul.

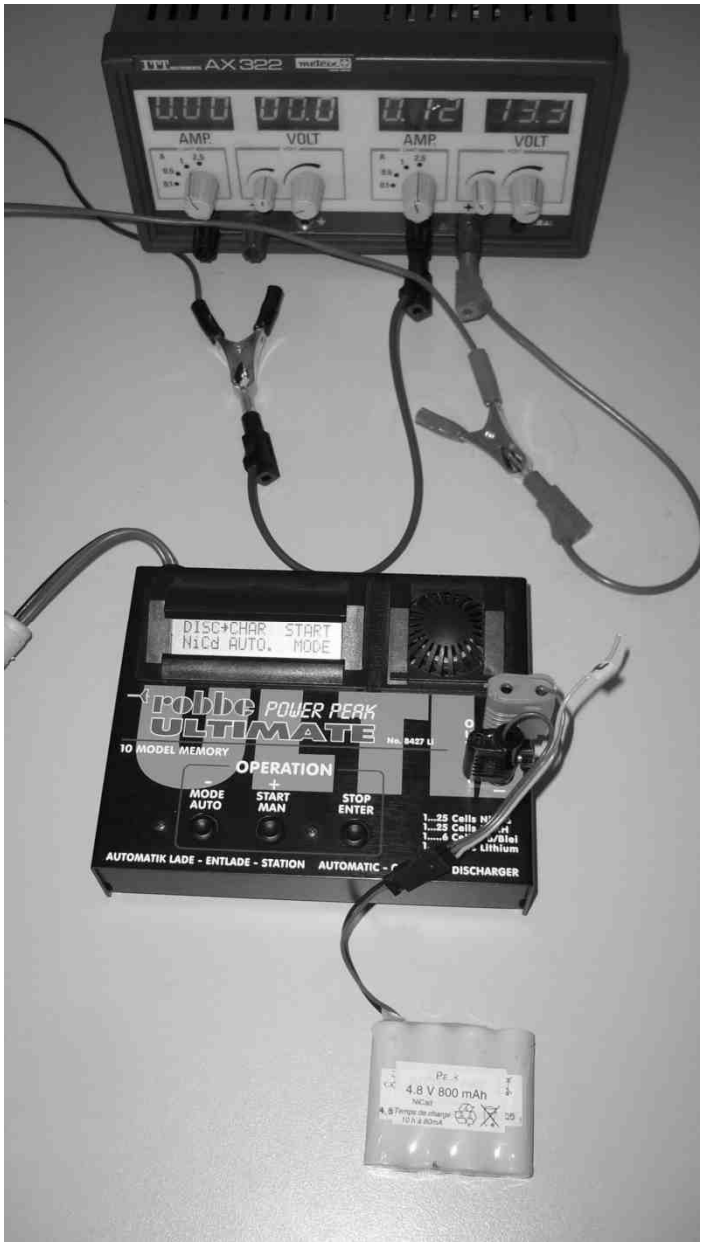
Se rendre ensuite dans le dossier « bin » de « Xtender3 », et vérifier la date de création du fichier « myflyanduser.bin ». Il s'agit du fichier compilé donc si tout s'est bien passé, il a dû être crée au moment de l'exécution de « appendUser.bat ». S'assurer alors que MicroPilot est éteint puis exécuter « flashld-myflyanduser.bat ». Allumer MicroPilot lorsque l'invite de commande le demande. Le programme est alors chargé dans le drone. L'opération prend environ trois minutes et se termine par un bip bip de confirmation. On peut alors éteindre MicroPilot et le rallumer pour lancer l'initialisation puis l'exécution du code.

PRÉPARATION DU DRONE, CHARGE DES ÉLÉMENTS

Pour fonctionner, les différents éléments du drone doivent être correctement alimentés. Il faut donc :

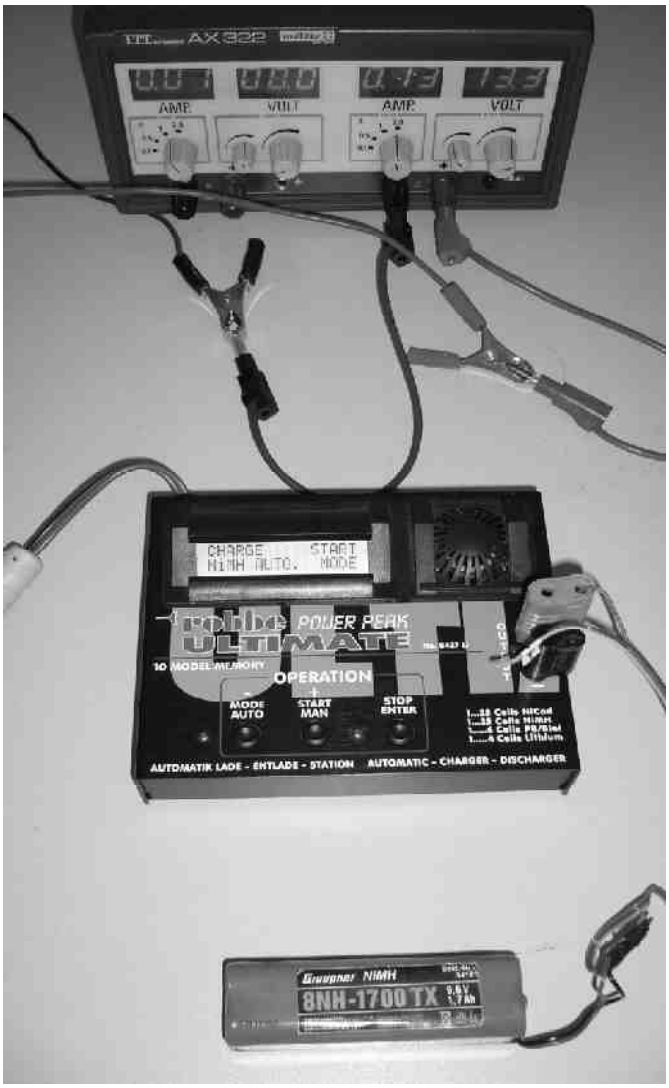
- Une batterie pour le récepteur radio et le servoboard
- Une batterie pour la carte MicroPilot
- Des batteries pour les moteurs
- Une batterie pour la télécommande

CHARGE DES BATTERIES



Les batteries de charge du récepteur et de la carte sont de type Ni-Cd (pack d'accus jaunes). Il faut une batterie pour la carte et une pour le récepteur et le servoboard, car le servoboard peut parfois consommer un fort ampérage, ce qui risquerait de désactiver MicroPilot. La charge de ces éléments est très facile avec le chargeur robbe Ultimate disponible au club Microdrone. Il suffit d'alimenter le chargeur avec l'alimentation, sous une tension d'environ 13 volts. Puis faire défiler sur le chargeur les modes de charge automatique avec le bouton « mode », et choisir un cycle décharge-charge pour un accu Ni-Cd. Le fait de décharger l'élément avant la charge évite l'effet mémoire. Brancher ensuite la batterie avec le cordon approprié et appuyer sur start.

Pour la télécommande, la procédure est la même. La batterie est de type Ni-Mh, choisir donc le mode approprié.

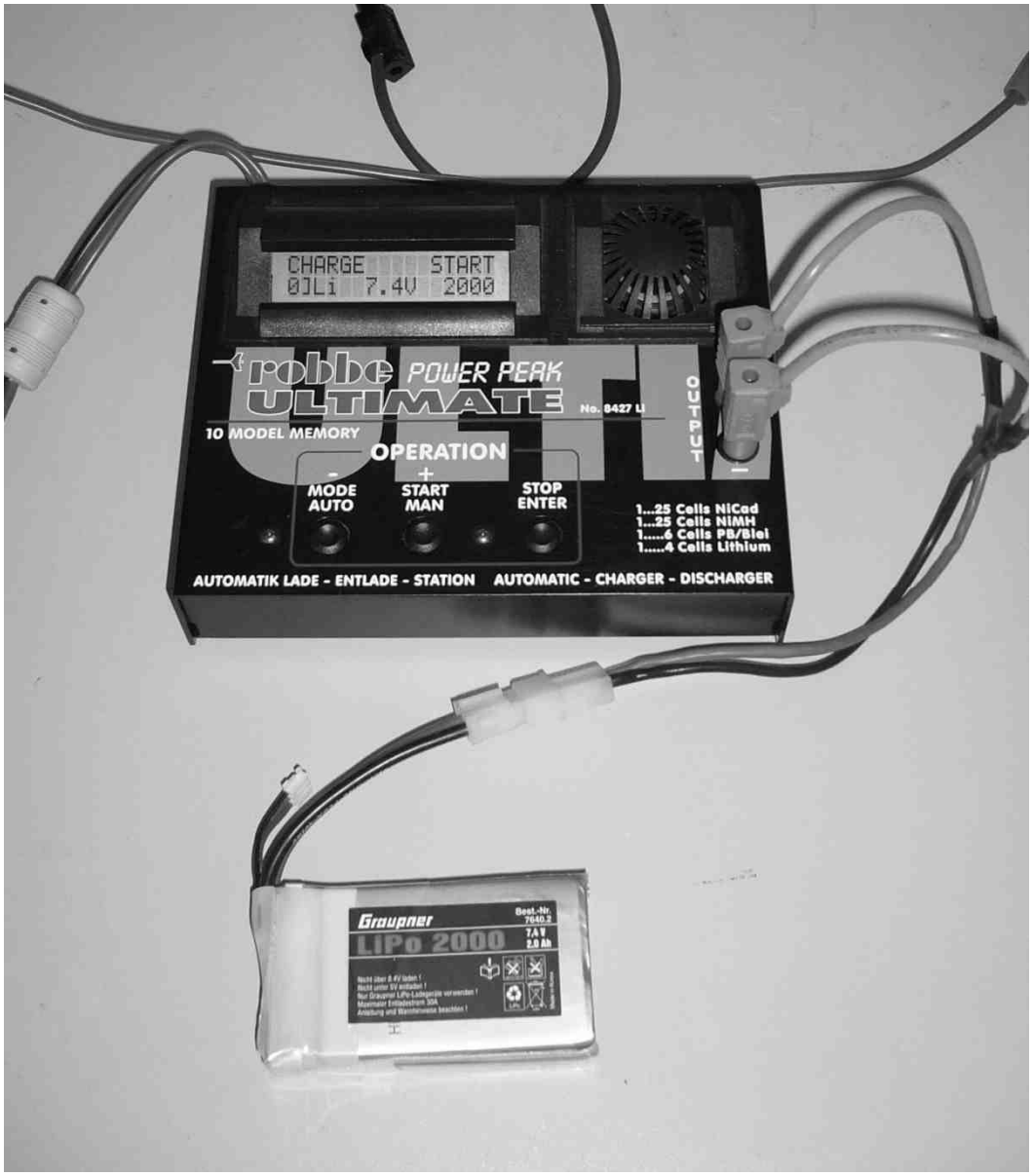


Charge de la batterie de télécommande

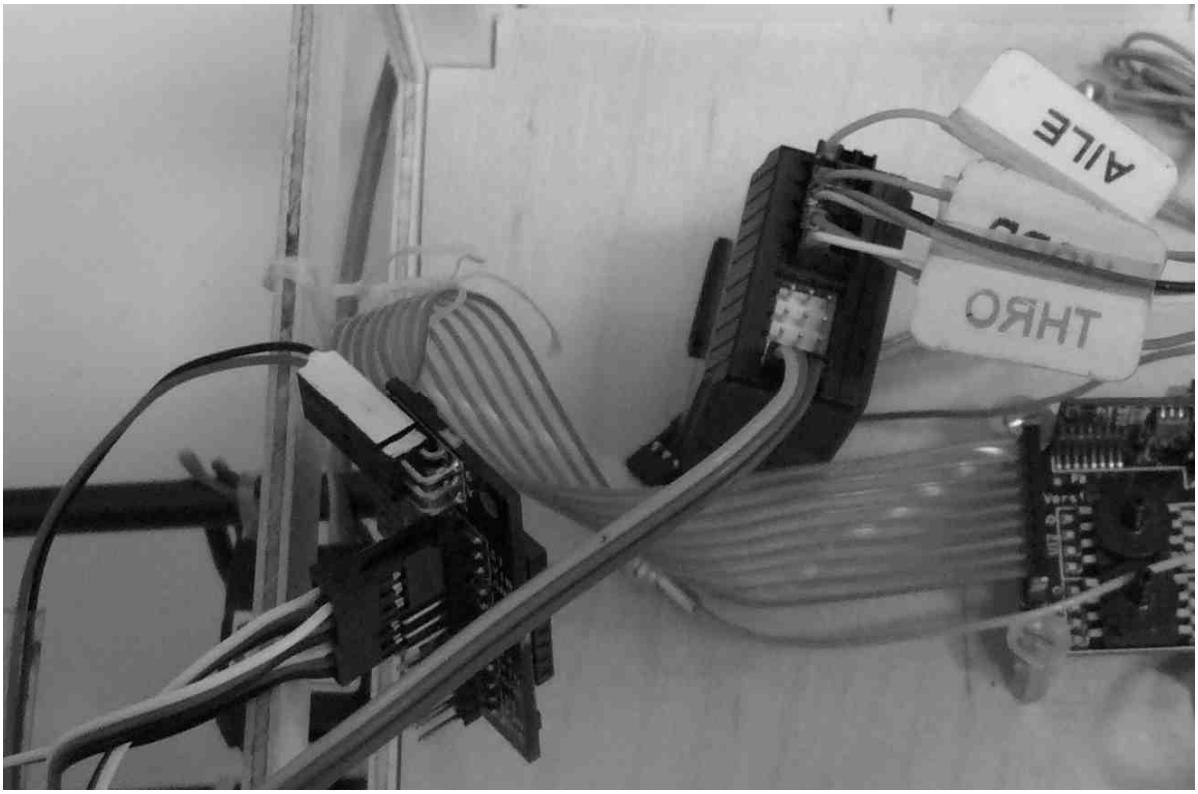
Branchement de la batterie dans la télécommande : la masse vient à gauche

La charge est plus compliquée en ce qui concerne les accus Li-Po des moteurs. Ces accus risquent de s'enflammer voire d'exploser lors d'une mauvaise utilisation. Il s'agit d'accus à deux éléments qui délivrent une tension de 7,4V. Leur avantage est leur légèreté et leur forte capacité. Cependant, ces batteries risquent d'être détruites si elles subissent une tension de charge trop importante, ou si elles sont déchargées en dessous d'une certaine tension. Ces valeurs sont précisées sur la batterie. Il ne faut donc pas les utiliser trop longtemps et les recharger fréquemment. Leur recharge peut s'effectuer avec le même chargeur, en se référant au manuel concernant les accus Li-Po.

Un programme de charge est préenregistré dans le chargeur. Pour y accéder, appuyer sur « ENTER » au démarrage. Vérifier en faisant défiler les données avec les touches + et – que les paramètres de charge sont corrects, puis sélectionner « start » et appuyer sur « enter ».

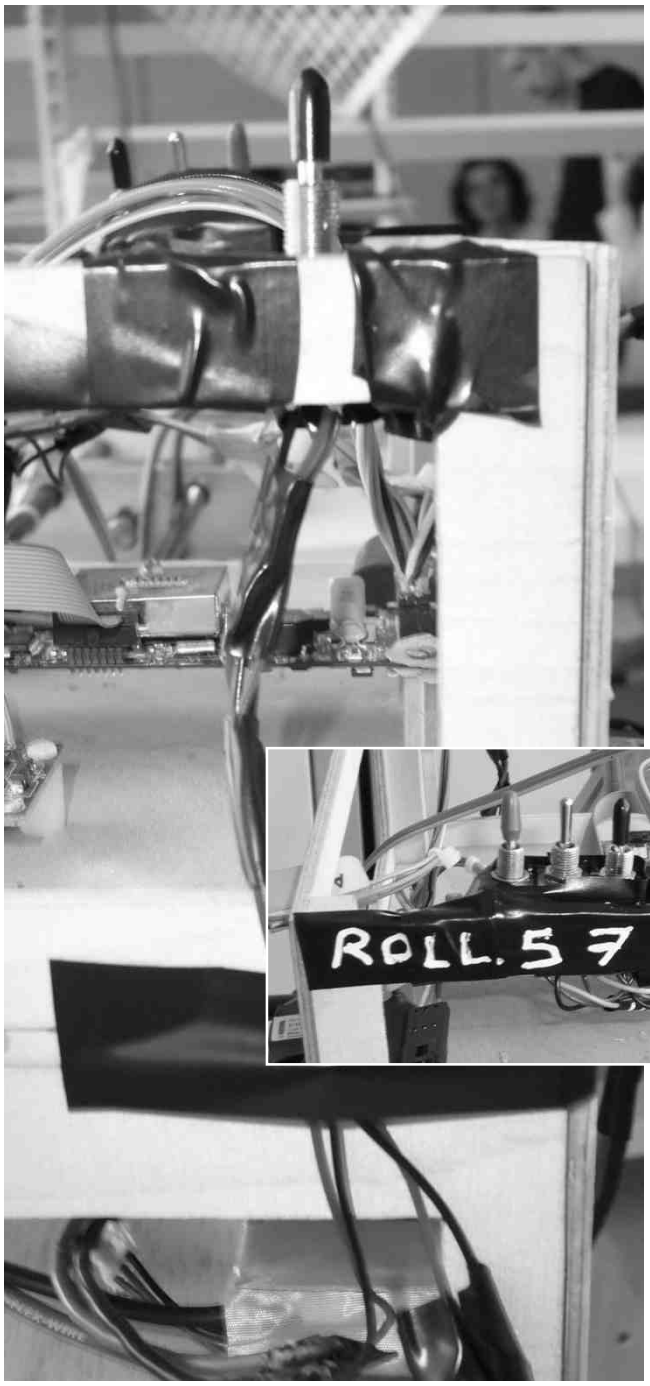


Une fois les batteries chargées, connecter une batterie Ni-Cd au câble reliant le servoboard et le récepteur. Si ce câble est débranché, faire attention au sens de branchement sur les éléments ! La masse est indiquée sur le servoboard et dans la documentation MicroPilot. Sur le récepteur, elle doit être placée « vers le bas » comme indiqué sur la photo :



L'autre batterie Ni-Cd sert à alimenter la carte MicroPilot (fiche PWR). Elle ne doit être connectée que lorsqu'on veut démarrer la carte.

Les batteries Li-Po sont stockées à l'étage inférieur, et reliées aux prises des moteurs par les prises Molex. Nous avons effectué un câblage qui reçoit deux batteries. Chaque batterie alimente un axe de moteurs, afin de garantir une légère stabilité en cas de défaillance d'une batterie. Avant de connecter les accus, vérifier que les interrupteurs d'alimentation des moteurs sont bien ouverts !



Interrupteur d'alimentation moteur

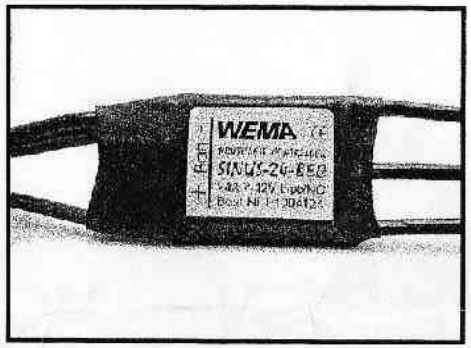
Interrupteurs servos

Avant de procéder à un essai, il faut également s'assurer que les éléments sont bien fixés dans le drone.

On peut alors allumer la télécommande (s'assurer que la manette des gaz est en position basse !) et brancher la carte MicroPilot, on ne doit alors plus toucher la télécommande avant que la carte ne soit prête. Lorsque celle-ci est prête, on branche les variateurs sur le servoboard en basculant les six interrupteurs servos vers l'extérieur du drone. Les moteurs émettent alors un bip. A ce moment, il faut basculer les deux interrupteurs d'alimentation des moteurs. Le drone est alors prêt à voler.

WEMA

FlyWARE



Notice d'utilisation

Contrôleur Brushless

Série - Sinus NC/Lipo

WEMA Elektronik – Wendelhofstraße 6 – 78120 Furtwangen
www.flyware.de

Félicitations,
pour l'achat du nouveau contrôleur pour moteur Brushless "Flyware". Vous disposez d'un produit à la pointe de la technologie et profitant d'excellentes qualités pour valoriser vos modèles. Une attention particulière a été adoptée sur la robustesse et la facilité de programmation par l'émetteur.

Il est cependant nécessaire de disposer de quelques connaissances de base pour aboutir à une bonne utilisation du contrôleur. Grâce à cette notice, vous comprendrez assez rapidement le fonctionnement du contrôleur. Pour avancer le plus rapidement possible, nous vous conseillons de lire attentivement cette notice et surtout les conseils de sécurité, avant de mettre le contrôleur en route. Nous vous souhaitons avec ce contrôleur de la série "Sinus", beaucoup de succès et de plaisirs.

Glossaire:

1. Sécurité / Exclusion de responsabilité

La mise en route et le pilotage des modèles RC, nécessite une multitudes de connaissances techniques, une prudence élevée et une attitude responsable. Des négligences lors de la conception ou le pilotage du modèle, pourront avoir des récercussions graves sur le matériel ou les personnes. Travaillez toujours de façon pointue et utilisez votre modèle avec précautions.

Un moteur électrique avec son accu branché, pourra suite à un problème technique ou mécanique se mettre en route accidentellement. Cela pourra aussi se produire si la réception est activé et l'émetteur est éteint. Dans ce cas il y a risque de blessure grave, par une hélice ou une pâle qui s'est mise à tourner accidentellement. Toute pièce susceptible de tourner par propulsion électrique représente un danger. Evitez de vous placer à un endroit dangereux, près d'une hélice. Veillez aussi, à ce qu'aucun objet ne puisse être en contact avec l'hélice ou les pâles en rotation !

Protégez le contrôleur contre la poussière, la saleté, l'humidité et les pressions mécaniques. Evitez de le placer dans la chaleur excessive ou le froid (respectez les indications techniques) ou les vibrations anormales. Pensez à contrôler de temps en temps le bon fonctionnement du contrôleur. Lors d'utilisation d'accus et de chargeur d'autres fabricants, respectez les indications d'utilisation de ces derniers.

Les contrôleurs ne sont pas protégés contre les inversions de pôles, un branchement inversé des câbles au niveau de l'accu entraînera la destruction du contrôleur. Si le moteur tourne à l'envers, vous pourrez inverser le sens de rotation en inversant le branchement de deux des trois fils de phases du moteur (C'est égal les quels). N'inversé jamais les fils d'alimentation de l'accu. Nos produits sont conçus pour une utilisation dans les modèles réduits, leur utilisation dans des modèles de grandeur réel est interdite.

Ne détachez jamais l'accu du contrôleur quand le moteur est en route, dans ce cas il y a risque de surtension (Effet générateur), ce qui pourrait détruire le contrôleur. Si vous n'utilisez pas le modèle, détachez l'accu du contrôleur. Sur les contrôleurs BEC (Sans couteur Opto inviolable),le risque d'interférence est plus élevés. Effectuez dans tous les cas avant tout vol, des tests de portée de votre systèmes de réception .
Veillez à utiliser des câbles le plus court possible entre le contrôleur et l'accu. Des câbles trop long pourrraient entraîner la destruction du contrôleur. Vérifiez quotidiennement les bonnes liaisons du contrôleur vers l'accu, du moteur et du récepteur.

En tant que fabricant, nous ne disposons d'aucun contrôle sur la bonne utilisation du modèle par le client final. Nous ne pouvons que prévenir des risques éventuels encourus lors d'une mauvaise utilisation du matériel. De ce fait nous déclinons toutes responsabilités. Nous n'assurons ni les coûts des dégâts matériels, ni les risques de blessure causés par une mauvaise utilisation de notre matériel.

2. Caractéristiques techniques

	Sirius 6A	Sirius 8/12A	Sirius 18/25A
Tension autorisée:	4-12 V	7-12 V	7-16V
Nb. d'éléments NC:	4-10 éléments	7-10 éléments	7-12 éléments
Nb. D'éléments Lipo:	1-3 éléments	2-3 éléments	2-4 éléments
Courant continu max:	6 A	8/12 A	18/25A
Crêtes (5s):	8 A	10/14A	22/30A
Mode de fonctionnement:	2	2	2
Mode d'accu:	3	3	4
Coupleur Opto:	-	-	-
BEC:	3,6/5V 2A/1,5W	5V/2A/2W	5V/2A/2,5W
Organe de sécurité:	Protection tension Basse et température	Protection tension basse et température	Protection tension basse et température
Frein-EMK:	Selon mode	Selon mode	Selon mode
Domaine de température:	-10° à +70°C	-10° à +70°C	-10° à +70°C
Dimensions:	18x24x8.5 mm	22x24x8.5 mm	45x25x6.5 mm
Poids:	7 g	9,3 g	22,5 g

3. Divers indications

Les contrôleurs Flyware sont des sélecteurs électroniques extrêmement complexes. Nous sommes arrivés à produire un contrôleur aux dimensions réduites. Les trois câbles noirs sur le coté gauche, sont les connexions vers le moteur, sur le coté droit se situe les sorties vers l'accu d'alimentation de couleur noir (-) et rouge (+) ainsi que la liaison vers le récepteur. Sur la face supérieure (Coté étiquette) se trouve les transistors de rendement, c'est ce coté qui devra être refroidi au mieux.
Lors de la conception de ce produit, nous y avons incorporé une multitude de petits détails très utiles. Le principal étant le mode de programmation simplifié. Les contrôleurs de la série "Sirius" disposent de deux modes de fonctionnement, dans les quelles les caractéristiques typiques de la catégorie des modèles ont été programmé.
Sur cette génération de contrôleur, il n'est plus nécessaire de régler le Timing !
Ce sera le contrôleur qui reconnaîtra le moteur à l'autre bout et se positionnera automatiquement dans le bon Timing !

Mode de fonctionnement 1 (Planeur)
Les principaux paramètres de ce mode, (Adapté pour planeur et Hotliner) sont :
- Reconnaissance de sous-tension
- Protection thermique
- Frein moteur EMK- actif
- Dynamique de démarrage : soft

Mode de fonctionnement 2 (Modèle à moteurs et bateaux)
Les principaux paramètres de ce mode, sont :
- Reconnaissance de sous tension
- Protection thermique
- Frein moteur EMK- à l'arrêt
- Dynamique de démarrage : maximale

Mode Accu 1 (éléments en NC/NiMh)
Reconnaissance automatique du nombre d'éléments:
- Indication de défaut lors de sur/sous tension

Mode Accu 2 - 2 éléments Lipo (Sur Sirius 6A : 1-3 Lipo)
- Fonction de contrôle spécifique pour mode Lipo:
- Indication de défaut lors de sur/sous tension
- Arrêt progressif Automatique si la tension est inférieure à 3V/élément
- Surveillance du chargement des accus, pendant le branchement de la batterie. (Indications d'erreur sur accu déchargé) < 10%

Mode Accu 3 - 3 éléments Lipo
- Fonction de contrôle spécifique pour mode Lipo:
- Indication de défaut lors de sur/sous tension
- Arrêt progressif Automatique si la tension est inférieure à 3V/élément
- Surveillance du chargement des accus, pendant le branchement de la batterie. (Indications d'erreur sur accu déchargé) < 10%

Mode Accu 4 - 4 éléments Lipo **(uniquement sur Sirius 18/25A)**
- Fonction de contrôle spécifique pour mode Lipo
- Indication de défaut lors de sur/sous tension
- Arrêt progressif Automatique si la tension est inférieure à 3V/élément
- Surveillance du chargement des accus, pendant le branchement de la batterie. (Indications d'erreur sur accu déchargé) < 10%

4. Mise en route

Lorsque vous branchez l'accu au contrôleur, vous allez entendre un premier son court vous indiquant la reconnaissance de la tension (Accu). Deux secondes après, un deuxième signal sonore vous indiquera le mode de fonctionnement et après le mode d'accu choisi. Vous entendrez un ou deux sons se suivant rapidement pour le mode de fonctionnement et après deux secondes; un, deux, trois ou quatre sons se suivant rapidement vous indiquant que vous êtes dans les mode d'accu. Le nombre des sons, dépendra du mode programmé. Les sons seront émis par le moteur !

Veuillez noter, que les émissions sonores se feront uniquement après la reconnaissance d'un signal d'émission correct, avec la manette des gaz sur position arrêt "0". Si le contrôleur ne reçoit pas de signal d'émission, 10 secondes après avoir branché la tension électrique (Accu), une indication de défaut sera émise.

Lors de l'installation dans le modèle, veillez que les câbles et branchements ne soient pas tendus mécaniquement. Veiller qu'une bonne circulation d'air soit établi autour du contrôleur, cela évitera que la protection thermique ne se mette en route. Eloignez le contrôleur le plus loin possible par rapport au récepteur.

5. Programmation

Les contrôleurs sortes d'usine, préprogrammé en mode de fonctionnement 1 et en mode d'accu 1, fonctionnent ainsi directement avec la plus part des émetteurs du marché ou les courses de servos sont réglées à 100%.
Si vous désirez modifier un mode ou reconnaître une nouvelle configuration des "Gaz", il faudra lancer une nouvelle programmation. Définissez auparavant les modes souhaités, puis suivez les indications suivantes.

Allumez l'émetteur
Mettez la manette des gaz à 100%
Branchez l'accu de propulsion

Patientez jusqu'à ce qu'une suite de 3 sons soient audibles.
Dans un délai de 3 secondes: Placez la manette des gaz à l'arrêt "0" (Stopp)

Programmation du mode de fonctionnement

Mode de fonctionnement 1	**Mode de fonctionnement 2**
Manette sur mi-Gaz	Manette sur mi-Gaz
Attendez jusqu'à ce qu'une suite de deux sons suite de deux sons soient audibles	Attendez jusqu'à ce qu'une suite de deux sons soient audibles
Baissez la manette des gaz à "0"	Baissez la manette des gaz à "0"
Manette sur plein Gaz	Manette sur mi-Gaz.
Attendre qu'une série de 3 sons soient audibles	Attendre qu'une série de 2 sons soient audibles
Placez la manette sur"0" gaz	Placez la manette sur "0" - arrêt
	Mettez la manette sur plein Gaz
	Attendre qu'une série de 3 sons soient audibles
	Placez la manette sur "0" - arrêt

Programmation du mode d'accu

Mode 1	Mode 2	Mode 3	Mode 4
Manette demi-gaz Attente suite 2 sons Manette sur "0" gaz	Manette demi-gaz Attente suite 2 sons Manette sur "0" gaz	Manette demi-gaz Attente suite 2 sons Manette sur "0" gaz	Manette demi-gaz Attente suite 2 sons Manette sur "0" gaz
Manette sur plein gaz Attente suite 3 sons Manette sur "0" gaz	Manette demi-gaz Attente suite 2 sons Manette sur "0"gaz	Manette demi-gaz Attente suite 2 sons Manette sur "0" gaz	Manette demi-gaz Attente suite 2 sons Manette sur "0" gaz
	Manette sur plein gaz Attente suite 3 sons Manette sur "0" gaz	Manette sur demi-gaz Attente suite 2 sons Manette sur "0" gaz	Manette sur demi-gaz Attente suite 2 sons Manette sur "0" gaz
		Manette sur plein gaz Attente suite 3 sons Manette sur "0" gaz	Manette sur demi-gaz Attente suite 2 sons Manette sur "0" gaz
			Manette plein gaz Attente suite 3 sons Manette sur "0" gaz

Fin de programmation

Après la fin de la programmation, un rappel des signaux va suivre vous indiquant le programme mémorisé. Vérifiez si c'est le bon.

Attention :

Après le rappel de programmation, les modes sont mémorisés et le moteur sera prêt à démarrer. Respecter les indications de sécurité !

5. Indications d'erreurs

Il est impossible de faire démarrer le moteur, après la programmation du contrôleur. Le moteur émet un signal d'erreur. Cela peu avoir plusieurs causes :

a.) Contrôlez le signal d'émission
b.) Contrôlez la tension électrique
c.) Programmez le bon mode d'accu.
Sur les éléments Lipo, une sécurité extrême est nécessaire, c'est pour cela que le contrôleur effectuera un auto contrôle de la tension, dès le branchement de l'accu. Si le programme du mode d'accu n'est pas le bon, il n'y aura pas de démarrage et le contrôleur signalera un message d'erreur.

7. Garantie

Les contrôleurs Flyware sont garanties pour une durée de 24 Mois. La facture d'achat faisant office de bon de garantie. Une éventuelle réparation ne prorogera pas la durée de garantie.

Si pendant la période de garantie, une panne ou un dysfonctionnement dû à la fabrication du produit est constaté, tous les frais de réparation seront à notre charge. Les pannes ou destruction dû à un mauvais emploi du produit, sont exclues de la garantie.

Expédiez le produit à l'adresse indiqué à la fin de cette notice. Le produit sera expédiez à vos frais et les frais de port pour le renvoi après la réparation sous garantie seront à notre charge.

Des envois en port dû, seront refusés. L'assurance pour le produit renvoyé en SAV sera à la charge du client, nous déclinons toute responsabilité, même en cas de perte du colis.

Pour que la réparation sous garantie puisse être acceptée, veuillez respecter ces quelques points :

- Rajoutez dans votre colis la facture d'achat.
- Le produit aura été utilisé conforme à la notice d'utilisation.
- Le contrôleur n'aura pas été ouvert ou réparé par un autre service que le notre.
- Les dommages dû à l'humidité, à un objet étranger, à une inversion de pôle, une surtension ou une pression mécanique, seront exclus de la garantie .
- Joignez un packet, en y indiquant le problème ou le disfonctionnement constaté.

Attestation de conformité
Le société WEMA Elektronik - Wendelhofstr. 6 - D-78120 FURTWANGEN, déclare que tous les contrôleurs Brushless de la gamme Sinus, Easy wave, Wave et Power wave, respectent les normes EN55014 de 1993 et EN55104 de 1995 sur la compatibilité électro-magnétique, selon la directive européenne 89/336 EWG.

Distributeur exclusif pour la France et la Belgique :

ASE Distribution
ZA - 7 , rue des Paîens
F-67720 HOERDT
Email: ASE@tele2.fr

High-end Technology RC®

Brushless Motors and ESC's · ARF Models · NEW Components

Typ Best.-Nr.	Typhoon micro 6/13	Typhoon micro 6/23	Typhoon micro 6/24	Typhoon micro 6/2014	Typhoon micro 15/13	Typhoon EDF 2W	Typhoon EDF 3W
Voltage	7,2-14,4V	7,2-14,4V	7,2 14,4V	7,2-14,4V	7,2...12V	7,2 -11,1V	8,4-15V
KV	2450	1500	1450	1500	1200	4580	2980
No of winds	13	23	24	20	10	2	3
Curr draw max efficiency	5-8	6-10	6-12	6-12	10-15	38	30
Max power ***	140 Watt	90 Watt	85 Watt	100 Watt	200 Watt	350 Watt	350Watt
Max efficiency	80 %	80%	80%	80%	82 %	78 %	78%
Direction	R and L	R and L	R and L	R and L	R and L	R and L	R and L
No poles	10	10	10	14	10	6	6
Length	40	40	40	40	50,5	49	49
Length (without axle)	25mm	25mm	25mm	25mm	34mm	37mm	37mm
Diameter in mm	29	29	29,2	29,2	29,2	28	28
From axle length	14	13	12	12	13	12	12
Shaft diameter	3,17	3,17	3,17	3,17	3,17	3,17	3,17
Weight aprox	43g	43g	43g	43g	73g	78g	78g
Recommended props	4.7x4.7 3S1P 5x5 3S1P 5.2x5.2 3S1P	8x3.8 3S1P 9x3.8 3S1P	8x3.8 3S1P 9x3.8 3S1P	8x3.8 3S1P 9x3.8 3S1P	11 x8 8C	4.7 x 4.7 on 7-8 cells 480 size fan on 3S Lipo	4.7 x 4.7 on 7-8 cells 480 size fan on 3S Lipo
A/C type	Pylon	3D	3D	3D	Glider	Pylon racer EDF JET	Pylon racer EDF JET
Modelweight	300-500 gram	250- 700 gram	250-700 gram	250-700 gram	700-2000 gram	500 -1200 gram	500-1200 gram
Recomended speed controller	Tsunami 10 Tsunami-18	Tsunami-10	Tsunami-10 Tsunami -18	Tsunami-10 Tsunami -18	Tsunami-18 Tsunami -30	Tsunami 35 Tsunami 45	Tsunami-30

**** If you go over these Wattage you risk overheating and damage the motor use sufficient cooling

BIBLIOGRAPHIE

ALEXIS FRENOT – ANTHONY GOSSMANN – ROMARIC GUILLERM

« Stabilisation d'un quadrirotor » Rapport de PIP 2005-2006

ANDREAS GÖRMER

« Flight Dynamics of a mini UAV » Rapport de stage Ensica

JOËL BORDENEUVE-GUIBÉ

« Commande des systèmes linéaires » Cours d'automatisme de deuxième année Ensica

YVES BRIÈRE

« Commande des systèmes par calculateur » Cours de module de deuxième année Ensica

BERNARD MASURE – PHILIPPE POLYCRONIADIS – FERNAND D'AMBRA

« Introduction à l'étude du comportement de l'hélicoptère » Cours de troisième année Ensica

DOCUMENTATION DIVERSE

« MP2028g Installation and Operation » Manuel MicroPilot

Manuels d'utilisation des composants