Stabilisation d'une aile volante

Rapport*

Lubin Kerhuel Jacques Auriol

27 juin 2002

*Une copie de ce rapport se trouve sur le site http://lubink.free.fr

Table des matières

Table des figures

Remerciements

Nous remercions vivement Monsieur Carriere d'avoir accepté notre proposition de projet et de nous avoir suivis tout au long de sa réalisation.

Nous remercions aussi Valentin Arboux de son aide concernant le microcontrolleur. Ainsi que Patrick du service de reproduction de l'ESIEE pour nous avoir relié un certain nombre de documents.

Nous remercions aussi toutes les personnes partageant leurs connaissances et expériences dans ces domaines, via internet.

1 Outils Utilisés

1.1 matlab

Matlab est un logiciel d'analyse numérique très complet qui permet de tester des modèles et différents concepts de filtre à l'aide de Simulink ainsi que de visualiser les signaux logiques de l'électronique embarquée. Il nous a donc servi à la fois de simulateur et d'analyseur logique. Il a été possible de combiner les deux : simuler en temps réel un filtre à partir des données recueillies par l'analyseur logique. Une description complète des différents packages utilisés ainsi que des outils réalisés à partir de ces packages se trouve dans les annexes A page 22.

1.2 Programmation PIC

1.2.1 MPLAB

Pour programmer le PIC, nous avons utilisé le logiciel MPLAB qui est l'outil développé par Michroship. C'est la référence pour la programmation des PICs. La version gratuite met à disposition un assembleur, un lieur ainsi qu'un débogueur. Il est possible d'ajouter des extensions de Microchip ou d'autres groupes. Nous avons ici utilisé le compilateur C CC5X.

1.2.2 Compilateur C : CC5X

L'avantage majeur de programmer en C est de bénéficier des algorithmes de calcul 16 bits et flottants. De plus, le programme ainsi réalisé gagne en lisibilité.

1.2.3 Possibilités de CC5X

Une version démonstration de CC5X est disponible à l'adresse : http://www.bknd.com. Cette version est cependant limitée. Il existe deux limitations. La première est la non disponibilité de toutes les bibliothèques mathématiques. Ainsi, les seuls types qu'il est possible d'utiliser sont :

- int8 et 16 : nombre entier signé
- uns8 et 16 : nombres entiers non signés
- float24 : flottant de 24 bits ayant une résolution de 4,8 digits

Toutes les opérations (+ / * - % ...)avec ces nombres sont cependant autorisées.

La seconde limitation est une limite de la taille de code généré fixée à 1Ko. Le PIC 16f877 disposant de 8Ko, c'est dommage, mais on va voir que l'on peut contourner cette limitation dans la section Installation.

Ce compilateur a cependant un inconvénient comparé à d'autres compilateurs : il ne respecte pas la norme AINSI C. Il existe d'autres compilateurs qui respectent cette norme, mais ceux-ci n'existent pas en version démonstration. Il existe aussi de grandes différences de prix entre ces différents programmes.

1.2.4 Installation

CC5X ne s'installe pas. Il suffit de copier tous les fichiers dans un répertoire quelconque. Il faut ensuite copier le fichier Tlcc5x.ini et Cc5x.mtc dans le répertoire de MPLAB avec les autres fichiers ini. Enfin, il ne reste qu'à configurer MPLAB afin qu'il puisse utiliser ce compilateur.

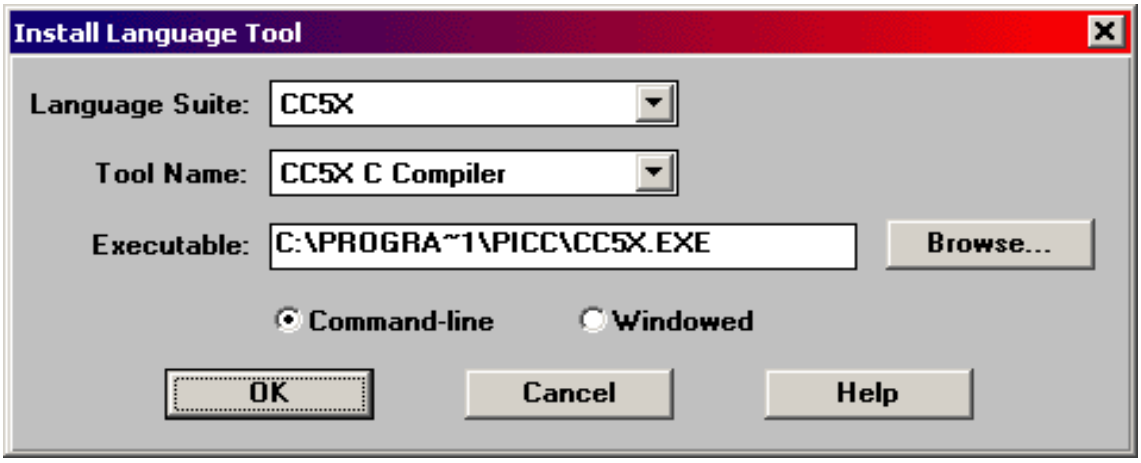

FIG. 1 – *Project -> Install Language Tool*

Dans Tools Name, il faut non seulement configurer le chemin du compilateur CC5X, mais aussi configurer le compilateur MPASM et MPLINK. On peut faire du copier-coller à partir de "language Suite : Microchip".

Á la création d'un nouveau projet, il faut indiquer avant tout le "language Tool Suite" utilisé : CC5X. ensuite, on pourra ajouter des "Node" en assembleur ou en C. Lorsque l'on utilise la programmation assembleur et C, il est nécessaire de générer du code relocalisable : rajouter dans la ligne de commande du compilateur C :

```
-r -xc:\progra~1\mplab\MPASMWIN.EXE -X/q -X/o+
```

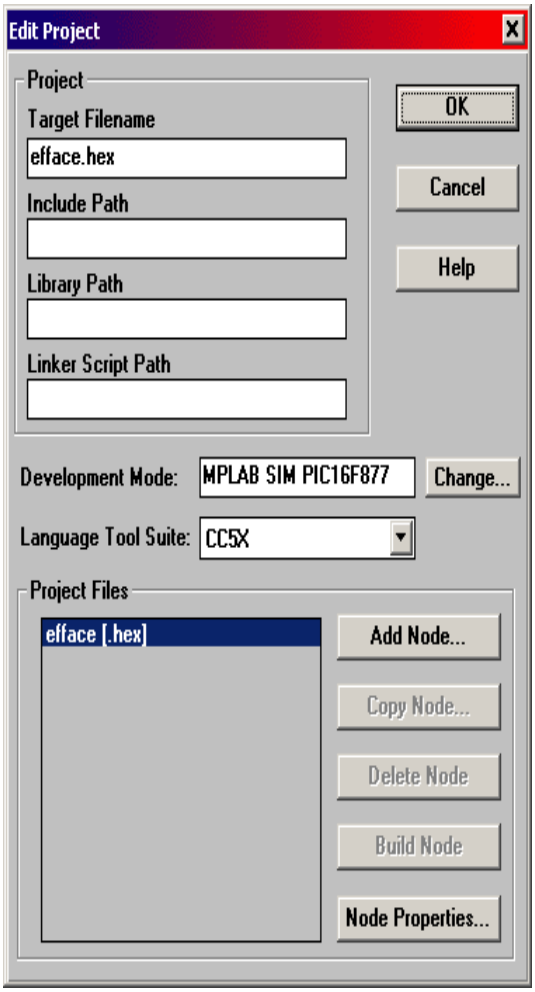

FIG. 2 – *Project -> New Project*

Il faut aussi ajouter dans les "Node" le fichier lkr correspondant au PIC que l'on utilise. Ce fichier permet au lieur de connaître les adresses de mémoires spécifiques du pic. On trouve ces fichiers dans le répertoire de MPLAB.(fig 1.2.4)

Pour passer outre la limite de 1Ko de code généré du compilateur CC5X, on remarque que ce compilateur génère tout de même le fichier assembleur (.asm) correspondant au code. Il suffit donc de compiler ce fichier pour obtenir le programme final. Si plusieurs fichiers asm sont nécessaires, il suffit de créer un autre projet comprenant seulement des "node" en assembleur.

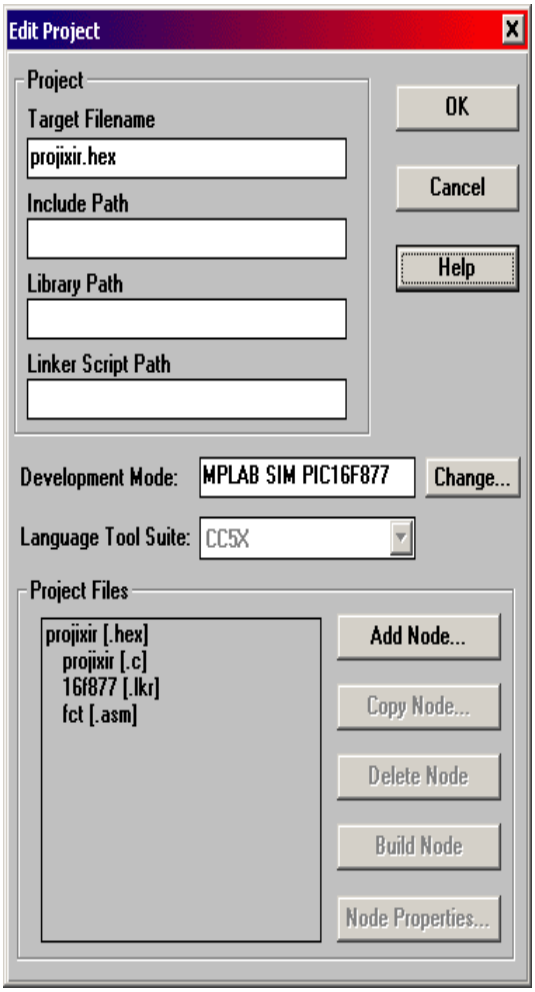

FIG. 3 – *Projet IXIR*

2 L'aile volante

2.1 Modélisme

2.1.1 Choix du modèle réduit

L'avion réalisé est L'IXIR II dont les plans sont sur Internet à l'adresse http://membres.lycos.fr/afstorm/ C'est une aile volante conçue en dépron d'une envergure de 1,25m pesant approximativement 400g. Ses caractéristiques sont les suivantes :

- faible coût de revient
- construction rapide
- grande envergure (plus de 1m) permettant une certaine souplesse dans la création d'un système embarqué point de vue poids et volume.
- bonne résistance aux atterrissages brusques

2.1.2 Entretien

L'aile volante est composée de dépron. C'est une sorte de polystyrene. Il est donc impératif d'utiliser des colles sans solvants car la majorité des colles contiennent des composés qui attaquent le polystyrene et le font fondre.

Les accumulateurs situés dans le nez de l'avion sont des LR03 encore notés AAA. Ce sont des Ni-Mh d'une capacité de 700 mAh.

Les points faibles du modèle sont le nez qui à tendance à s'écraser au fur et à mesure des crashes, ainsi qu'une perte de rigidité entre les deux ailes. Pour le nez, on ne peut pas faire grand chose si ce n'est consolider à l'aide de résine epoxy et de fibre de verre. Pour arranger la perte de rigidité des ailes, il est simple d'ajouter une latte rigide sous l'avion. Une règle plate de 40cm fait parfaitement l'affaire. Un autre point à surveiller est le point d'attache entre la tringlerie et les volets. En effet, l'interface entre le système mécanique rigide et le dépron est critique. Pour les prochains modèles, on pourra renforcer le nez en collant avec de la résine un CD sous le nez afin de répartir les choques, et placer deux morceaux de CD dans les volets aux endroits ou sont situées les chapes.

2.2 Modèle mathématique

L'aile étant un modèle réduit, il n'existe pas de modèle précis caractérisant son vol. En recoupant les équations trouvées dans plusieurs sites internet, Nous sommes parvenus au modèle décrit ici.

Parmi les sites les plus intéressants, le site
http://www.chez.com/aerodynamique/stabilite.html qui décrit le fonctionnement physique d'un planeur téléguidé. Le site
http://www.para-net.org/articles/aero/ dédié aux parapentes, mais qui décrit remarquablement bien les phénomènes liés aux ailes. Tous les profils sont expliqués ici. Et enfin,
http://aeropic.free.fr/pages/modelisme/theorie/theorie.htm qui est un site dédié aux foamies qui sont des ailes volantes téléguidées construites en polystyrène.[1] Ces ailes étant des autostables, on y trouvera des informations sur ce type de profil.

La définition des variables est la suivante :

- θ : Angle de plané
- α : Angle d'incidence
- ϕ : Angle des volets avec la position neutre

La portance et la traînée sont toujours définies par rapport à la direction du vent relatif quelque soit cette direction. Fig(4)

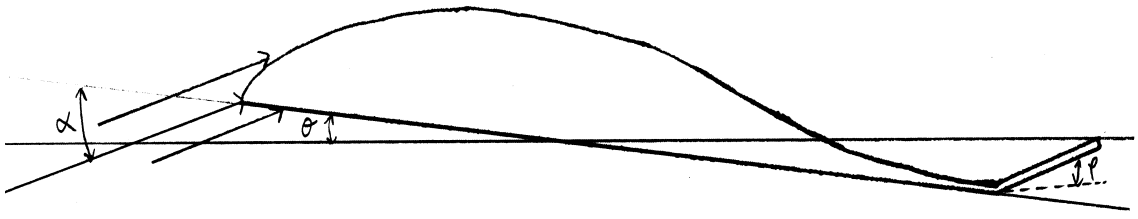

FIG. 4 – *Définition des angles*

La force gravitationnelle s'écrit :

$$\vec{P} = m\vec{g} \tag{1}$$

dirigée vers le bas, elle est négative

1. l'aile volante réalisée ici est en dépron. La technique de construction n'est pas comparable à celle des foamies.

Portance de l'aile :

$$R = \begin{cases} R_p = & (R * \cos\theta) & = \frac{C_p * Ro * S * V^2}{2} \\ R_t = & -(R * \sin\theta) & = -\frac{C_t * Ro * S * V^2}{2} \quad = -\beta * R_p \end{cases} \tag{2}$$

ou C_p et C_t sont respectivement les constantes caractérisant la portance perpendiculaire à la vitesse de l'avion, et la traînée tangente à cette même vitesse.

Action de l'angle des volets ϕ sur la portance : Le coefficient de portance C_p varie linéairement, proportionnellement à l'angle des volets ϕ pour des angles inférieurs à 15° :

$$\Delta C_p = -A * (\phi - \phi_0) \tag{3}$$

ou A est un coefficient qui varie selon que ϕ soit positif (volet levé) ou négatif. La variation de portance est plus importante lorsque ϕ est positif. Nous négligerons ces paramètres dans le modèle en considérant A constant.
ϕ_0 est l'angle pour lequel la portance devient nulle. Cette configuration de vol n'existe pas forcément. Pour trouver ϕ_0, on prolonge la droite donnant la portance en fonction de l'angle α.

Prise en compte de l'incidence α : L'incidence est l'angle que fait l'axe longitudinal de l'aile avec la direction de plané, donc avec l'air.
Le comportement de C_p varie proportionnellement a α. Pour une vitesse donnée, on peut écrire la relation suivante :

$$\Delta C_p \quad = -B * (\alpha - \alpha_0) \tag{4}$$

On considérera ici que B est valable pour toutes les vitesses de l'avion.
α_0 est l'angle d'incidence pour lequel la portance est nulle. Cette angle varie de 0° à +12° suivant les profils. Il est généralement voisin de +5°.

En combinant les équations (2)(3) et (4), on obtient pour le coefficient de portance :

$$C_p = C_{p_0} * -(A * (\phi - \phi_0) - B * (\alpha - \alpha_0)) \tag{5}$$

Ou C_{p_0} est le coefficient de portance lorsque l'avion est en vol stabilisé.

un jeu d'équations qui définissent la portance qu'engendre l'aile s'en déduit :

$$\begin{array}{ll} R_p & = \frac{C_p * Ro * S * V^2}{2} \quad = \frac{(-B*(\alpha-\alpha_0)-A*(\phi-\phi_0))*C_{p_0}*Ro*S*V^2}{2} \\ R_t & = -\beta * R_p \end{array} \tag{6}$$

Projection sur les axes X et Z Avec la convention de signe choisie, on obtient sur les axes X et Z :

$$\begin{aligned} R_z = &\quad R_p * cos\theta + R_t * sin\theta \\ R_x = &\quad -R_p * sin\theta - R_t * sin\theta \end{aligned} \tag{7}$$

Mouvement de rotation de l'aile : Si l'on ne prend pas en compte la résultante aérodynamique de l'aile, l'avancement de celle-ci dans l'air crée un couple Cm_0 dû aux frottements et a des phénomènes d'écoulements. Sur les différents types de profils, qu'ils soient classique ou autostables, ce couple tend à faire croître l'incidence de l'aile.
Le centre de poussée qui est le point d'application de la résultante aérodynamique est situé sur la corde du profil et se déplace en fonction de l'incidence.
Sur une aile classique, il se déplace vers l'avant lorsque l'incidence augmente renforçant ainsi le moment Cm_0. L'aile est instable : L'incidence augmente jusqu'à ce que l'écoulement sur l'aile ne soit plus laminaire. La portance disparaît alors brusquement. La résultante est principalement constituée de la traînée et le centre de portance recule aussi brusquement entraînant l'abattement de l'aile.
Contrairement à ces profils d'aile classique, l'aile volante à un profil autostable. La résultante aérodynamique se déplace vers l'arrière lorsque l'incidence croît. Il en résulte un mouvement piqueur de l'aile qui tend à la ramener vers son incidence initiale lorsqu'elle en est écartée.

Le couple ainsi engendré s'écrit :

$$Cm = -(Cm_0 - 0{,}25 * C_p) \tag{8}$$

Avec $Cm_0 < 0$ du au profil autostable. Voir fig(5).

La variation de l'angle d'incidence α s'en déduit alors :

$$\ddot{\alpha} = \frac{Cm}{I} \tag{9}$$

I est le moment inertiel de l'avion.
On voit sur les équations (13) et (3) que les volets ont une influence sur le couple au travers de C_p.

Principe fondamentale de la dynamique On se place ici dans le repère terrestre que l'on appelle x et z. V_z correspond à la vitesse de chute verticale de l'avion et V_z correspond sa vitesse horizontale. Les équations précédentes ont montré une dépendance par rapport à la vitesse de l'avion en V^2. V^2 ne

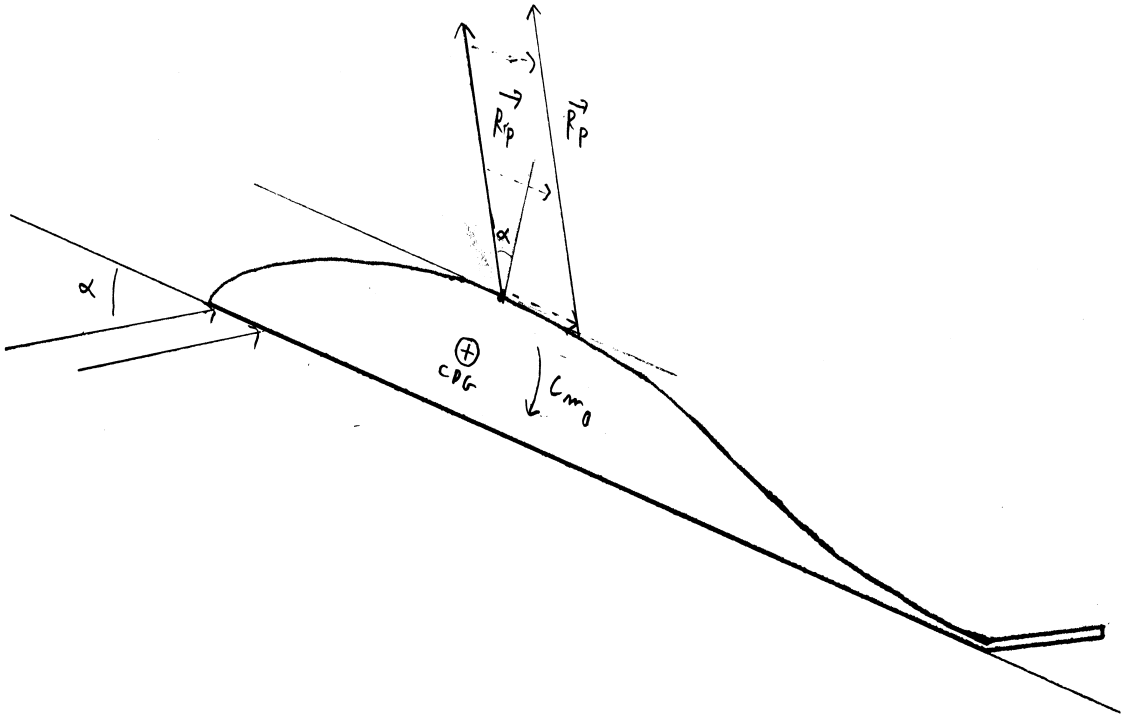

FIG. 5 – *Couple*

dépend pas du repère choisi. Le choix de ce repère est justifié par le calcul de l'angle de plané θ qui se déduit directement des vitesses V_x et V_z comme nous le montrerons un peu plus loin.

$$\begin{aligned} \dot{V}_z &= \tfrac{1}{m}(\vec{P} + \vec{R}_z) \\ \dot{V}_z &= \tfrac{1}{m}(-mg + R_z) \\ \dot{V}_x &= \tfrac{1}{m}(R_x) \end{aligned} \tag{10}$$

$$\theta = \arctan\left(\frac{V_z}{V_x}\right) \tag{11}$$

Ce qui permet d'obtenir l'angle de plané.
La variation de l'angle de plané θ est couplé avec la variation de l'angle d'incidence α. L'évolution de celui se déduit alors des expressions (9) et (11) :

$$\ddot{\alpha} = \frac{d^2}{dt} * \arctan\left(\frac{V_z}{V_x}\right) + \frac{Cm}{I} \tag{12}$$

Condition de stabilité : Pour que l'avion soit stable, il faut que le moment de l'aile le fasse cabrer lorsque l'incidence est trop faible, et qu'il l'entraîne à

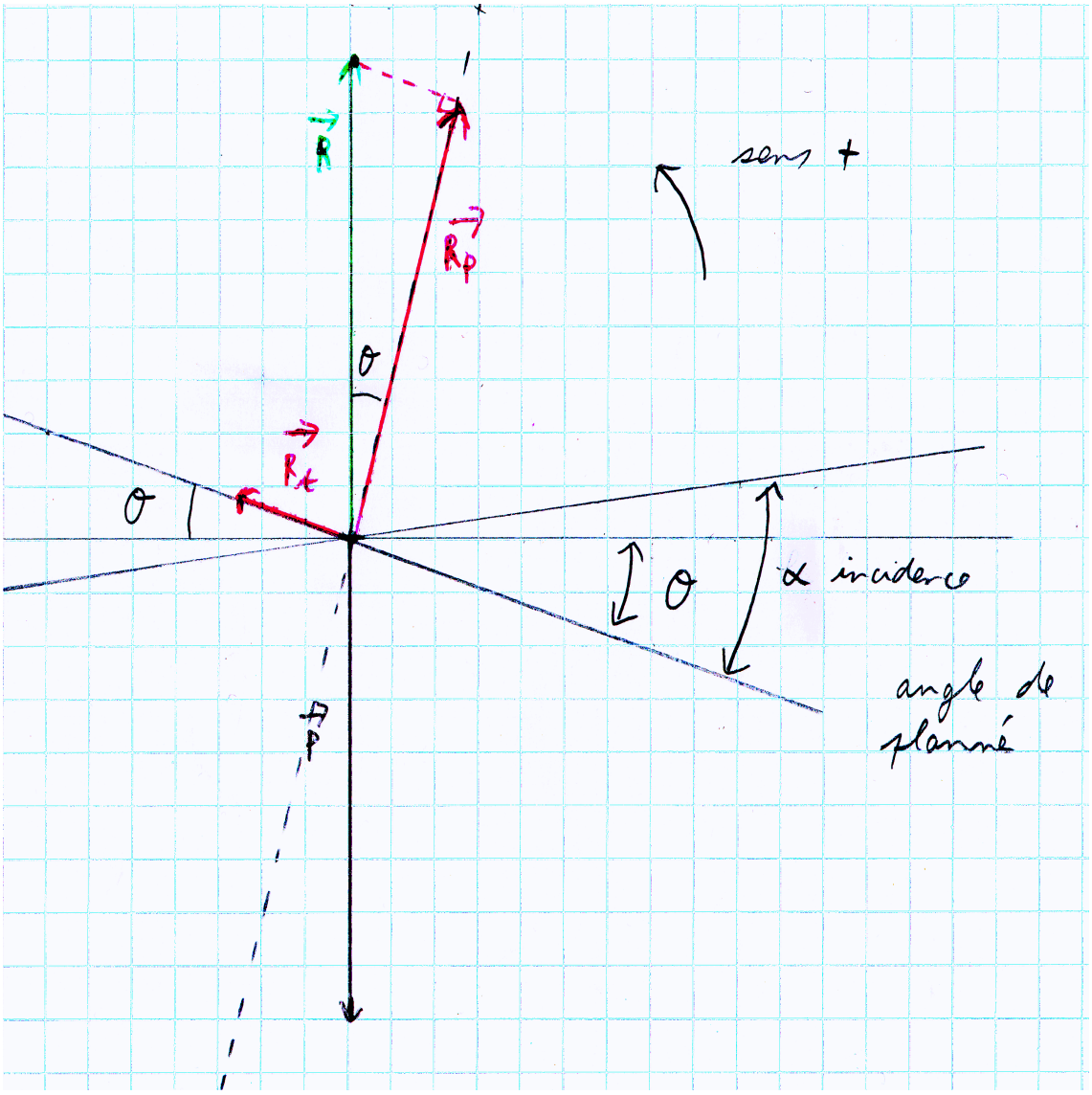

FIG. 6 – *Bilan des Forces*

piquer lorsque l'incidence est trop importante. Cela se traduit ainsi :

$$\frac{dCm}{d\alpha} < 0 \tag{13}$$

En combinant (5) et (8)

$$\frac{dCm}{d\alpha} = -0{,}25 * C_{p_0} * B \tag{14}$$

Comme B est positif, le modèle de l'avion est stable.

3 Choix des composants

3.1 Les capteurs

Le choix des capteurs dépend des grandeurs des phénomènes mesurés. Nous allons nous fier à notre connaissance approximative de l'avion pour caractériser l'ordre de grandeur des paramètres limites. On décide :

- Vitesse de vole compris entre 2 et 10 m/s
- Vitesse angulaire inférieure à 180 deg/s [2]

Pour ce projet, nous avons décidé d'utiliser des gyromètres miniatures. Ces composants sont très récents, et la société Murata est la seule que nous avons trouvé, à en proposer au grand public. Les deux gyromètres dans le commerce sont les suivants :

ENV05F-03

Caractéristiques :
- Poids 50g
- Vitesse Angulaire maximum 60deg/s
- échelle de sortie 27mV/deg
- coût 151E (radiospares)

ENC03-J

Caractéristiques :
- Poids $<$ 1g
- Vitesse Angulaire maximum 300deg/s
- échelle de sortie .63mV/deg
- coût 53E (radiospares)

Le premier modèle présenté (ENV05F-03) est le plus précis, et sans doute le mieux calibré pour les vitesses angulaires prises par l'avion lors d'un vol stabilisé. Cependant, il est lourd, encombrant et onéreux. C'est pourquoi dans un premier temps, nous allons tester les résultats obtenus avec un ENC03-J. (En fonction des résultats obtenus, on avisera...)

2. Lorsque les commandes sont braquées

3.2 le microcontrôleur

Le micro-contrôleur employé est un PIC de chez micochip, choisi pour sa simplicité d'utilisation, qui entraînera un poids embarqué minimum. Le modèle retenu est le 16F877. Ce modèle est très répandu. Il dispose de 8K de mémoire programmable et de 8 convertisseurs analogiques numériques 10 bits. Il est cadencé a une fréquence de 20MHz. Pour fonctionner, il n'a besoin que d'un quartz de la fréquence souhaitée et de deux capacités de 33pF. C'est un micro-contrôleur RISC : Il a peu d'instructions (35 au total) mais exécute toutes les instructions en quatre cycles. Chaque instruction est codée sur 1 octet. La mémoire se divise en trois catégories :

La mémoire morte contient le programme. Elle est de type flash. Cela signifie qu' elle n'a pas besoin d'énergie pour se maintenir. On peut donc débrancher et rebrancher le micro-contrôleur. De plus, elle est effaçable. Il est alors possible de recharger un nouveau programme dans le PIC. Elle se programme avec un programmateur dédié. Celui-ci peut être le même que celui du PIC 16F84. On trouve des schémas simples de ce programmateur sur Internet.

La mémoire vive permet de stocker les variables liées à l'exécution du programme. Celle-ci est totalement effacée lorsque le composant est non alimenté.

L'EEPROM permet de stocker de manière durable des variables. Son utilisation est cependant bien plus contraignante que la mémoire vive. On ne l'utilise donc que lorsque le micro-contrôleur a besoin de stocker des données qui serviront à la prochaine utilisation de celui-ci, ou lorsque l'on veut récupérer les donnés dans un ordinateur, si on ne peux pas utiliser le protocole de communication décrit dans les annexes (enregistrement durant un vol par exemple).

4 Montage électronique

Le montage électronique est très simple. Deux condensateurs et un quartz permettent de faire fonctionner le micro-contrôleur. Le récepteur est connecté à deux entrées numériques du micro-contrôleur. Deux sorties numérique du micro-contrôleur sont connectés aux servos. Enfin, le montage amplificateur du gyromètre est connecté à une entrée analogique du micro-contrôleur.

4.1 Amplification du signal du gyromètre

Au départ, nous pensions réaliser cette amplification à l'aide d'un Amplificateur Opérationnel, mais une erreur de choix du composant ne nous à pas permis de réaliser ce montage avec une alimentation de 0 - 5V. Nous avons finalement amplifié le signal du gyromètre à l'aide de deux transistors montés en Darlington. Les deux résistances sont respectivement de 100 Ohms et de 1Kohm. Le β du transistor est d'environ 565. La sortie du gyromètre est connectée au travers des deux transistors à la résistance de 100 Ohms.

5 Stabilisation

5.1 Asservissement de vitesse angulaire

Á partir du montage effectué, plusieurs types de stabilisateur sont envisageables. La première, qui est la plus facile à réaliser est l'asservissement de commande angulaire. La télécommande permet de contrôler la profondeur comme un avion téléguidé classique, mais pour le roulis, elle donne une commande de vitesse angulaire. Lorsque le manche est au neutre, l'avion doit avoir une vitesse de roulis nulle. Si une rafale de vent perturbe l'avion, le calculateur doit compenser la vitesse de rotation de celui-ci en actionnant les volets.
La correction que nous avons utilisée ici est une simple correction proportionnelle. La formule ressemble à cela :

$$Commande_{Ailerons} = Vitesse_{Gyro} - Commande_{Telecommande}$$

Des coefficients pondèrent cette formule. un filtre faisant une moyenne sur les cinq dernières commandes envoyées aux ailerons permet de limiter les vibrations engendrées par le bruit du gyromètre.

5.2 Stabilisation autonome

Nous avons effectué une tentative de Stabilisation autonome. En intégrant le signal du gyromètre numériquement, le calculateur devait être capable de connaître sa position relative à la position qu'avait l'avion au moment ou l'on a placé le dispositif sous tension. Cependant, le bruit engendré par le gyromètre et l'amplificateur monté derrière fait diverger l'intégrale en quelques secondes. Cette divergence est encore plus rapide lorsque l'avion est en mouvement. Nous n'avons donc pas pu expérimenter cette méthode en vol. Il faudrait testé avec le gyromètre plus précis qui à été décrit précédemment.

6 Conclusion

6.1 Pour aller plus loin

Des petites améliorations faciles sont possibles comme l'ajout de jumper sur le montage électronique qui permettrait de sélectionner les routines exécutées par le micro-contrôleur. Cela permettrait de gagner du temps car le chargement du programme dans le micro-contrôleur est longue.

Il est aussi possible d'améliorer le résultat de la conversion indiquant au micro-contrôleur les deux tensions de références entre lesquelles le signal analogique varie. Actuellement, le micro-contrôleur effectue cette conversion entre 0 et 5V, mais avec quatre résistances, il est possible d'améliorer sensiblement la résolution.

Il est aussi nécessaire de modifier la routine qui récupère les données du récepteur. Dans sa version actuelle, le programme attend le signal des deux voies l'une après l'autre. Or si on inverse les voies, une donnée sur deux sera alors récupérée. Il est possible d'écrire une routine capable de récupérer les deux voies simultanément ou dans "n'importe quel sens".

Pour continuer le projet, il faudrait réétudier le modèle mathématique sous matlab. Une fois un modèle fonctionnel trouvé, il faudrait imaginer des experiences permettant de relever les caractéristiques de l'avion. On peux ensuite aussi caractériser les servos moteurs contenus dans les ailes, ainsi que faire une étude précise des mouvements des ailerons en fonction de la position des servos. Cela revient à étudier géométriquement la tringlerie.

Il faudrait tester d'autres capteurs. Ainsi, il existe des gyromètres plus précis permettant peut être de faire une stabilisation autonome comme il était prévu de réaliser au début du projet. On peut aussi ajouter un "niveau à bulle électronique" permettant de recaler l'intégrale du gyromètre. Ensuite, il est sans doute possible de réaliser un capteur d'incidence et un capteur de vitesse.

Pour la partie programme, on peut imaginer un système d'apprentissage ou l'avion apprendrait au fur et à mesure des vols ses propres caractéristiques.

6.2 Bilan

6.2.1 Objectif initiale

L'objectif initial était de stabiliser l'avion dans le sens ou, connaissant sa position par rapport au sol, il devait être capable de rétablir son horizontal, et de corriger sa trajectoire. L'avion devait donc voler en ligne droite.
Deux difficultés ne nous ont pas permis de réaliser cela. La première difficulté était la réalisation d'un modèle non linéaire sous matlab. Le modèle que nous avons réalisé diverge. Ensuite, un problème de capteur s'est posé. Le gyromètre que nous avons employé ne semble pas être précis pour cette application. A l'origine, ce gyromètre est destiné à analyser les mouvements pour stabiliser l'image des camescopes par exemple.

6.2.2 Connaissances acquises

Bien que l'objectif initiale ne soit pas atteint, nous sommes très contents d'avoir réalisé ce projet. Tout d'abord, En un temps relativement court[3], nous avons réussi en partant de rien, à réaliser une aile dont les caractéristiques de vole sont très bonne, ainsi qu'un montage électronique embarqué fonctionnel. Et Surtout, l'asservissement en commande angulaire du roulis de l'avion fonctionne bien.

De plus, nous avons beaucoup appris pendant ce projet. Tout d'abord, nous connaissons maintenant le fonctionnement générale de Matlab. Plus particulièrement, nous maîtrisons bien Simulink, Stateflow et certains packages comme XPc Target et nous savons mettre en oeuvre quelques astuces permettant d'étendre les capacités de matlab.

Nous avons aussi appris à programmer en C sur le pic ainsi qu'à mélanger du C et de l'assembleur, ce qui permet un gain de temps considérable sur la conception de programme.

3. Projet d'environs un mois et demi

A Matlab

Matlab V6.0.0.88 Release 12 a été utilisé lors de la réalisation de ce projet. Il tournait sous Windows 2000.

A.1 Analyse Temps Réel : Real Time Workshop & XPc Target

Les packages suivants ont été utilisés :

- Real Time Workshop
- XPc Target
- Stateflow Coder : permet de générer un code correspondant au schéma
- Compilateur C : Visual Studio 98[4]
- un petit driver permettant d'accéder aux ports dans simulink même http://www.ntportlibrary.com

Matlab n'arrive pas toujours à reconnaître Microsoft Visual C 98. Il est parfois nécessaire de modifier les fichiers .m utilisés par le processus de compilation.

A.1.1 Description

Le package Real Time Workshop permet de générer du code source en plusieurs langages dont C et ada. Le compilateur peut ensuite compiler ces fichiers afin de permettre une analyse en temps réel.

Il est très intéressant de pouvoir récupérer des données dans Matlab. Pour cela, l'idéal est de relier le pic a Matlab. Cependant, Windows rend l'accès aux ports difficile et l'accès aux ports n'est pas prévu dans la bibliothèque standard de simulink.

Le package XPc Target permet d'exécuter les simulations des schémas provenant de Simulink sur un second ordinateur. Il est intéressant de noter que XPc Target comprend des blocs permettant d'écrire et de lire des données sur les ports du PC servant pour la simulation. Le port parallèle LPT1 peut donc servir pour acquérir et exporter des données. Les adresses de ces trois registres sont généralement 278h, 279h et 27Ah. On peut facilement vérifier ces données dans le bios.

4. Nécessaire pour Xpc Target

Le driver permettant d'accéder aux ports permet de ne pas utiliser le package XPc target. En réalité, le driver ne contient pas de bloc dans simulink, mais ajoute une fonction matport à matlab. Avec cette commande, il est alors possible d'éditer un bloc simulink permettant de lire ou d'écrire dans le port pendant une simulation sur l'ordinateur sur lequel on exécute matlab. L'avantage de cette méthode est qu'on a plus besoin d'avoir deux ordinateurs pour réaliser une simulation. De plus, on gagne en temps de compilation et en temps de téléchargement. En revanche, la simulation avec acquisition en temps réel sous windows est bien plus lente. Windows étant un système multitâche, le temps réel n'est pas respecté et surtout, Simulink ne sait pas simuler en temps réel. Simulink est un outil permettant de réaliser des simulations, mais il les effectue le plus rapidement possible. La seule véritable application de ce driver est donc le déboggage de Stateflow lorsque celui-ci communique avec un système extérieur.

```
function [sys,x0,str,ts] = SimulinkPort(t,x,u,flag,NumPort,Lecture,Ecriture)
%TIMESTWO S-function whose output is two times its input.
%   This M-file illustrates how to construct an M-file S-function that
%   computes an output value based upon its input. The output of this
%   S-function is two times the input value:
%
%     y = 2 * u;
%
%   See sfuntmpl.m for a general S-function template.
%
%   See also SFUNTMPL.

%   Copyright 1990-2000 The MathWorks, Inc.
%   $Revision: 1.5 $

%
% Dispatch the flag. The switch function controls the calls to
% S-function routines at each simulation stage of the S-function.
%
switch flag,
  %%%%%%%%%%%%%%%%%%
  % Initialization %
  %%%%%%%%%%%%%%%%%%
  % Initialize the states, sample times, and state ordering strings.
  case 0
    [sys,x0,str,ts]= mdlInitializeSizes(Lecture,Ecriture);

  %%%%%%%%%%
  % Outputs %
  %%%%%%%%%%
  % Return the outputs of the S-function block.
  case 3
        if (Lecture) sys=mdlOutputs(t,x,u,NumPort)
        else sys = [];
        end


    case 2       %
        if (Ecriture) mdlUpdate(t,x,u,NumPort);
        end

  %%%%%%%%%%%%%%%%%%%%
  % Unhandled flags %
  %%%%%%%%%%%%%%%%%%%%
  % There are no termination tasks (flag=9) to be handled.
  % Also, there are no continuous or discrete states,
  % so flags 1,2, and 4 are not used, so return an emptyu
  % matrix
  % 0 : initialisation
  % 1 : calcule édrivative ( pour bloque continue)
  % 2 : update discret state
  % 3 : Calcule les sorties
  % 4 : Calcule next sample hit ( si variable sample time)
  % 9 : End of simulation task

  case { 1, 4, 9 }
```

```
    sys=[];

  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
  % Unexpected flags (error handling)%
  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
  % Return an error message for unhandled flag values.
  otherwise
    error(['Unhandled flag = ',num2str(flag)]);

end

% end timestwo

%
%=============================================================================
% mdlInitializeSizes
% Return the sizes, initial conditions, and sample times for the S-function.
%=============================================================================
%
function [sys,x0,str,ts] = mdlInitializeSizes (Lecture,Ecriture)

sizes = simsizes;
sizes.NumContStates = 0;
sizes.NumDiscStates = 0;
sizes.NumOutputs  = Lecture;
sizes.NumInputs   = Ecriture;
sizes.DirFeedthrough = 1        % has direct feedthrough
sizes.NumSampleTimes = 1;

sys = simsizes(sizes);
str = [];
x0  = [];
ts  = [-1 0];   % inherited sample time

% end mdlInitializeSizes

%
%=============================================================================
% mdlOutputs
% Return the output vector for the S-function
%=============================================================================
%
function sys = mdlOutputs(t,x,u,NumPort)

sys = matport('Inport', NumPort);

end



function sys = mdlUpdate(t,x,u,NumPort)

sys = matport('Outport', NumPort,u);

end
```

A.1.2 Protocole de Communication Matlab - Pic

En créant un petit diagramme dans Stateflow(fig. 7), il a été possible de créer un protocole afin que Matlab puisse communiquer avec le micro-contrôleur. Nous avons ainsi pu visualiser les variables du micro-contrôleur en temps réel sur l'ordinateur "target". Matlab pouvait aussi envoyer des données au micro-contrôleur.

Le protocole réalisé a les caractéristiques suivantes :

- Permet la transmission de données dans les deux sens
- Transmet des octets de 8 bits
- Seulement 3 fils utilisés + un fil de masse. Donc seulement trois ports du microcontrôleur condamnés.

- La synchronisation est automatique. Il est possible de déconnecter l'un des deux éléments et de le reconnecter.
- Pas d'horloge : les deux éléments se répondent l'un l'autre

Les routines RSsend et RSreceive permettent d'émettre et de recevoir des données dans le registre W. Ces deux routines sont bloquantes. Les données reçues peuvent être erronées lorsque la synchronisation n'est pas encore terminée.

Le bloc Stateflow de Matlab a en entrée une constante qui peut prendre les valeurs 0 ou 1 respectivement pour configurer le bloc en Réception seule ou en Réception/Emission. Lorsque celui-ci est configuré en Réception/Emission, Matlab teste si le PIC est en train d'émettre. Si le pic n'émet pas, alors Matlab considère que c'est à lui d'envoyer une donnée.

L'ordinateur "target" a été considéré comme lent devant le micro-contrôleur lors de la réalisation de ce protocole. Le protocole est symétrique, à ceci près que Matlab-Stateflow comprend des "Time Out" nécessaires pour une synchronisation sure.

L'envoi des données se fait en réalité sur 10 bits. Le premier bit envoyé est toujours 0 alors que le dernier bit envoyé est toujours 1. Cette caractéristique est utilisée lors la synchronisation. Cette méthode a un avantage de simplicité mais un inconvénient si dans la suite d'octets envoyée par exemple, il existe le motif constitué d'un 1 suivi 6 bits de données plus loin au lieu de 10, et pour terminer un 0, il est possible que la synchronisation ne s'effectue pas. Dans ce cas, les données lues seront erronées.

Pour utiliser ce protocole, il suffit de connecter Tx sur Rx, Rx sur Tx et RTx, RTx2 et TRx sur le même fil et sur une pine de lecture/écriture du PC. Celle-ci peut être réalisée en connectant une sortie et une entrée ensemble sur le port lpt1. Cette solution à été utilisée ici. Il a été nécessaire d'ajouter une résistance entre ces deux pines. Nous utilisons un potentiomètre qui nous permet d'adapter le montage à plusieurs ordinateurs. La valeur de la résistance équivalente utilisée est de l'ordre de 400 ohms. Si Matlab n'envoie pas de données, ce potentiomètre n'est pas nécessaire. Pour faire des tests de fonctionnement, il est plus judicieux de mettre une résistance importante, voire pas de résistance du tout.

Les trois fils sont initialement à 1. Lors de l'émission d'une donnée, le fil de l'émetteur RTx passe à 0 ainsi que le Tx. Le destinataire met alors sa ligne Tx à 0. Le Tx du récepteur est utilisé comme accusé de réception. l'émetteur inverse

alors RTx qui passe à 1, et place dans Tx la première donnée. Le récepteur inverse à son tour Tx et ainsi de suite.

A.2 Commandes et infos Utiles

mex -setup permet de sélectionner le compilateur C utilisé par Matlab. Avec xPC Target, le compilateur Visual C++ V6 ou V5 est nécessaire.

xpcsetup permet de configurer les options de xPC target. On configure notamment la méthode de communication entre les ordinateurs host et target. On choisi aussi la destination de la simulation. Celle-ci peut être un ordinateur target ayant démarré grâce à la diskette de démarrage que l'on peut aussi créer ici. On peut aussi faire une simulation sur un ordinateur qui a démarré sous un système DOS et il est aussi possible de créer une diskette de boot avec le schéma inclus. Dans les deux derniers cas, la communication entre les deux ordinateurs n'est plus nécessaire.

www.mathworks.com est le site officiel de Matlab. En plus de contenir toute l'aide en ligne, il contient des informations en plus sur les packages ainsi qu'une bonne description de ceux-ci avec des exemples d'utilisation.

A.3 Outils potentiellement intéressants

Le Package Windows Target semble très intéressant. Il permettrait d'utiliser les blocs du package xPC Target (notamment le bloc permettant d'acceder aux ports du PC) dans une simulation en temps réel réalisée dans windows même. Il serait alors possible d'utiliser le protocole Matlab-Pic décrit plus haut. L'avantage est que l'on n'aurait besoin que d'une seule machine. L'inconvénient comparé à xPC Target serait une exécution de la simulation plus lente ou plus précisément, une valeur maximum de fréquence de simulation plus faible comparée au même modèle exécuté sour xPC Target sur une même machine. Cette lenteur est due à l'exécution des multiples taches windows en parallèle. Cependant, cet inconvénient peut être relativisé car les ordinateurs utilisés comme Target dans xPC Target sont en général des machines de récupération, donc plutôt lentes.
Un autre package winTarget ressemble au package décrit ici, mais est en plus gratuit. Disponible à l'adresse :
http://www.wfw.wtb.tue.nl/control/home_of_wintarget.htm

Real-Time Workshop Embedded Coder permet de générer du code C à partir d'un schéma Simulink et d'un diagramme conçu avec Stateflow en supposant Stateflow Coder installé. Le code C ainsi généré est peut être implémentable dans un micro-contrôleur si l'on dispose d'un compilateur C pour ce dernier. Si on arrive à effectuer ceci, on obtient un gain de temps considérable car il suffit alors de dessiner le programme avec Simulink et Stateflow pour pouvoir le faire tourner dans un micro-contrôleur.

FIG. 1 – Diagramme Stateflow

B Listing Programme

B.1 Parti C

```
#pragma chip PIC16F877

//#pragma codepage 1
#include "math16.h"
//#include "math16m.h" // optim speed
#include "math24f.h"
//#include "math24lb.h" // fcts ( sin  ... )
//#pragma codepage 0

extern page0 void RSinit() ;
extern page0 void RSsend(uns8);
extern page0 uns8 Gyro();

        int16   VarRecepteurTang ;
        int16   VarRecepteurRoul ;

void configuration(void) {

OPTION_REG = 0; /*                      ; Valeur registre option (OPTION_REG) :
                        ;_____________________________________
                                ; 7: é Rsistance pull-up active(0)
                                ; 6: Interruption RB0 front montant(1) ou descendant(0)
                                ; 5: CLK Timer0 interne(0) ou externesur RA4(1)
                                ; 4: Si CLK TImer0 sur RA4, front montant(0) ou descendant(1)
                                ; 3: Prediviseur sur WDT(1) ou Timer0(1)
                                ; 2-0: Valeur de predivision (voir doc)


                    ; configuration des interuptions
                    ; ------------------------------

*/
INTCON = 0; /*                      ; Masque d'interruption principale (INTCON) :
                        ;_____________________________________________
                                ; 7: Interruption global active(1)
                                ; 6: Interruption peripherique active(1)
                                ; 5: Interruption debordement Timer0 active(1)
                                ; 4: Interruption RB0 active(1)
                                ; 3: Interruption RB7-4 active(1)
                                ; 2-0: Flags
*/
PIE1 = 0; /*                        ; Masque d'interruption éperiphrique 1 (PIE1) :

                        ;_____________________________________________
                                ; 7: Interruption lecture/ecriture du Port Parallele active(1)
                                ; 6: Interruption convertion A/D active(1)
                                ; 5: Interruption reception sur USART active(1)
                                ; 4: Interruption emission de USART active(1)
                                ; 3: Interruption Port Serie active(1)
                                ; 2: Interruption Capture1/Compare1/PWM1 active(1)
                                ; 1: Interruption comparaison TMR2/PR2 active(1)
                                ; 0: Interruption debordement Timer1 active(1)
*/
PIE2 = 0;                       /*      ; Masque d'interruption éperiphrique 2 (PIE2) :
                        ;_____________________________________________
                                ; 7: Non affecter
                                ; 6: Reserve
                                ; 5: Non affecter
                                ; 4: Interruption fin ecriture EEPROM active(1)
                                ; 3: Interruption Collision active(1)
                                ; 2-1: Interruption
                                ; 0: Interruption Capture2/Compare2/PWM2 active(1)

                    ; configuration des modes éspciaux
                    ; ---------------------------------

                    ; mode convertisseur analogique/énumrique
                    ; ---------------------------------------
*/
ADCON0 = 0x81; /*   H'0081' ; Registre de config ADC 0 (ADCON0) :
                        ;_____________________________________
                                ; 7-6: Horloge de convertion
                                ; 5-3: Maintenir a 0, é slection de l éentre analogique a convertir
                                ; 2: Flag dé'tat
                                ; 1: Non affecter
                                ; 0: Convertisseur a/d éactiv(1)
```

```
*/
ADCON1 = 8+4+2; /* EQU   8+4+2  ;Registre de config ADC 1 (ADCON1) :
                            ; Port AN0 éconfigur en éentre analogique
                            ;_____________________________
                                    ; 7: Format du resultat sur 10 bits, 000000XX-XXXXXXXX(1) ou XXXXXXXX-XX000000(0)
                                    ; apparement, ce serait éinvers  !!!
                                    ; 6-4: Non affecter
                                    ; 3-0: config des ports d'éentre analogique ou énumrique et de Vref+ et Vref-


*/
#pragma bit RecepteurTang @ PORTB.4
#pragma bit RecepteurRoul @ PORTB.2
#pragma bit ServoGauche @ PORTB.0
#pragma bit ServoDroit @ PORTB.1

TRISB.4 = 1;
TRISB.2 = 1;            // édfinie  en éentre

TRISB.0 = 0;
TRISB.1 = 0;            // édfinie  en sortie

ServoGauche = 0;
ServoDroit = 0;         //  initialise  sortià   0 : servos "mou"

FpRounding = 1;         // arrondi les  calculs  flotant  ( != tronque )
} // Fin configuration



int16  callageRecepteur(int16  Var) {      // Valeur comprise entre -278 et +278
        return (Var-833);   // -(555.5*3/2))
        }


int16  Sature(int16  a) {
                if (a > 278) a = 278;
                if (a < -278) a = -278;
                return a;
                }

void EmmetServos(int16 Sg , int16 Sd) {
                int16    tmp;
        // On Sature les  sorties  pour que le servo  ne fasse  pas de trou  dans l'avion ...

                Sd = Sature(Sd);
                Sg = Sature(Sg);

        // Callibrage  de l'attente  : éopration  inverse  de callageRecepteur

                Sg = Sg +833;    // +(555.5*3/2 ))+
                Sd = Sd +833;    // +(555.5*3/2 ))+

                Sg = Sg*9;
                Sg = Sg/12;     // Corrige édcallage  entre  Emission/reception  ( 12 cycles  / 9 cycles)

                Sd = Sd*9;
                Sd = Sd/12;


                ServoGauche = 1;                // éégnration  de l'impulsion
                while (Sg != 0 ) {Sg = Sg-1;}  // Attente
                ServoGauche = 0;                // Fin impulsion
                                // boucle de 10 instruction


                ServoDroit = 1;          // éégnration  de l'impulsion
                while (Sd != 0 ) {Sd = Sd-1;} // Attente
                ServoDroit = 0;          // Fin impulsion
                                // boucle de 10 instruction
                return;

        }

void recoitRecepteur() {
                VarRecepteurTang = 0;
                VarRecepteurRoul = 0;


//-------------------------------------------------------------------------------
// éRception  des édonnes  du éRcepteur
```

```
//-------------------------------------------------------------------------------

                while (RecepteurRoul == 0) {}                   // èChronomtre le PWM éétlcommande
                while (RecepteurRoul == 1) {VarRecepteurRoul++;}
                                /* boucle de 9 instruction  : (20 Mhz/4)/9*1ms = 20e6*e-3/36
                                = 20e3/36 = 555.5 */

                while (RecepteurTang == 0 ) {}                  // èChronomtre le PWM éétlcommande
                while (RecepteurTang == 1) {VarRecepteurTang++;}
                                /* boucle de 9 instruction  : (20 Mhz/4)/9*1ms = 20e6*e-3/36
                                = 20e3/36 = 555.5 */

//              while (RecepteurTang == 0 && RecepteurRoul == 0){}
//              while (RecepteurTang == 1 RecepteurRoul ==1)
//-------------------------------------------------------------------------------
// Callage des édonnes
//-------------------------------------------------------------------------------
                VarRecepteurTang = callageRecepteur(VarRecepteurTang);
                VarRecepteurRoul = callageRecepteur(VarRecepteurRoul);
        }



void main(void)
        {
        configuration();
        RSinit();

        float24  CommandeRoul;
        float24  CommandeRoulFiltre;

        int16    VarServoGauche;
        int16    VarServoDroit;



        int16    i;
        int16    inTmp;
        int16    MoyenneGyro;

        float24  vitesseGyro;
        float24  fTmp;
        float24  integGyro;
        float24  VitesseGyro;

        float24  integGyro128;
        float24  moyGyro128;

//-------------------------------------------------------------------------------
// Attente pourê tre en érgime permanent ( Tps de chauffage )
//-------------------------------------------------------------------------------
        for(i=250 ; i != 0 ; i--) {             // Attend 5 sec ( 50*5)
                while (RecepteurTang == 0) {}
                while (RecepteurTang == 1) {}
                }


//-------------------------------------------------------------------------------
// Calcule Moyenne Gyrometre
//-------------------------------------------------------------------------------
        inTmp = 0;
        for (i = 100 ; i!=0 ; i--) {            // éitration < 127, overflow non possible
                inTmp =inTmp + Gyro();
                }

        MoyenneGyro = inTmp ;           // on garde les édonnes avec un facteur *100 ( simulation d'un fixed point )
        integGyro = 0;

        CommandeRoulFiltre = 0;

//-------------------------------------------------------------------------------
//-------------------------------------------------------------------------------
//-------------------------------------------------------------------------------
// Grande boucle
//-------------------------------------------------------------------------------
//-------------------------------------------------------------------------------
//-------------------------------------------------------------------------------

        while(1){
```

```
//-------------------------------------------------------------------------------
// Integration Gyrometre
//-------------------------------------------------------------------------------
        inTmp = 0;
        for (i = 10 ; i!=0 ; i--) {
                inTmp = inTmp + Gyro();      // on divise l'erreur/le bruit
                }
        inTmp = inTmp * 10    ;

                inTmp = inTmp - MoyenneGyro;
                VitesseGyro = (float24) inTmp / 100;

                integGyro = integGyro + VitesseGyro;
                if (integGyro > 500) integGyro = 500;
                if (integGyro < -500) integGyro = -500; // 6400 = 128*500


//-------------------------------------------------------------------------------
// RECEPTION signal RECEPTEUR
//-------------------------------------------------------------------------------

                recoitRecepteur();


//-------------------------------------------------------------------------------
// Commande de Vitesse Angulaire Rouli
//-------------------------------------------------------------------------------
/*


                CommandeRoul = (float24) VarRecepteurRoul / 8.0;     // Influence de la Telecommande
                CommandeRoul = CommandeRoul - VitesseGyro;           // Comparaison au Gyro
                CommandeRoul = CommandeRoul * 12.0;                  // é Sensibilit  du tout

                CommandeRoulFiltre = CommandeRoulFiltre * 4.0;       // Filtre pour supprimer les vibrations
                CommandeRoulFiltre = CommandeRoulFiltre + CommandeRoul;
                CommandeRoulFiltre = CommandeRoulFiltre / 5.0;

*/
//-------------------------------------------------------------------------------
// Mixage simple pour pilotage manuel
//-------------------------------------------------------------------------------


                CommandeRoulFiltre = VarRecepteurRoul;       // Sans mixage


//-------------------------------------------------------------------------------
// Controle d'angle
//-------------------------------------------------------------------------------
/*

                CommandeRoul = integGyro128 ;

//              CommandeRoul = Sature(CommandeRoul);
//              VarRecepteurRoul = VarRecepteurRoul* 2.0;
//              CommandeRoul = CommandeRoul + VarRecepteurRoul;

                CommandeRoulFiltre = CommandeRoul;

//              CommandeRoulFiltre = CommandeRoulFiltre * 4.0;       // Filtre pour supprimer les vibrations
//              CommandeRoulFiltre = CommandeRoulFiltre + CommandeRoul;
//              CommandeRoulFiltre = CommandeRoulFiltre / 5.0;


*/
//-------------------------------------------------------------------------------
// Mixage
//-------------------------------------------------------------------------------
                VarServoGauche = ((int16) CommandeRoulFiltre + VarRecepteurTang) ;
                VarServoGauche = VarServoGauche /2.0;

                VarServoDroit =  ((int16)CommandeRoulFiltre - VarRecepteurTang);
                VarServoDroit = VarServoDroit /2.0;
```

```
//-------------------------------------------------------------------------
// Emission des édonnes
//-------------------------------------------------------------------------


                EmmetServos(VarServoGauche , VarServoDroit);

//              EmmetServos(0,0);


//              inTmp = VarRecepteurTang / 2.0;
//              inTmp = inTmp + 128;
//              RSsend((uns8) inTmp);

//              inTmp = VarRecepteurRoul / 2.0;
//              inTmp = inTmp + 128;
//              RSsend((uns8) inTmp);



//              RSsend( (uns8) VarRecepteurTang%256 );

//              RSsend(Gyro());


//              RSsend((uns8) fTmp);
//              fTmp = integGyro +128;
//              RSsend((uns8) fTmp);


        } // WHILE (1)
        } // Fin Main
```

B.2 Routines Assembleur

```
        LIST      p=16F877          ; é Dfinition de processeur
        #include <p16F877.inc>      ; é Dfinitions des constantes

        __CONFIG    _CP_OFF & _WRT_ENABLE_OFF & _DEBUG_OFF & _CPD_OFF & _LVP_OFF & _BODEN_OFF & _PWRT

;*****************************************************************
;                       MACRO                                    *
;*****************************************************************

BANK0 macro
        bcf     STATUS , RP0  ; passer banque0
        bcf     STATUS , RP1  ; passer banque0
        endm

BANK1 macro
        bsf     STATUS , RP0  ; passer banque1
        bcf     STATUS , RP1  ; passer banque1
        endm

BANK2 macro
        bcf     STATUS , RP0  ; passer banque2
        bsf     STATUS , RP1  ; passer banque2
        endm

BANK3 macro
        bsf     STATUS , RP0  ; passer banque3
        bsf     STATUS , RP1  ; passer banque3
        endm
;*****************************************************************
;                       DEFINE                                   *
;*****************************************************************

#DEFINE Rx  PORTB,7
#DEFINE RTx PORTB,6
#DEFINE Tx  PORTB,5


;*****************************************************************
;               DECLARATIONS DE VARIABLES                        *
;*****************************************************************

UDATA
```

```
status_Save RES 1       ; pour sauvegarder l'état de RP1 et RP0 ( page active )
                        ; le compilateur C èconsidre qu'ils ne changent pas...
w_temp RES 1            ; Sauvegarde du registre W sur 1 octet
status_temp RES 1               ; Sauvegarde du registre STATUS sur 1 octet

RS_buff RES 1           ; RS232 : buffer ( 1 octet ) pour l'envoi ou la reception
RS_count RES 1                  ; RS232 : Nbr de bits de édonnes de la liaison é rie
RS_tmp RES 1            ; RS232 : variable temporaire pour boucle de tempo

Temp RES 1                      ; temp

;*********************************************************************
;                       Transmet                                    *
;*********************************************************************

 CODE


;       routines de gestion RS232C pour Pic 16F84
;       (C)2001 Offset : HTTP://WWW.multimania.com/offset
;
;       commandes :
;
;       RSinit  :  initialisation  du port é rie
;       RSsend : envoi un caractere ( contenu dans w ) sur le port é rie
;       RSrec   : attend et recois un ècaractre du port é rie  ( è caractre çreu dans RS_buff )
;
;
;       NOTA : la reception d'un octet depuis le port é rie se fait avec une boucle .
;              Dans ces conditions , le pic est é bloqu sur l'attente de la reception d'un octet.
;              Si vous utilisez une interruption , vous pouvez appeler RSreci au lieu de RSrec pour
;              recevoir les édonnes ( NDLA : je n'ai pas é test cette methode )
;

RSinit
        GLOBAL RSinit

        movf    STATUS,w
        movwf   status_Save     ; sauvegarde RP0 et RP1

        BANK1
        bcf     Tx              ; Tx en Sortie
        bcf     RTx             ; RTx en Sortie
        bsf     Rx              ; Rx en éentre

        BANK0
        bsf     Tx      ; par defaulté, tat haut
        bsf     RTx     ;  initialisation  de RTx ( ligneà  1 au repos )

        movf    status_Save,w
        movwf   STATUS          ; restaure RP0 et RP1

        return


RSsend
       GLOBAL RSsend

        movwf   RS_buff         ; rs_octet = octetà  envoyer            RX = 0
        movlw   8               ; 8 bits
        movwf   RS_count        ; compteur de bits éenvoys

        movf    STATUS,w
        movwf   status_Save     ; sauvegarde RP0 et RP1
        BANK0

        ; Peut-on Emmetre?

        btfss   Rx              ; si oui : Rx et RTx = 1
        goto    $-1

                ; L'autre ecoute probablement
        bcf     RTx             ; pour synchro
        bcf     Tx              ; Demande d'envoie de donnee
        btfsc   Rx              ; compris ?             RX = 1
        goto    $-1             ; non                   attend RX = 0



RSsendBoucle
        rrf     RS_buff,f       ; on recupere dans C le bità  envoyer
```

```
        btfsc   STATUS,C        ; bità  envoyer = 1 ?
        goto    $+3             ; oui
        bcf     RTx             ; sinon 0
        goto    $+2             ; on continue sur la tempo
        bsf     RTx             ; bità  1
        bsf     Tx              ; bit Envoye
                ; Verifie  si  ligne  a 1

        btfss   Rx              ;
        goto    $-1             ; attend RX = 1
                                                                ;RX = 1

        rrf     RS_buff,f       ; on recupere dans C le bità  envoyer
        btfsc   STATUS,C        ; bità  envoyer = 1 ?
        goto    $+3             ; oui
        bcf     RTx             ; sinon 0
        goto    $+2             ; on continue sur la tempo
        bsf     RTx             ; bità  1
        bcf     Tx              ; bit Envoye
        btfsc   Rx              ;
        goto    $-1             ; attend TX = 0

        decf    RS_count,f      ; on decremente le compteur de bits éenvoys
        decfsz  RS_count,f      ; on part de 8, donc le 2nd tombera sur 0
        goto    RSsendBoucle

        bsf     RTx             ; Retour Etat de base
        bsf     Tx
        btfss   Rx              ;
        goto    $-1             ; attend RTx = 1
                                                                ;RX = 1
        movf    status_Save,w
        movwf   STATUS          ; restaure RP0 et RP1

        return                  ; les 8 bits sont éenvoys et on est ds l'etat de base


RSrecSynchro
        bcf     Tx
        btfss   Rx
        goto    $-1
RSrecbsf
        bsf     Tx

RSrec
        GLOBAL RSrec

        movf    STATUS,w
        movwf   status_Save     ; Sauve RP0 et RP1

        BANK1
        bsf     RTx             ; RTx en entree
        BANK0                   ; selectionne  banque 0

        btfsc   Rx              ; L'autre Envoie une donnee ?
        goto    $-1             ; non , on boucle
                                                                ;TRX = 0
        btfsc   RTx             ; envoie vraiement donnee ?
        goto    RSrecSynchro    ; non, on fait  un cycle

        ; on receptionne donnee

        clrf    RS_buff         ; rs_octet = octetà  recevoir
        movlw   8               ; 8  bits
        movwf   RS_count        ; compteur de bits éenvoys

reception:
        bcf     Tx              ; recu

        btfss   Rx              ; attend donnee
        goto    $-1
                                                                ;TRX = 1
        bcf     STATUS,C
        btfsc   RTx
        bsf     STATUS,C
        rrf     RS_buff,f
        decf    RS_count

        bsf     Tx              ; recu
        btfsc   Rx              ; attend donnee
        goto    $-1
                                                                ;TRX = 0
```

```
        bcf     STATUS,C
        btfsc   RTx
        bsf     STATUS,C
        rrf     RS_buff,f

        decfsz  RS_count
        goto    reception


        bcf     Tx
        btfss   Rx              ; attend donnee et Synchro
        goto    $-1

        btfss   RTx             ; test Synchro
        goto    RSrecbsf

        bsf     Tx                                                      ;TRX = 1
        ; Synchro !!

        BANK1                   ; é slectionner banque 1
        bcf     RTx             ; RTx en sortie
        BANK0                   ; selectionne banque 0


        movf    status_Save,w
        movwf   STATUS          ; restaure RP0 et RP1

        movf    RS_buff,w

        return


;*********************************************************************
; è                     Gyromtre                                    *
;*********************************************************************


Gyro
        GLOBAL Gyro

        movf    STATUS,w
        movwf   status_Save     ; sauvegarde RP0 et RP1

        LOCAL loop              ; declaration de boucle local

        BANK0                   ; passer banque 0
        bcf     ADCON0,CHS0 ; efface le énumro du portà traiter
        bcf     ADCON0,CHS1 ; efface le énumro du portà traiter
        bcf     ADCON0,CHS2 ; efface le énumro du portà traiter

        bsf     ADCON0,ADON ; active le convertisseur
        bsf     ADCON0,GO   ; lance l'acquisition
loop                        ; boucle d attente de fin de convertion
        btfsc   ADCON0,GO   ; test si convertion en cours
        goto    loop        ; si oui, attendre

        movf    status_Save,w
        movwf   STATUS          ; restaure RP0 et RP1

        movf    ADRESL,w    ; éè rcupre édonne
        return




        END;
```

Références

[CAR00] A.Carrière,Y.Hamam, F.Rocaries *Système dynamiques et systèmes asservis : la représentation d'état*, poly Esiee, 2001.

[www00] Site internet sur les parapentes :

[www01] http://www.ntportlibrary.com : Site proposant un driver permettant a matlab d'accéder aux ports du PC

[www02] http://www.mathworks.com : Site de support de Matlab

[www03] http://membres.lycos.fr/afstorm : Plan de l'avion

[www04] http://www.chez.com/aerodynamique : Physique du planneur

;