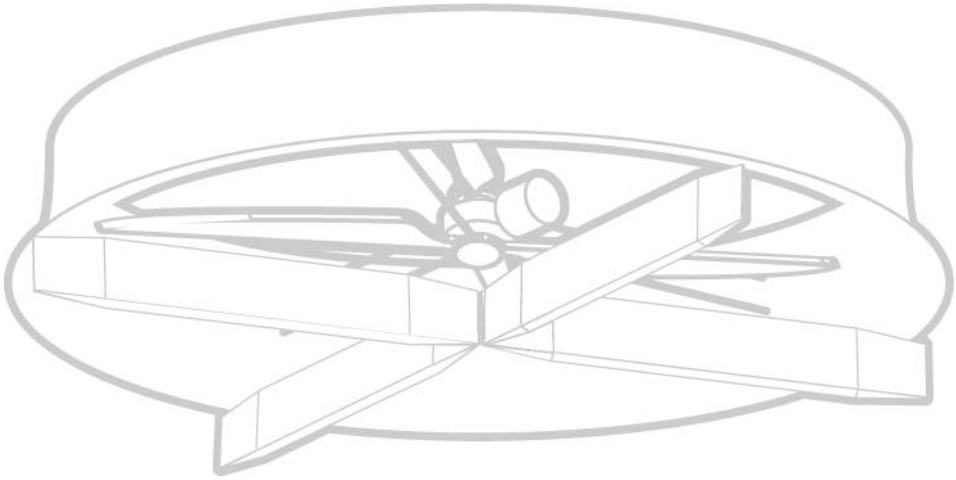
STABILISATION D'UN DRONE BIROTOR CONTRAROTATIF VIA AUTOPILOTE



Département Avionique et Système

Etudiants : Jennifer MAGOT
Julien DRAMET

Tuteur : Yves Brière

ENSICA 2005

TABLE DES MATIERES

I. Présentation générale du dispositif

A. Le concours

Ce PIP se place dans le cadre du concours international de micro drones universitaires auquel participe l'E.N.S.I.C.A.. « Ce concours a pour objet de démontrer la faisabilité technique et l'intérêt opérationnel des drones miniatures utilisés comme aide au fantassin dans sa progression en milieu hostile. L'aide attendue est non agressive : il s'agit en quelque sorte de déporter la vue du fantassin au-delà de son horizon naturel. »
L'objectif est donc la mise au point de drones de taille réduite pour l'observation en milieu urbain. Ces drones doivent être capables d'observer leur environnement, d'acquérir des images de haute qualité et de se déplacer facilement. Il est également demandé de respecter une contrainte de taille : le drone doit être de taille inférieure ou égale à celle d'une sphère de 70 cm diamètre.

Afin de répondre à ce concours, la plate-forme nommée CASPER a été retenue. Il s'agit en fait d'un VTOL, c'est à dire un engin capable de vol stationnaire, d'atterrissage et de décollage vertical. Le drone est architecturé autour de deux rotors contrarotatifs coaxiaux qui lui permettent de se maintenir en l'air tout en évitant les effets gyroscopiques néfastes. Cette architecture est déjà utilisée par Sagem et Bertin. Les grandeurs caractéristiques du drones sont : une masse de 1200 g dont 300 g dédiés à l'avionique embarqué et un diamètre de 60 cm. Il a pour but dans le cadre du concours, de donner les informations visuelles sur le terrain et les différents obstacles ou pièges qui peuvent y être présents. Il est aidé dans sa mission, des informations déduites de la carte incomplète délivrée le jour de l'épreuve et des informations obtenues grâces au second vecteur Pégase.

B. Les études menées

Le vecteur CASPER fut développé en 2003-2004 dans le cadre d'un PIP mené par **Gaël Monnier**, **Édouard Pinel** et **Lionel Renault**. Leur étude avait montré la faisabilité et viabilité d'une telle plate-forme, mais ne comportait que peu de considérations quant à l'automatisation et l'intégration complète du vecteur dans le programme de participation au concours. Une première version de Casper avait été alors réalisée mais aucune avionique ne fut installée à bord. La stabilisation ne fut pas non plus abordée.

Dans ces conditions, d'autres études furent menées au cours des années 2004-2005 : deux PIP et un projet de fin d'étude. Le but du premier PIP, mené par **Samuel Couzinet** et **Sébastien Hélaut**, est de finaliser la conception mécanique de Casper. Pour cela, ils ont résolu les problèmes de la première version, réduit la masse totale et intégré l'avionique. Les travaux menés par **Andreas Goërmer** et notre PIP ont pour but de stabiliser en lacet, roulis et tangage le drone. Les difficultés de stabilisation nous ont contraint à nous limiter au contrôle en lacet dans un premier temps.

C. Dispositif

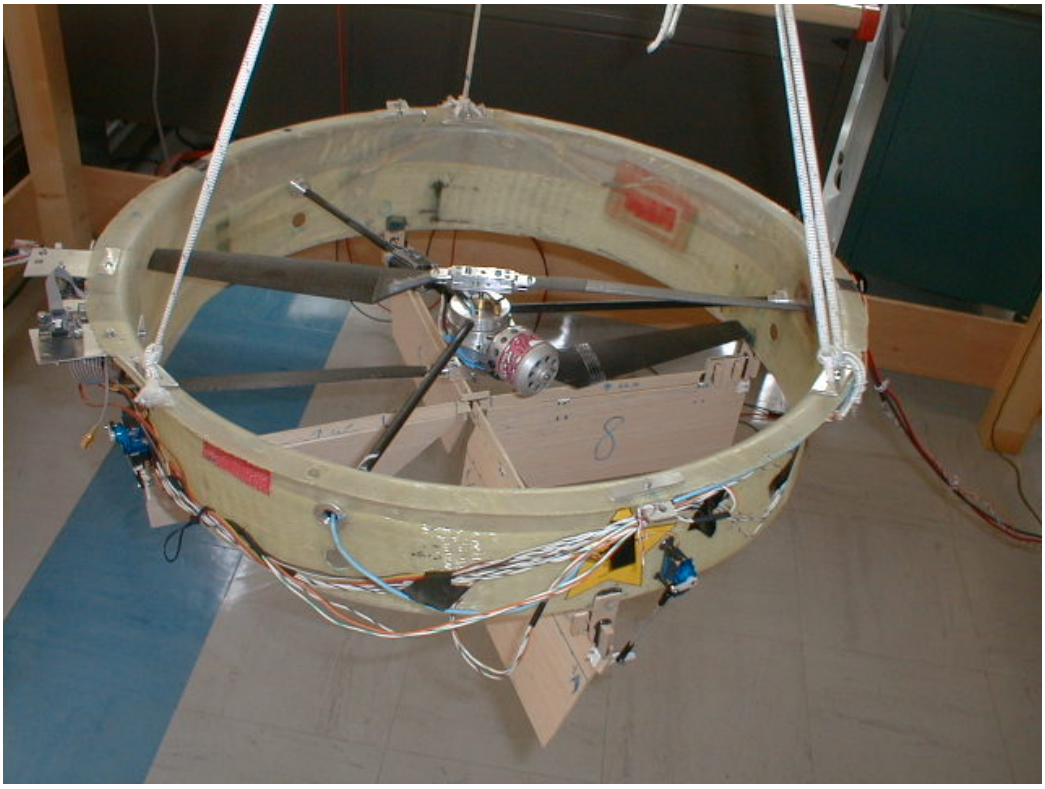


Figure I-1 Le micro drone Casper

Drone :

La plate-forme Casper est architecturée autour de **deux rotors contrarotatifs coaxiaux** qui permettent de fournir la poussée nécessaire au vol.

Sous ces rotors, sont disposés **quatre volets**. Ils permettent de commander les moments de lacet, roulis et tangage de Casper en étant actionnés par **quatre servocommandes**.

Un **moteur électrique** permet d'alimenter les rotors par le biais d'une transmission mécanique. L'énergie nécessaire au moteur est apportée par un jeu de deux batteries pour l'utilisation en vol de Casper. Cependant, pour les essais réalisés lors de la stabilisation, nous avons préféré utiliser un générateur pour fournir l'énergie dont le moteur a besoin. En effet, cette solution était moins contraignante et évitait d'attendre le temps de charge des batteries.

Casper est naturellement instable. Cela présente un avantage : Casper est donc très maniable. Par contre son instabilité implique la réalisation de commande permettant d'asservir son comportement en dynamique. Pour se faire, Casper dispose d'une carte Micropilot intégrée : **la MP2028**. Celle-ci présente plusieurs avantages. Tout d'abord, elle est peu encombrante. Ensuite, elle contient des **capteurs** qui seront utiles dans la conception d'un contrôleur. Elle possède, par exemple des gyromètres et des accéléromètres (en roulis et tangage) un compas et un GPS. Grâce à une radio des échanges d'informations entre la station au sol et la MP2028 sont possibles.

Tous ces éléments sont fixés et protégés par un **carénage** sensé améliorer la poussée crée par les rotors. Dans la version finale de Casper, celui-ci est en fibre de carbone. En effet, le carénage du prototype était en fibre de verre renforcée par du plastique. Cette matière présentait l'avantage d'être plus rigide. Elle permettait donc de réaliser des essais et d'asservir le drone avec moins de risque de rupture. Mais pour la version définitive, la réduction de masse est un objectif principal, ce qui explique l'utilisation de la fibre de carbone.

Les principaux composants sont présentés sur les figures I-2 et I-3 :

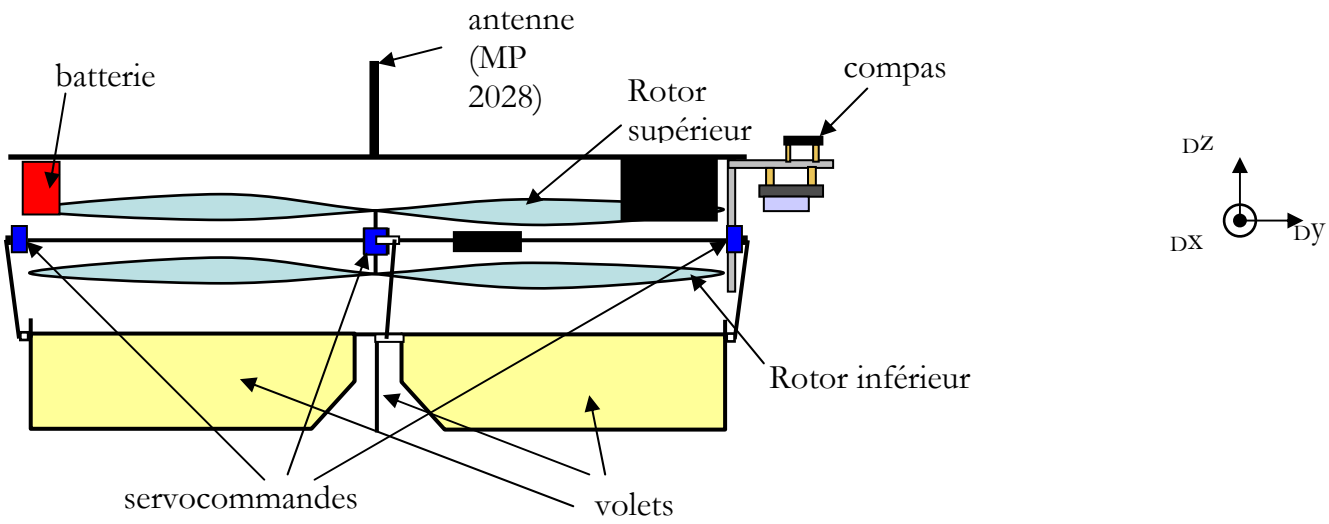


Figure I-2 Les principaux constituants de Casper vu de profil

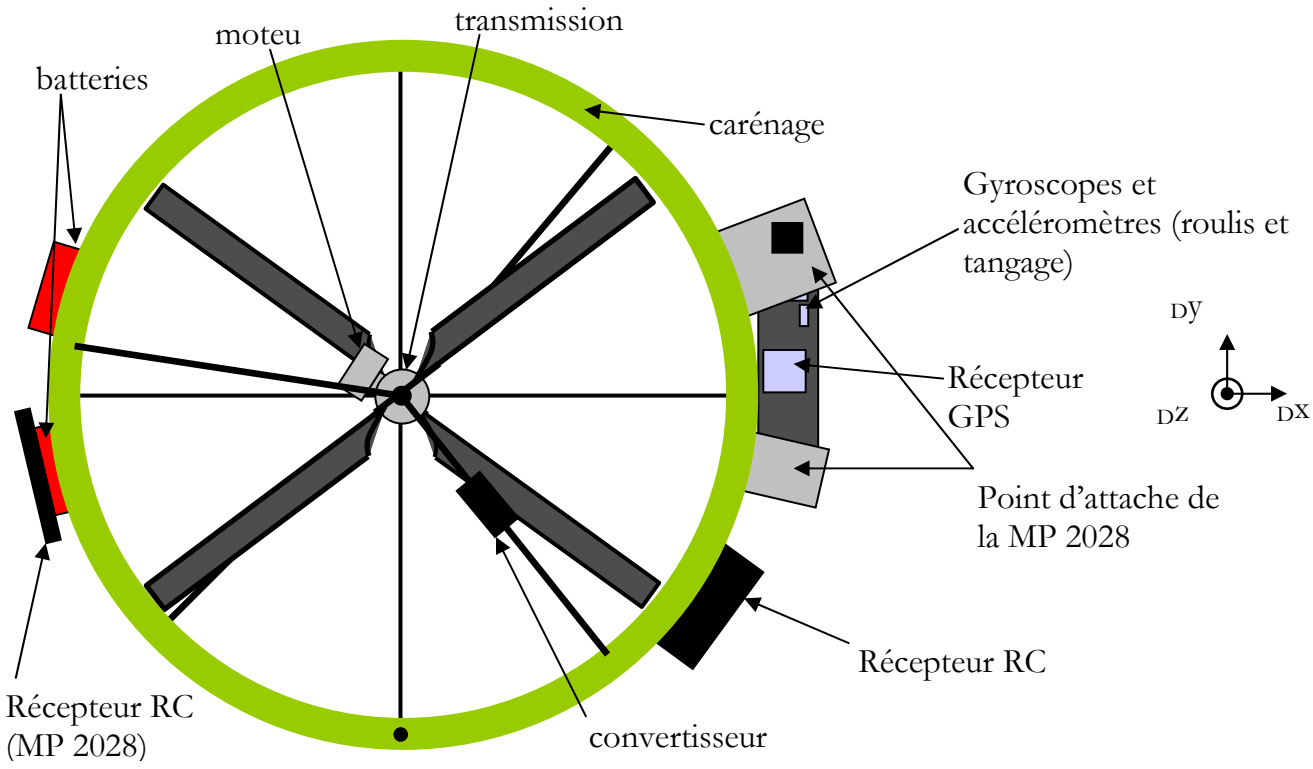


Figure I-3 Les principaux constituants de Casper vu de dessus

Banc d'essai :

Les différentes manipulations effectuées s'appuient sur un banc d'essai constitué par une cage en bois. Les personnes sont protégées par une vitre en plexiglas. Dans la cage le drone est suspendu à la charpente au moyen d'une corde fixée par un crochet. Cette corde se décompose en trois parties étoilées fixées au carénage du prototype. Ce mode de fixation est adapté pour les essais de contrôle en lacet car il permet la rotation de l'appareil suivant l'axe vertical et est peu rigide. Cela nous permet de nous affranchir des questions de dissymétries en masse du drone.
L'essentiel des actionneurs utilisés par le drone sont pilotés par la carte MP2028. La liaison entre la carte pilotée par l'opérateur et le drone s'effectue de trois manières différentes :

-Directement via le port série d'un Pc : ce premier moyen de contrôle permet d'intéragir complètement avec la carte et notamment de « flasher » sa mémoire. Nous pouvons alors mettre à jour les différents paramètres utilisateurs et surtout redéfinir le comportement de la carte.

-A distance via la liaison sans fil autorisée par le module de la carte : cette méthode est utilisée principalement pour le retour des informations en provenance des capteurs et autres instruments. Il permet de modifier les gains de l'asservissement par le biais

d'Horizon sans pour autant avoir à réécrire le code et recompiler. C'est donc un moyen de transmission privilégié pour le réglage de l'asservissement.

-A distance via une télécommande. Ceci a l'avantage de représenter un contrôle plus réaliste de la situation qui devra être mise en œuvre pour le concours. Ainsi nous utiliserons la télécommande pour fournir les consignes en entrée de l'asservissement.

II. Synthèse d'un correcteur

A. Le correcteur RST

L'objectif principal de notre PIP était d'asservir le drone en lacet, roulis et tangage.
L'axe de lacet étant celui le plus naturellement stable, nous nous sommes tout d'abord concentrés sur son asservissement.

En se basant sur le fait que la carte automatique utilisée pour asservir Casper, fonctionne en recevant des signaux discrets, notre première stratégie a été de chercher à stabiliser l'axe de lacet à l'aide d'un PID numérique. En effet, celui-ci paraissait plus adapté au drone et à la carte. Il offrait donc en théorie la possibilité d'obtenir un meilleur asservissement.

Les asservissements numériques n'étant pas au programme des enseignements de première et deuxième années, notre PIP nous a amené à découvrir, par nous même, le fonctionnement de ceux-ci. Nous nous sommes basés, pour comprendre ces nouveaux contrôleurs, sur les cours de Mr Landau.

1. Synthèse d'un PID numérique :

Pour construire un PID numérique, il faut s'appuyer sur le modèle du PID continu qui lui correspond. Le PID numérique résulte de la discrétisation du régulateur PID continu.

Le PID continu s'écrit :

$$HPID(s) = K\left[1 + \frac{1}{T_i.s} + \frac{T_d.s}{1 + \frac{T_d}{N}.s}\right]$$

où K : gain proportionnel
T_i : action intégrale
T_d : action dérivée
T_d/N : filtrage de l'action dérivée

Pour discrétiser, la correspondance suivante est utilisée:

$$s \to \frac{(1-q^{-1})}{Te}$$

On obtient le PID numérique suivant :

$$H_{PID1}(q^{-1}) = \frac{R(q^{-1})}{S(q^{-1})} = K\left[1 + \frac{T_e}{T_i}\cdot\frac{1}{1-q^{-1}} + \frac{\frac{N.T_e}{T_d + N.T_e}(1-q^{-1})}{1 - \frac{T_d}{T_d + N.T_e}.q^{-1}}\right]$$

Le système s'écrit de la manière suivante :

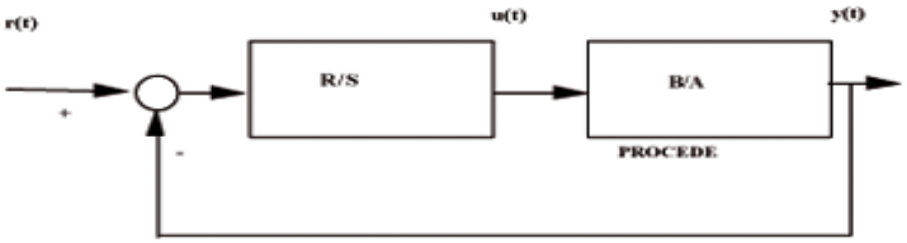


Figure II-1 Schéma de mise en place d'un correcteur PID

Ce qui est équivalent à :

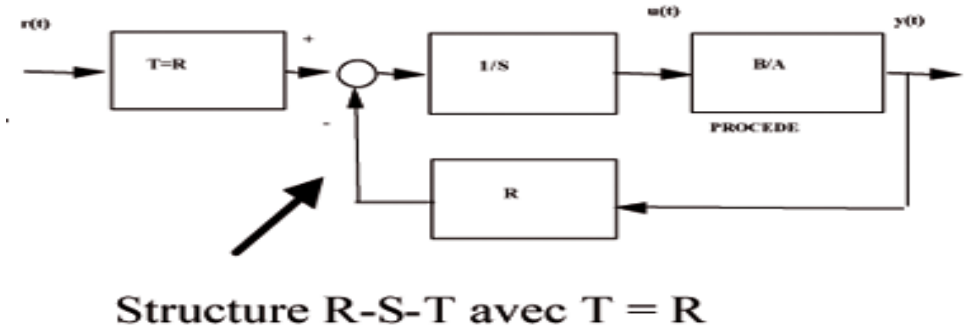


Figure II-2 Structure d'un corrrecteur RST

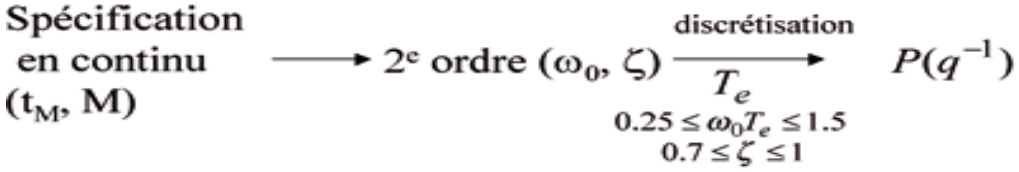
La structure du PID numérique est donc celle d'un correcteur RST. Tout comme pour le PID continu, il faut régler 4 paramètres.

Dans cette structure, le système B/A est le système à corriger. Ce système a été obtenu à partir d'une modélisation discrète du système en un troisième ordre : un premier ordre sert à contrôler l'accélération angulaire. IL est suivi de deux intégrateurs qui permettent d'obtenir successivement la vitesse angulaire et la position angulaire.

La fonction de transfert en boucle fermée s'écrit *:*

$H_{BF}(q^{-1}) = B(q^{-1})R(q^{-1}) / (A(q^{-1})S(q^{-1})+B(q^{-1})R(q^{-1}))$
$= B(q^{-1}) R(q^{-1})/P(q^{-1})$

Où $P(q^{-1})$ spécifie les pôles de la boucle fermée



Pour résumer, nous avons un système de la forme :

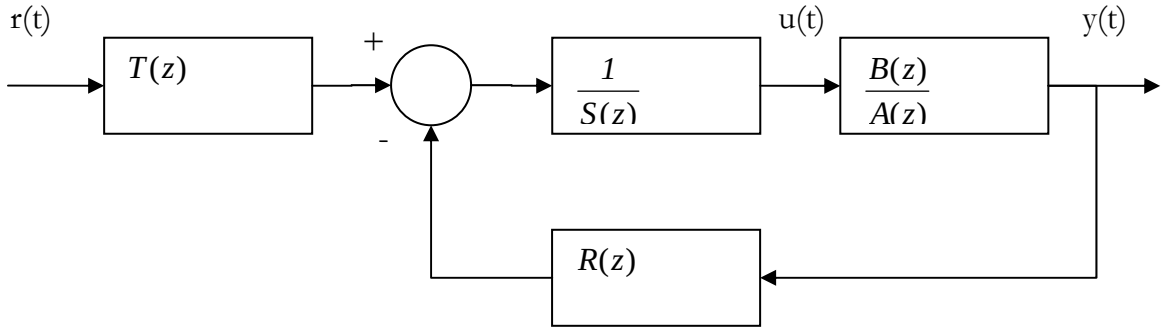


Figure II-3 Contrôleur RST issus d'un PID

Où : $\frac{B(q^{-1})}{A(q^{-1})}$ est le système à asservir

$P(q^{-1})$ spécifie les performances désirées

Pour construire le système il faut donc calculer $R(q^{-1})$ et $S(q^{-1})$. Pour cela, on résout grâce au théorème de Bézout l'équation suivante :

$$P(q^{-1}) = A(q^{-1}).S(q^{-1}) + B(q^{-1}).R(q^{-1})$$

Ainsi, tous les paramètres sont maintenant connus.

A partir de ce modèle, nous avons testé sous Matlab, à l'aide de l'outil Simulink, le correcteur réalisé. Nos multiples essais nous ont amené à conclure que ce correcteur n'était pas adapté à notre système. Il semble, en effet, que ce régulateur ne soit pas applicable pour les systèmes d'ordre supérieur à 3. Or une des caractéristiques de notre système est $\frac{B}{A}$ d'ordre 3.

Nous avons donc cherché un autre régulateur numérique, qui s'utilise avec des systèmes d'ordre 3.

2. Le placement de pôles :

Notre choix s'est donc porté vers la technique du placement de pôles. Elle permet, en effet, de calculer des régulateurs R-S-T pour des systèmes de type $\frac{B}{A}$ sans restriction sur les degrés des polynômes A et B.

La structure de ce régulateur est la suivante :

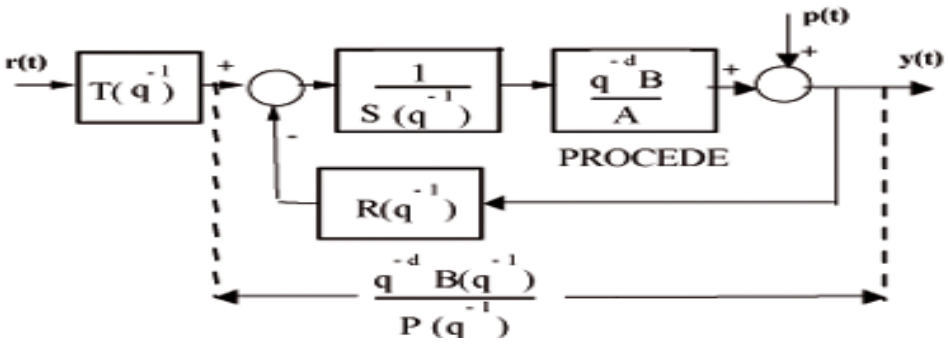


Figure II-4 Correcteur R-S-T placement de pôles

Le calcul du régulateur s'effectue en 2 étapes : le calcul de R et S, obtenus en spécifiant les propriétés de régulation du système, puis le calcul de T, obtenu en spécifiant les propriétés en poursuite du système.

a) Le régulateur : calcul de R et S

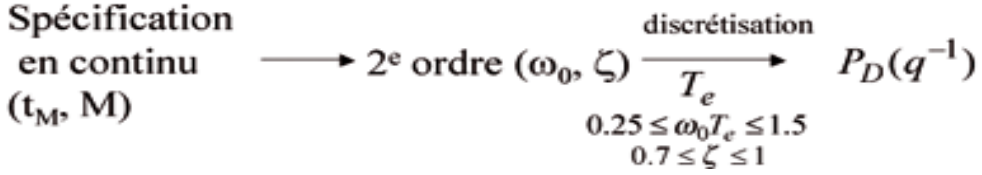
La fonction de transfert en boucle fermée est :

$$H_{BF}(q^{-1}) = \frac{q^{-d}T(q^{-1})B(q^{-1})}{A(q^{-1})S(q^{-1}) + q^{-d}R(q^{-1})B(q^{-1})} = \frac{q^{-d}T(q^{-1})B(q^{-1})}{P(q^{-1})}$$

Avec d le retard du système

Là encore, P (q^{-1}) définit les pôles de la boucle fermée. Il peut se décomposer en :

$P(q-1) = P_D(q^{-1}) \cdot P_F(q^{-1})$ Où P_D : définit les pôles dominants
P_F : définit les pôles auxiliaires



Les pôles auxiliaires sont introduits pour la robustesse et doivent être choisis plus rapides que les pôles dominants.

Comme pour le PID numérique, pour construire le régulateur à placement de pôles, il faut calculer R et S. Pour cela, on résout l'équation de Bézout suivante où A et B sont premiers entre eux :

$$P(q^{-1}) = A(q^{-1})S(q^{-1}) + q^{-d}B(q^{-1})R(q^{-1})$$

Les deux inconnues R et S contiennent une partie fixe, que nous devons fixer :

$$R(q^{-1}) = R'(q^{-1})H_R(q^{-1}) \qquad S(q^{-1}) = S'(q^{-1})H_S(q^{-1})$$

Où H_R et H_S sont les deux polynômes pré spécifiés :

H_S permet d'annuler l'erreur statique.
H_R laisse passer le signal sans le perturber aux fréquences correspondant à ses pôles..

H_R et H_S définissent la capacité qu'a le système à suivre une dynamique imposée. Il s'agit donc de la définition des performances en poursuite.

Pour calculer R' et S', on résout par le théorème de Bézout l'équation suivante :

$$P(q^{-1}) = A'(q^{-1}).S'(q^{-1}) + q^{-d}.B'(q^{-1}).R'(q^{-1})$$

Où
A' = A H_S
B' = B H_R

b) Les performances en poursuite : calcul de T

A ce stade, nous avons défini un régulateur. Mais le placement de pôles permet également de définir des performances en poursuite. Pour cela, nous utilisons le modèle suivant :

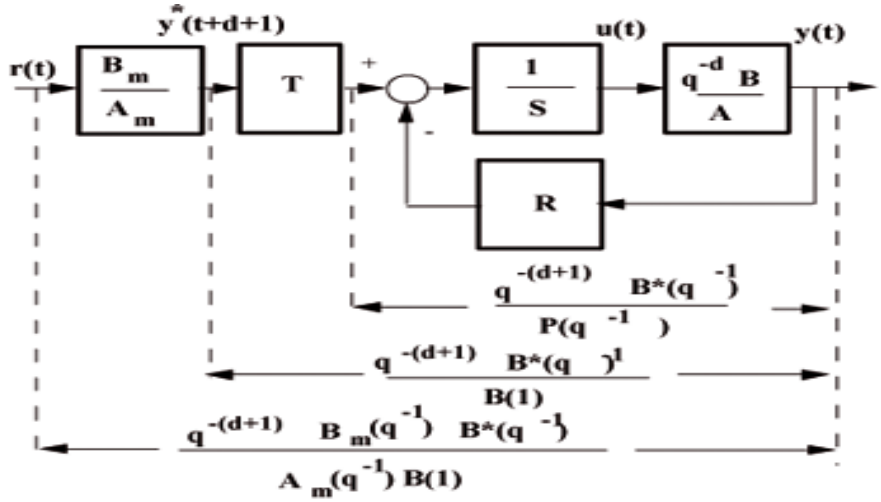


Figure II-5 Modèle R-S-T en poursuite

B_m / A_m permet de définir les performances en poursuite.
T permet de définir la dynamique en poursuite et corrige B_m / A_m
On le détermine par :

$$T(q^{-1}) = G.P(q^{-1}) \qquad G = \begin{cases} \dfrac{1}{B(1)} \; si B(1) \neq 0 \\ 1 si B(1) = 0 \end{cases}$$

B. La correction par retour d'état

Pour présenter le fonctionnement du régulateur par retour d'état, nous nous appuierons sur les cours de monsieur Bordeneuve, dispensés en deuxième année à l'ENSICA et de son polycopié intitulé « COMMANDE DES SYSTEMES LINEAIRES ».

Le régulateur à retour d'état peut être interprété comme une généralisation de la commande par PID. Il présente l'avantage de modifier la dynamique du système sous les seules conditions de gouvernabilité du système et de mesurabilité des états. Le principe général repose sur une contre-réaction constituée par un retour linéaire du vecteur d'état.

1. Conditions :

- Le système doit être gouvernable : les états peuvent être modifiés par action sur la commande.
- Les états doivent être mesurables : cela signifie que, grâce à des capteurs en général, on a accès aux variables d'état.

2. Equations d'état dans la base canonique :

Le comportement entrée-sortie du système s'écrit :

$$\dot{x}(t) = A.x(t) + B.u(t)$$
$$y(t) = C.x(t)$$

La commande est calculée par différence entre la consigne e (t) et une combinaison linéaire de tous les états :

$u(t) = e(t) - K.x(t)$ Où K est un vecteur de gains constants que nous devons calculer selon la dynamique que l'on souhaite obtenir.

Toute la dynamique du système est contenue dans les coefficients de l'équation caractéristique. La base la mieux adaptée pour travailler est donc la base compagne de commande. Soit M la matrice de passage de la base initiale à la base canonique de commande.

On transforme les équations dans la nouvelle base. On obtient :

$$\dot{\tilde{x}}(t) = \tilde{A}.\tilde{x}(t) + \tilde{B}.u(t)$$
$$y(t) = \tilde{C}.\tilde{x}(t)$$

où $x = M.\tilde{x}$

$$\tilde{A} = M^{-1}.A.M$$
$$\tilde{B} = M^{-1}.B$$
$$\tilde{C} = C.M$$

Avec $\tilde{K} = K.M$

3. Calcule de K :

On se fixe un polynôme caractéristique en boucle fermée :

$$\psi(p) = \det(p.I - \tilde{A} + \tilde{B}.\tilde{K}) = p^n + \alpha_{n-1}.p^{n-1} + \ldots + \alpha_0$$

Or après changement de base, le système devient :

$$\dot{\tilde{x}}(t) = \begin{pmatrix} 0 & 1 & 0 & \cdots & 0 \\ & 0 & 1 & & \vdots \\ \vdots & & \ddots & \ddots & 0 \\ 0 & & & 0 & 1 \\ -a_0 & -a_1 & & \cdots & -a_{n-1} \end{pmatrix}.\tilde{x}(t) + \begin{pmatrix} 0 \\ \vdots \\ \vdots \\ 0 \\ 1 \end{pmatrix} u(t)$$

En identifiant les deux matrices, on obtient les valeurs de $\tilde{K}$:

$$\tilde{K} = \begin{pmatrix} \tilde{k}_0 & \cdots & \tilde{k}_{n-1} \end{pmatrix}$$

$$\tilde{A} - \tilde{B}.\tilde{K} = \begin{pmatrix} 0 & 1 & 0 & \cdots & 0 \\ & 0 & 1 & & \vdots \\ \vdots & & \ddots & \ddots & 0 \\ 0 & & & 0 & 1 \\ -(a_0 + \tilde{k}_0) & -(a_1 + \tilde{k}_1) & & \cdots & -(a_{n-1} + \tilde{k}_{n-1}) \end{pmatrix} = \begin{pmatrix} 0 & 1 & 0 & \cdots & 0 \\ & 0 & 1 & & \vdots \\ \vdots & & \ddots & \ddots & 0 \\ 0 & & & 0 & 1 \\ -\alpha_0 & -\alpha_1 & & \cdots & -\alpha_{n-1} \end{pmatrix}$$

IL vient : Pour tout i de 0 à n-1, on a : $\tilde{k}_i = \alpha_i - a_i$

4. Correcteur :

$K = \tilde{K}.M^{-1}$

Le correcteur sera : $$u(t) = e(t) - \sum_{i=0}^{n-1} k_i.x_i(t)$$

III.Mise en œuvre pratique

A. Problématique d'un VTOL

1. Description physique

La plate-forme Casper est un VTOL (Vertical Take Off and Landing). Cela signifie que Casper est un engin capable de vol stationnaire, d'atterrissage et de décollage vertical.

Comme nous l'avons vu précédemment, Casper possède quatre volets, disposés à 90° les uns des autres et commandés indépendamment, qui permettent une commande des moments du micro drone. Nous allons présenter les moments que l'on obtient selon les positions des volets.

Les volets agissent sur le flux d'air en le déviant. Comme pour une aile d'avion, la vitesse du fluide sur le volet crée une dépression à l'extrados et une surpression à l'intrados. Ainsi, une force est créée par le fluide sur le volet.
Or la force exercée par le fluide sur le volet dépend de l'angle du volet. Cette force génère un moment par rapport au centre de gravité. Celui-ci provoque la rotation du drone.

Ainsi lorsque l'on braque deux volets disposés à 180° dans le même sens, le drone se met en rotation suivant l'axe de ces volets (roulis ou tangage). Si en revanche, nous braquons tous les volets dans le même sens, nous obtenons de la part du drone un mouvement de lacet. Ces résultats serviront de base pour le contrôle en rotation du drone.

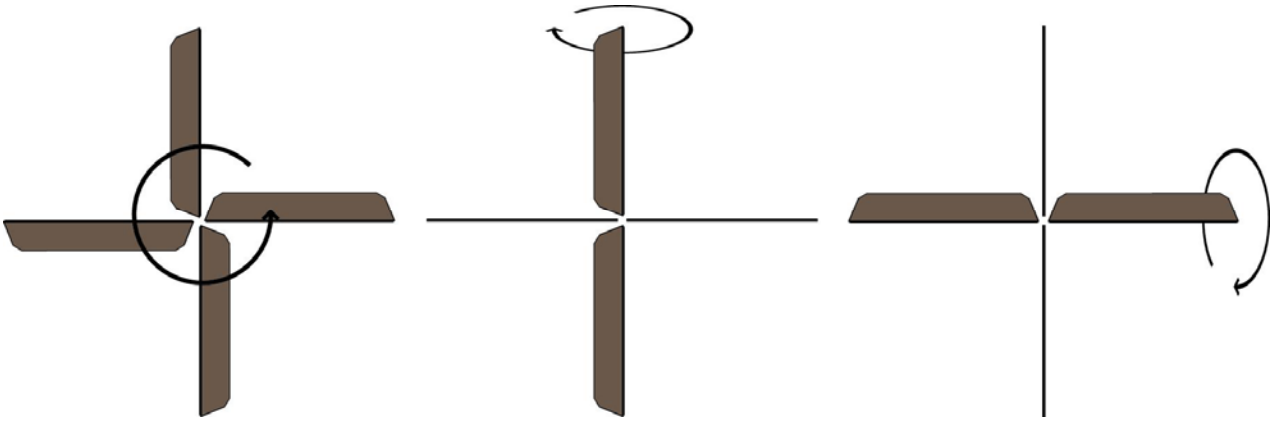


Figure III-1 Effets des volets sur le mouvement en rotation du drone

2. Modèle

Les quatre volets de Casper créent chacun une force et donc un moment au centre de gravité.
La force crée par chaque volet est :

$$Fvolets = \frac{1}{2}.\rho.(Vfluide/volet)^2.Cz.S$$

Où :

- C_z : coefficient aérodynamique de portance dépendant de l'angle d'incidence des volets
- S : surface d'un volet
- V : vitesse du fluide par rapport au volet
- r : masse volumique de l'air

Le moment aérodynamique créé par les quatre volets est donc :

$$M_{4volets} = 4.l.F_{volet}$$

Ces hypothèses permettent donc de modéliser le drone sous forme d'un premier ordre et de deux intégrateurs.

Etant donné la symétrie de révolution du drone il est possible de considérer que la matrice principale d'inertie est diagonale. En effet un calcul à partir du modèle Catia de cette matrice d'inertie nous fournit le résultat suivant :

$$J = \begin{bmatrix} 1.27 \cdot 10^{-4} & -1.25 \cdot 10^{-5} & 9.27 \cdot 10^{-8} \\ -1.25 \cdot 10^{-5} & 1.7 \cdot 10^{-4} & 6.28 \cdot 10^{-6} \\ 9.27 \cdot 10^{-8} & 6.28 \cdot 10^{-6} & 1.14 \cdot 10^{-4} \end{bmatrix} \quad \text{en } Kg.m^2$$

Si on néglige les effets dus au couplage des forces de Coriolis, il est possible d'écrire l'équation des moments suivante :

$$\begin{pmatrix} J_{xx} \cdot \dot{\omega}_x \\ J_{yy} \cdot \dot{\omega}_y \\ J_{zz} \cdot \dot{\omega}_z \end{pmatrix} = \sum {}_D \vec{M}^{cg}$$

Où :

- J_{xx}, J_{yy}, J_{zz} Sont les moments principaux d'inertie.
- $\dot{\omega}_x, \dot{\omega}_y, \dot{\omega}_z$ Sont les accélérations angulaires.
- $\sum {}_D \vec{M}^{cg}$ Est la somme des moments calculés au centre de gravité.

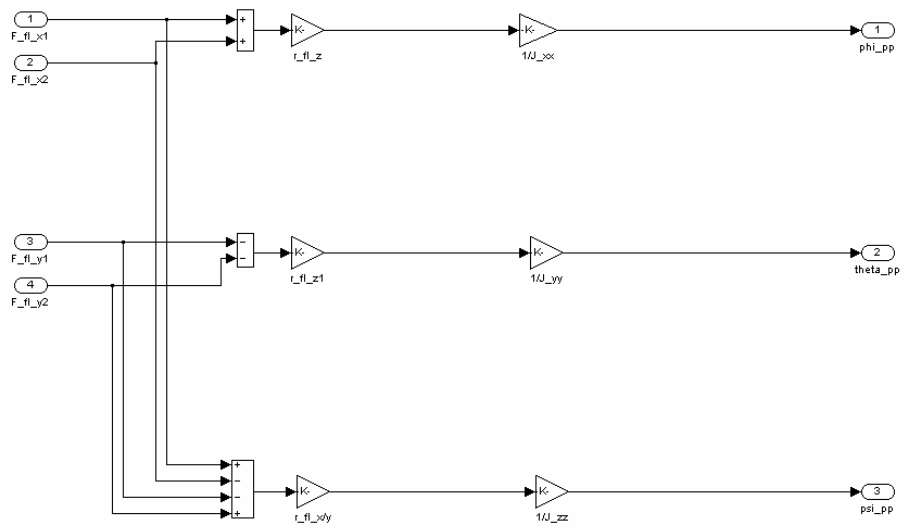


Figure III-2 Relation entre les forces et les vitesses angulaires (Schéma fourni par Andréas)

Il convient de compléter notre modélisation par l'influence de la poussée produite par le moteur. En effet nous avons vu que les efforts produits au niveau des volets dépendent de la vitesse de l'air qui les entoure et donc de la poussée produite par le moteur.
Cependant il n'existe pas de relation simple entre la poussée et le moment produit en réalité du fait de nombreuses non linéarités.

Le micro drone Casper doit être stabilisé sur ces 3 axes de rotation. Etant donné sa forme cylindrique, il est possible de dissocier les axes de rotation en 2 catégories :

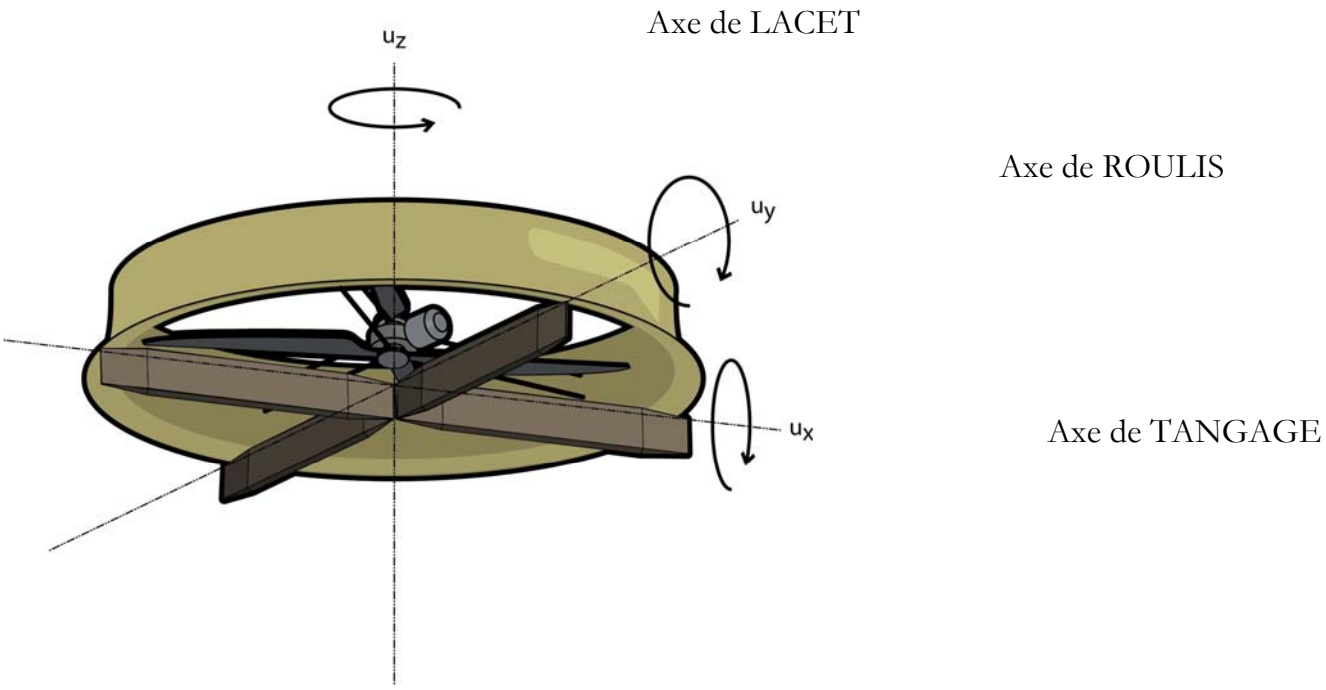


Figure III-3Axes principaux du drone Casper.

L'axe de lacet :

Il fait intervenir les quatre volets. Cet axe est assez peu soumis aux problèmes d'équilibrage du drone. En effet, c'est un axe de symétrie pour le drone ; on peut donc considérer que la matrice d'inertie est diagonale.

Les axes de tangage et de roulis :

Ils font intervenir deux volets. Etant donné la symétrie du drone, il est possible de considérer dans un premier temps que la stabilisation sur ces deux axes est identique. Cependant il est très important de remarquer que l'avionique embarquée ne peut être disposée de manière uniforme sur tout le carénage du micro drone. Or cette répartition influe énormément sur l'équilibre du drone et donc sur la stabilisation en lacet et en roulis

Ainsi dans un premier temps, il est plus facile de réaliser l'asservissement en lacet, puis en capitalisant l'expérience acquise de réaliser, par la suite, les asservissements en tangage et en roulis.

3. Identification.

Afin de pouvoir effectuer des simulations convenables, il est nécessaire de modéliser au mieux le système. Notre première démarche fut donc de réaliser une étude dynamique du drone afin de comprendre au mieux sa mécanique. Cependant, il faut

rappeler que Caper est avant tout un prototype qui c'est peu à peu éloigné de sa modélisation initiale. Ainsi les valeurs des différentes inerties que nous avons pu calculer sont en fait assez éloignées des valeurs réelles. Cela est notamment du à l'avionique et aux modifications de structures et de matériaux qui eurent lieu entre sa conception et sa fabrication.

La meilleure manière d'obtenir un modèle convenable du système était donc de réaliser une identification automatique. L'identification du drone a été réalisée par Andréas Goërmer, qui réalisa au cours de l'année 2005 son projet de fin d'étude au Département Avionique et Systèmes de l'ENSICA.

Le principe de cette identification est l'étude de la réponse du système à un signal PRBS (Pseudo Random Binary Signal). Ce signal prend deux valeurs de manière aléatoire. Il est ainsi possible d'exciter le système avec de multiples fréquences ce qui rend alors possible son identification par l'étude de sa réponse. La carte réalisant l'avionique permet la création de fichiers de télémétrie indiquant l'état de ces capteurs au cours du temps et notamment les valeurs à l'aide des données du cap et de la vitesse angulaire ainsi que de la commande fournie aux servomoteurs. L'avantage de cette démarche est qu'elle permet de s'affranchir des problèmes d'unités puisque la commande est à priori adimensionnée. Ainsi dans un premier temps, les fonctions de transfert de la vélocité angulaire par rapport à la commande des servomoteurs ont été déterminées $(\frac{\dot{\psi}}{u})$:

Forme factorisée	Forme polynomiale	Particularités du modèle
$\frac{0.034}{(1-0.66\cdot q^{-1})\cdot(1-0.9\cdot q^{-1})}$	$\frac{0.034}{1-1.56\cdot q^{-1}+0.594\cdot q^{-2}}$	Model discret d'échantillonnage 5 Hz
$\frac{0.037937\cdot(1-0.693\cdot q^{-1})}{(1-0.9826)\cdot(1-0.9331)}$	$\frac{0.03794-0.02629\cdot q^{-1}}{1-1.916\cdot q^{-1}+0.9168\cdot q^{-2}}$	Model discret d'échantillonnage 30 Hz
$\frac{0.10034\cdot(s+1.091)}{(s+2.078)\cdot(s+0.5268)}$	$\frac{0.10034\cdot s+1.094}{s^2+2.604\cdot s+1.094}$	Model continu
$A=\begin{bmatrix}0 & 1 & 0\\ 0 & 0 & 1\\ 0 & -1.094 & -2.604\end{bmatrix}$ $B=\begin{bmatrix}0\\0\\1\end{bmatrix}$ $C=\begin{bmatrix}5.472 & 1.047 & 0.064\end{bmatrix}$ $D=0$		Représentation d'état du model continu

Figure III-4 Récapitulatif des modèles, extrait du rapport d'étude d'Andréas Goërmer

Cependant le paramètre que nous souhaitons asservir est ψ et non $\dot{\psi}$. L'identification de la fonction de transfert de l'angle de cap par rapport à la consigne de tension fournit le résultat suivant :

$$H_{BO}(z^{-1}) = \frac{0.00514}{1 - 0.5175 \cdot z^{-1}}$$ à une fréquence de 5Hz

Le modèle continu est lui :

$$H_{BO}(s) = \frac{1.003 \cdot s + 10.94}{s^3 + 2.604 \cdot s^2 + 1.094 \cdot s}$$

Ce sont ces deux modèles que nous utiliserons par la suite pour calculer nos asservissements.

B. Implémentation aspect temps réel/simulation

1. En simulation

Afin de mettre en pratique la théorie de la correction RST, nous avons utilisé le logiciel Matlab. Celui-ci nous a permis de tester les différents correcteurs que nous avions déterminés. Nous avons procédé en deux étapes. D'abord nous avons écrit un code Matlab sous forme de fichier .m qui permet d'obtenir l'asservissement désiré à partir de la donnée de la dynamique en boucle fermée. Puis nous avons ensuite modélisé l'asservissement à l'aide de l'outil Simulink.

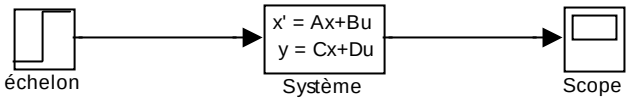


Figure III-5 Système sous sa représentation d'état

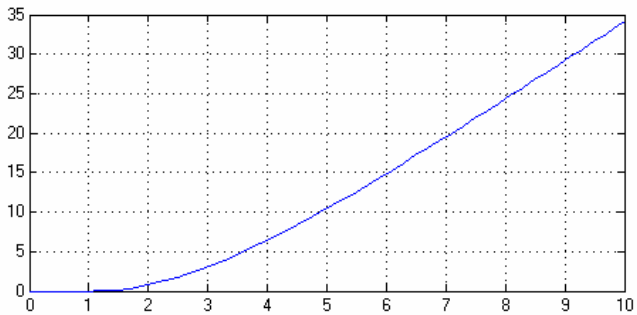


Figure III-6 Réponse en boucle ouverte

Sur la figure II.5 apparaît le principe utilisé pour tester notre correcteur. Le modèle est simulé par la fonction de transfert déterminée par identification. Etant donné que l'on cherche à corriger l'angle de lacet et non son accélération, le modèle est suivi de deux intégrateurs. Les retours sur l'angle et l'accélération angulaire sont corrigés par les gains R2 et R1 qui sont des paramètres importants de notre correcteur. La principale difficulté de la modélisation est le choix de la dynamique en boucle fermée du système. En effet si théoriquement la simulation permet d'obtenir toute dynamique désirée, en pratique plus les performances désirées sont importantes et plus les résultats seront éloignés de la théorie. En particulier, le système que nous étudions souffre de nombreuses non linéarité difficilement simulables, et dont nous ne tenons pas compte lors du calcul de l'asservissement.

La simulation est effectuée à partir d'un échelon unitaire représentant une consigne d'angle en lacet. Cela est assez proche de ce qui sera effectivement demandé lors du pilotage du drone, même si pour des raisons de confort de pilotage, nous verrons lors de la mise en pratique qu'il est parfois préférable de commander une vitesse angulaire.

Choix de la dynamique :

- Amortissement : 0.9
- **Pulsation propre : 1** soit une fréquence propre de 0.16 Hz

L'amortissement devrait normalement être suffisamment élevé pour éviter tout dépassement de la consigne. En effet Casper doit théoriquement supporter une caméra dédiée à l'acquisition stabilisée de la cible et à faciliter le pilotage. Par conséquent les oscillations sont à proscrire ; nous avons cependant choisi une valeur d'amortissement inférieur à un car la pratique montre que le comportement de Casper est en réalité plus amorti qu'il ne l'est en pratique. La pulsation a été choisie assez faible compte tenue de la faible réactivité de l'engin, et afin d'être en première expérimentation sous dimensionnée.

2. Contrôleur RST simple

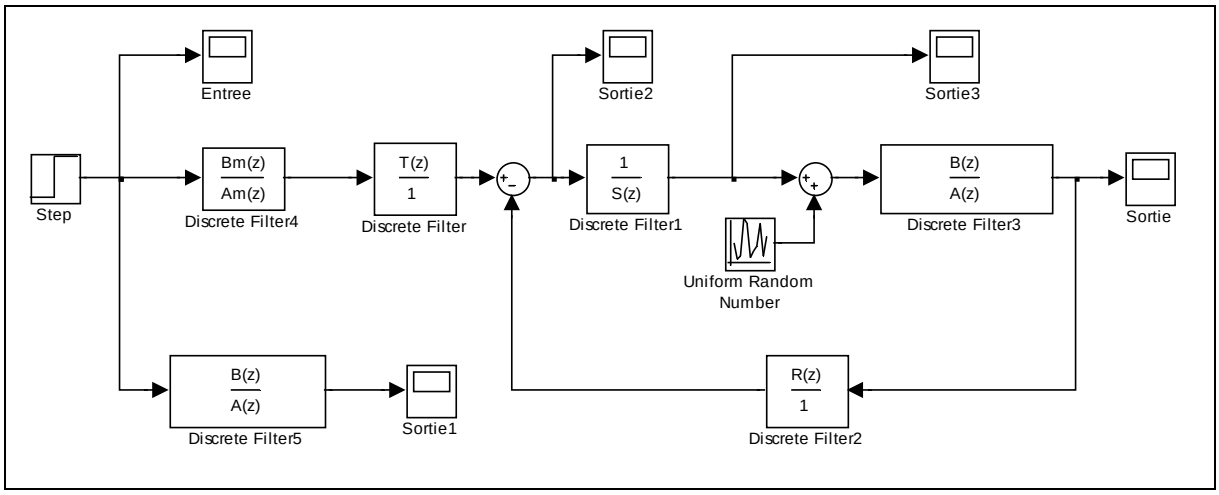


Figure III-7Schéma de l'asservissement RST sipmple

Evolution du système en fonction du temps :

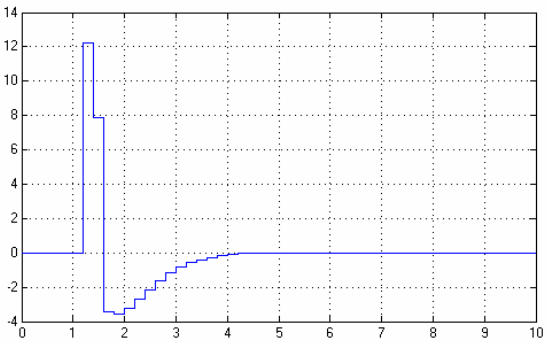


Figure III-8 Consigne

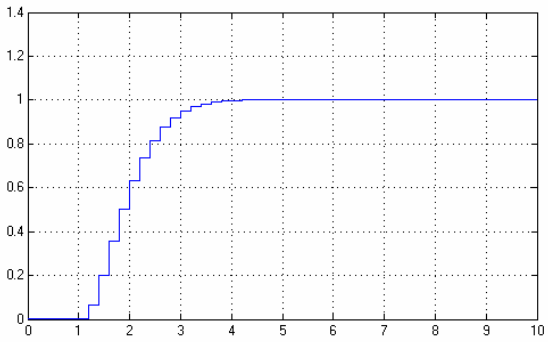


Figure III-9 Réponse temporelle

Paramètres de l'asservissement :

Système :

$Bm = 0 \quad 0.0630 \quad 0.0495$

$Am = 1.0000 \quad -1.3742 \quad 0.4868$

$P = 1.0000 \quad -0.7371 \quad 0.1653$

Correcteur :

$R = 164.0651 \quad -36.1663 \quad -106.6348 \quad 77.8192 \quad -15.7773$

$S = 0.1567 \quad -0.1567$

$T = 194.5525 \quad -143.4059 \quad 32.1593$

Lors de cette simulation nous avons été confrontés à certaines difficultés dues au fait que Matlab ne parvenait pas à résoudre notre système. Etrangement ce problème survenait en régime établi. Afin de contourner cette difficulté, nous avons introduit dans notre simulation une perturbation aléatoire d'amplitude très faible (0.01) au regard des autres signaux. Ce type de perturbation permet de ne pas trop modifier le comportement dynamique tout en empêchant la sortie de devenir constante. Ainsi nous avons pu mener à bien notre simulation.

Le modèle précédent a plusieurs inconvénients :

D'une part, il repose entièrement sur la connaissance de l'angle de cap du drone. Il est donc nécessaire de connaître avec précision cet angle. Or, d'après les spécifications de la carte, la valeur de cet angle est mise à jour à une fréquence de 5Hz. Comme notre asservissement numérique fonctionne à la même fréquence, il est peu performant.

D'autre part, le calcul du correcteur est entièrement fondé sur la valeur de l'angle de cap du drone. Or cette valeur n'est pas toujours très pertinente. C'est pourquoi il peut être intéressant d'utiliser d'autres paramètres mis à disposition par la carte.

Classiquement, le modèle d'un contrôleur RST permet d'obtenir la loi de commande suivante :

$$S(q^{-1}) \cdot u(t) = T(q^{-1}) \cdot e(t) - R(q^{-1}) \cdot \psi(t)$$

$$\text{Avec } R(q^{-1}) = r_0 + r_1.q^{-1} + \cdots + r_n.q^{-n}$$

Il est possible de factoriser $R\,(q^{-1})$ de manière à faire intervenir la dérivée de $\psi(t)$. En effet, en effectuant la division euclidienne de $R\,(q^{-1})$ par $(1-q^{-1})$ on obtient :

$$R(q^{-1}) = R1(q^{-1}) + R2(q^{-1}) \cdot (1 - q^{-1})$$

Or, pour un signal discret, la multiplication par $(1-q^{-1})$ équivaut à dériver le signal. Par conséquent :

$$R(q^{-1}) = \left[R1(q^{-1}) + R2(q^{-1}) \cdot (1 - q^{-1})\right] \cdot \psi(t) = R1(q^{-1}) \cdot \psi(t) + R2(q^{-1}) \cdot \dot{\psi}(t)$$

3. Contrôleur RST avec retours

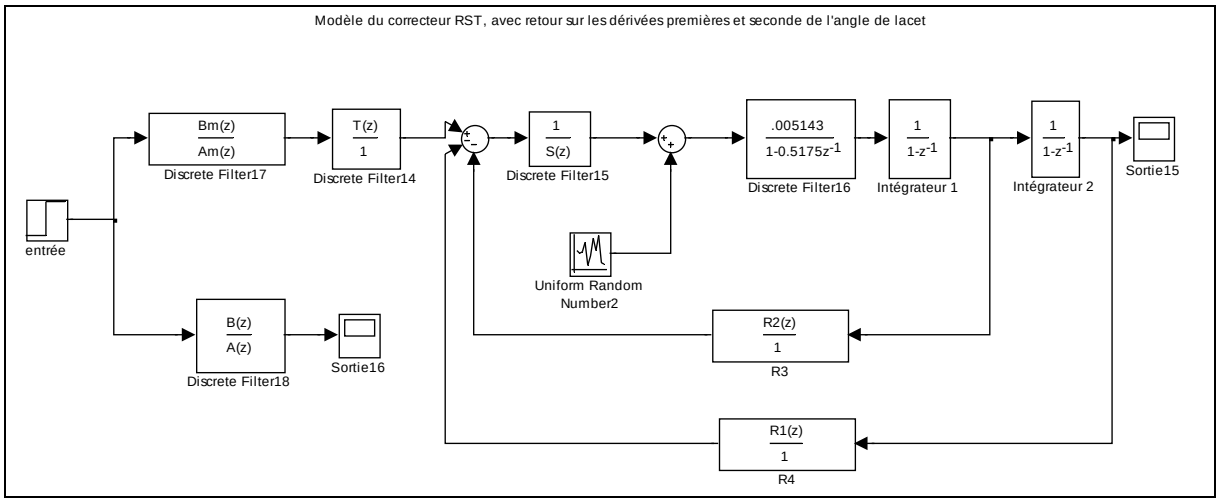


Figure III-10 Schéma de l'asservissement RST avec dissociation R1 et R2

Evolution du système en fonction du temps :

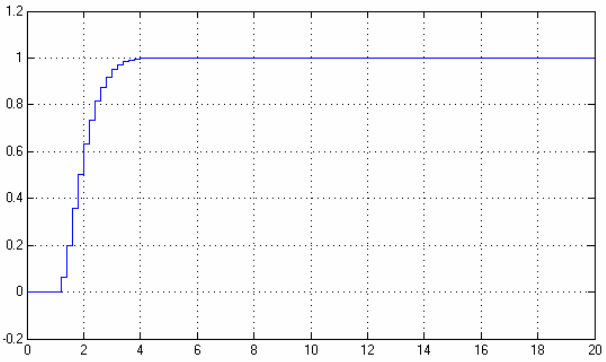


Figure III-11 Sortie

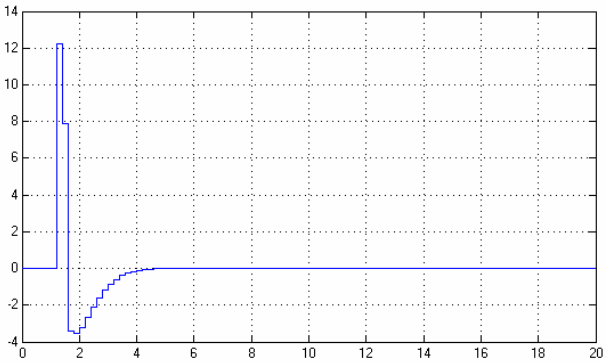


Figure III-12 Consigne

Paramètres de l'asservissement :

Système

$Am = 1.0000 \quad -1.3742 \quad 0.4868$

$Bm = 0 \quad 0.0630 \quad 0.0495$

$P = 1.0000 \quad -0.7371 \quad 0.1653$

Correcteur

$R1 = 83.3059$

$R2 = 80.7591 \quad 44.5929 \quad -62.0420 \quad 15.7773$

$S = 0.1567 \quad -0.1567$

$T = 194.5525 \quad -143.4059 \quad 32.1593$

4. Contrôleur avec retour d'état

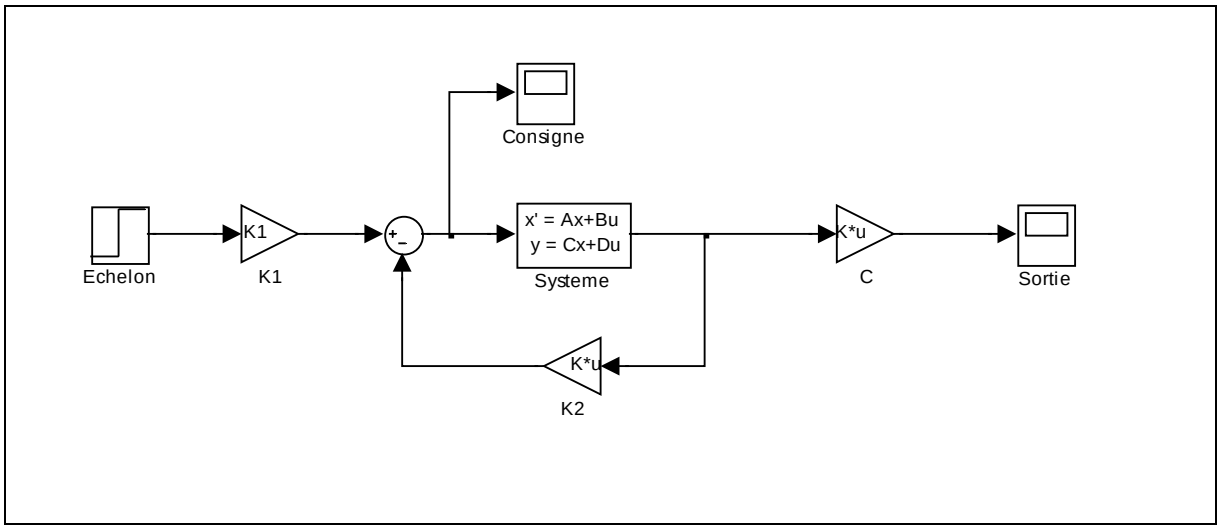


Figure III-13 Schéma de l'asservissement par retour d'états

Evolution du système en fonction du temps :

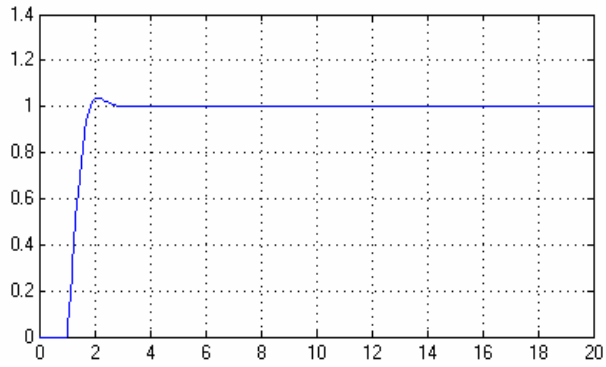


Figure III-14 Sortie

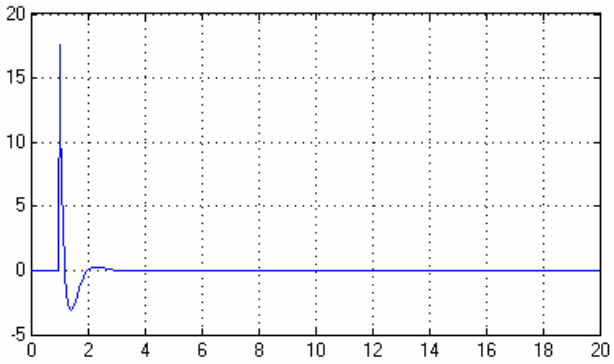


Figure III-15 Consigne

Paramètres de l'asservissement :

Système :

$A = [0\,1\,0;0\,0\,1;0\,-1.095\,-2.604]$

$B = [0;0;1]$

$C = [5.472\,1.047\,0.064]$

$D = [0]$

Correcteur :

$K2 = 96.0000 \quad 48.5050 \quad 8.9960$

$K1 = 17.5439$

5. Implémentation temps réel

Schéma de l'asservissement implémenté pour le contrôle du lacet :

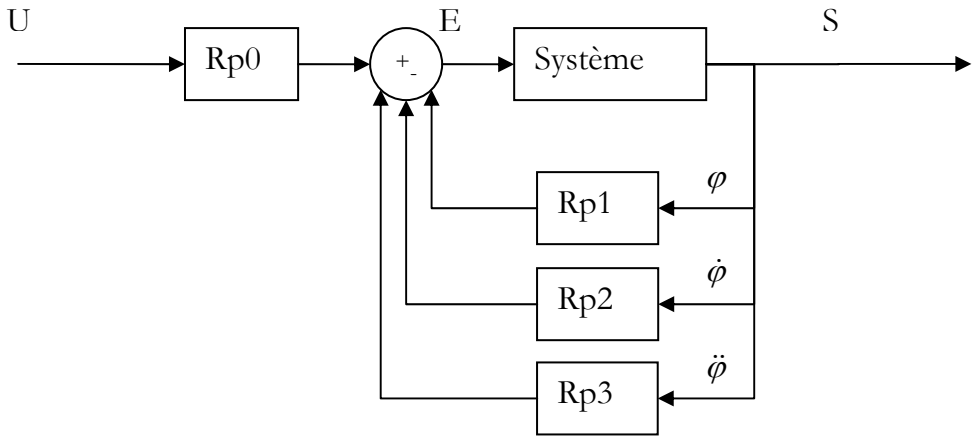


Figure III-16 Schéma du système implanté dans la MP2028

Il s'agit d'un asservissement à retour d'état. L'objectif est d'obtenir une dynamique en boucle fermée grâce aux informations angulaire de position, vélocité, et accélération. Les blocs Rp1, Rp2 et Rp3 sont des gains déterminés à partir d'une dynamique idéale souhaitée. Ces 3 blocs permettent de fixer complètement la dynamique du système. Le principal inconvénient du retour d'état est la modification du gain en boucle fermée qui n'est pas réglable via Rp1..3. La manière la plus simple de modifier le gain statique est de placer Rp0 avant le comparateur ; ainsi il est possible d'obtenir en boucle fermée un gain unitaire. En effet pour un système :

$$\frac{N(p)}{D(p)} \quad \text{avec un retour d'état ,} \quad u(t) = e(t) - K.x(t)$$

On aura en boucle fermée : $\frac{N(p)}{K(p)+D(p)}$, et par conséquent le gain statique sera :

$$\frac{N(0)}{K(0)+D(0)}$$

En posant $Rp0 = \frac{K(0)+D(0)}{N(0)}$, on obtiendra en boucle fermée un gain unitaire.

Cas de l'asservissement par retour d'état :

Afin de réaliser un asservissement performant, nous avons utilisé la carte Mp2028 de la société Micropilot. Cette carte a l'avantage de regrouper l'essentiel des capteurs nécessaires à la navigation pour un poids de 28g seulement. Micropilot fournit en outre un logiciel : horizon.Il permet d'exploiter le fonctionnement de la carte en modifiant les paramètres de l'asservissement. Ces paramètres sont les gains de boucle de type P.I.D. Cependant le modèle d'asservissement est entièrement figé, ce qui rendait inutilisable en l'état la carte pour Casper. En effet le modèle de l'asservissement a été conçu pour des appareils à voilure fixe fonctionnant sur le schéma classique des avions, ce qui n'est absolument pas le cas de Casper. Cependant avec l'acquisition du Kit de développement « XTender » nous avons désormais la possibilité de réaliser notre propre code gérant l'asservissement. Il est à noter que nous ne pouvons pas modifier le code source de la carte qui reste la propriété de Micropilot. Nous pouvons cependant gérer une couche « supérieur ». Ainsi le code source assure le calcul de différents paramètres comme l'angle magnétique et l'accélération angulaire (sauf en lacet). Nous pouvons alors utiliser et traiter ces paramètres pour mettre en œuvre notre propre correcteur.

Micropilot fournit un exemple de code d'asservissement, mais une fois de plus ce code était inadapté à notre usage.

Dans un premier temps nous nous sommes attachés au contrôle en lacet qui semblait le plus simple à réaliser. Afin de réaliser pratiquement l'asservissement que nous avions calculé, il nous fallut transposer notre calcul en langage C et en un processus discret pour qu'il soit pleinement exploitable par la carte.

C'est pourquoi la dérivation d'une grandeur est obtenue (au premier ordre) par la différence entre cette grandeur à un instant *n* et cette même grandeur à l'instant *n-1* divisée par le temps d'échantillonnage :

$$\dot{x}(n.T) = \frac{x(n.T) - x(n.T - T)}{T}$$

Ainsi nous aurons les variables u0, u1,… chaque incrémentation correspondant à un retard d'un échantillon (n.T*, (n-1).T, (n-2)T,…*)

Lors de l'implémentation du contrôleur, nous avons en grande partie suivi la structure du modèle proposé par Andreas Goërmer pour le fonctionnement de la carte avec les différentes interfaces, que sont les logiciels Horizon et la radiocommande FUBATA, notamment en ce qui concerne l'utilisation des canaux d'informations.

La programmation d'un contrôleur est soumise à de nombreuses contraintes. Outre le manque d'informations fournies par Micropilot dans les diverses documentations, il est nécessaire d'être très vigilant sur les grandeurs manipulées. En effet plus les calculs sont nombreux et plus le traitement prend du temps. Or la carte fonctionne de manière numérique, il faut donc qu'à chaque nouvelle période

d'échantillonnage, les calculs de la période précédente soient effectués. Cela conduit à restreindre l'utilisation de nombreuses variables de taille trop importante, telles que les *float* ou les *double*. Ainsi nous n'avons utilisé que le type primitif long lors des calculs. C'est pourquoi, bien souvent dans le code les grandeurs sont multipliées par 10 ou 100 de manière à ne pas traiter les virgules flottantes. Il s'agit à la fin simplement de diviser par le bon facteur pour obtenir les valeurs réelles.

Le taux d'échantillonnage peut être choisi à 5Hz ou à 30Hz. Afin d'assurer un asservissement le meilleur possible, il serait judicieux de choisir 30Hz. Cependant il est parfois nécessaire de prendre 5Hz lorsque des problèmes inattendus surviennent. En effet ils peuvent être dus à l'interruption brutale de calculs lourds n'ayant pu se faire dans le temps imparti. De plus l'angle de cap fourni par la carte n'est réactualisé qu'à la fréquence de 5Hz. C'est pourquoi nous avons choisi de concevoir un asservissement fonctionnant à cette fréquence de 5Hz.

Influence de la poussée.

La poussée, nous l'avons vu, a une grande influence sur l'efficacité des volets. Il est donc absolument nécessaire d'en tenir compte lors de la conception de l'asservissement. Il est complexe d'extraire une loi reliant la poussé et l'efficacité des volets du fait des non linéarités. Le parti pris de ce PIP est d'utiliser une table de comparaison. On définit des intervalles de poussé que l'on associe à un gain. Ce gain sera utilisé pour compenser l'efficacité des volets. Attention, la poussé n'est pas directement mesurée. Nous nous fondons essentiellement sur la commande passée, qui n'est pas forcément identique au résultat. Malgré ceci nous considérerons que cette approximation est valable.

IV. Mise en œuvre expérimentale

A. Correcteur RST

Sous Simulink le modèle d'asservissement fonctionne très bien. Cependant, le passage à la pratique se révèle plus délicat. En effet Simulink ne permet pas vraiment une simulation en temps réel : pas de calcul « ordonné » des différents paramètres, ce qui ne permet pas vraiment de conclure quant à la pertinence de notre asservissement. .Ainsi pour simuler au mieux le fonctionnement de la carte nous avons mis en place un fichier Matlab « simulant le fonctionnement de notre asservissement » dans la carte de manière discrète. Nous avons alors pu constater de grandes divergences entre les résultats attendus et ceux obtenus. En particulier la présence de saturations sur la commande et la sortie

conduit à des oscillations incontrôlables. En effet les volets se braquent en butée dans un sens puis dans l'autre.

Ainsi, il nous a semblé nécessaire de tester d'avantage les asservissements en simulation avant de les implémenter dans la MP2028. Pour cela, nous avons simulé le fonctionnement d'un algorithme tel qu'il sera implémenté dans la carte (cf code Matlab xx).

L'inconvénient de cette nouvelle simulation est qu'elle se fonde sur le modèle théorique du drone qui n'est pas exact.
En testant notre asservissement nous obtenons :

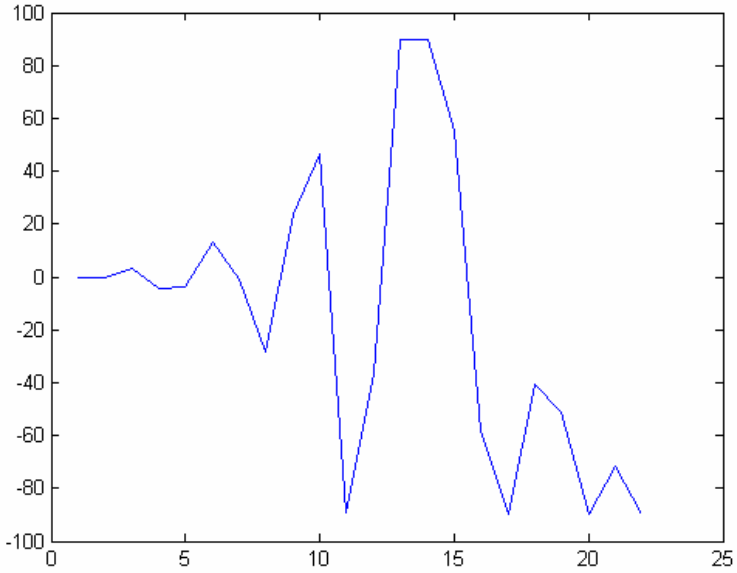


Figure IV-1 Réponse simulée de l'algorithme RST

Ainsi le résultat de la simulation n'est pas satisfaisant. En analysant les causes de l'échec de notre correcteur RST, nous avons constaté que la commande qu'il produisait au système ne pouvait pas être correctement suivie.

Figure IV-2 Essai à distance

B. Correcteur à retour d'état

Nous avons eu la possibilité de tester notre asservissement avec Andréas Goërmer sur le drone. L'asservissement s'est montré assez rapide (temps de réponse d'environ 3 secondes) et peu oscillant. Les faibles oscillations observées sont dues à l'inadaptation entre le code qui était implémenté et notre modèle d'asservissement. En effet le code qui était implémenté par Andréas supposait Rp0=Rp1. C'est pourquoi nous avons créé notre propre code dont l'implémentation dans la carte autopilote Mp2028 s'est révélée ardue. A l'heure actuelle des efforts se poursuivent pour rendre cet asservissement opérationnel.

De plus, la correction que nous avons mise en place doit permettre une commande de l'angle de cap du drone. Cependant, d'un point de vue pratique, il est souhaitable que l'opérateur n'ait pas à commander un angle, mais une vitesse angulaire. Ainsi, au repos, le drone conservera son cap.

V. Conclusion

A. Bilan

Ce PIP nous a permis de nous confronter aux différences entre théorie et pratique. En effet au cours de ce projet nous nous sommes impliqués dans toutes les étapes de la conception d'un asservissement :
Utilisation de théories connues, recherche de compléments théoriques, conception de correcteurs de différents types, simulation de leur fonctionnement, recherche d'optimum, et enfin tests sur le système réel. Nous avons ainsi pu constater à quel point il était nécessaire de sans cesse adapter nos modèles à la réalité, ce qui n'est pas toujours possible. En particulier, il est nécessaire de garder une certaine distance par rapport aux résultats obtenus par simulation avec l'outil Simulink.

Au delà de l'aspect purement technique et scientifique de ce projet, ce PIP fut pour nous l'occasion de constater à quel point une démarche structurée était nécessaire pour obtenir rapidement les résultats souhaités. En particulier le respect d'un planning prévisionnel peut être rendu très difficile à mesure que de nouveaux obstacles se présentent.

B. Remerciements

Nous souhaitons tout particulièrement remercier Andréas Goërmer, qui nous a communiqué plusieurs résultats préliminaires nécessaire à notre étude. Il nous a également fait profiter de son expérience dans la manipulation et le fonctionnement de Casper et de son avionique et ce, même après la fin de son stage au D.A.S.

Nous tenons également à remercier Mr Vincent, qui nous a aidé à réaliser une manipulation et à exploiter les résultats, ainsi que Mr Yves Brière.

Enfin, merci à tous les membres de l'équipe micro drone et au département du DAS pour les moyens techniques et humains, qu'ils ont mis à notre disposition.

VI. Annexes

A. Procédures d'utilisation de la MP2028

Mise en place du matériel :

L'ensemble de l'avionique est réparti sur le carénage du micro drone. Dans la phase d'essai, une alimentation stabilisée fournit la puissance nécessaire au

fonctionnement de la carte et des servomoteurs. Ces derniers fonctionnent sur des sources séparées de manière à ce que de fortes variations de tension sur les actionneurs ne perturbent pas le fonctionnement de la carte.

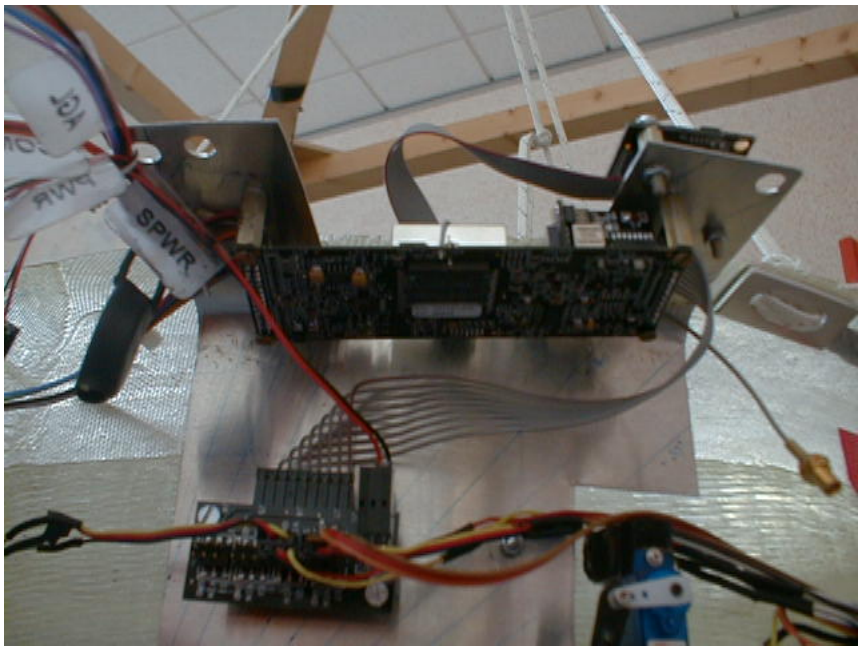


Figure VI-1 Implantation de la carte sur Casper

Mise en œuvre logicielle :

Ce guide à été élaboré à partir de celui proposé par Andréas Goërmer, de la documentation de la carte MP2028g fournie par Micropilot et de notre propre expérience. Il se veut le plus complet possible et doit permettre de faire face aux principales situations susceptibles d'être rencontrées lors de l'implémentation d'un nouveau code de contrôle. Cependant nous supposerons que le logiciel Horizon et le kit de développement ont correctement été installés au préalable (se référer au manuel d'utilisation de la carte).

La carte MP2028 doit être branchée via le port série du PC.

Nous allons décrire la démarche pour intégrer notre propre code. On suppose que les logiciels sont installés dans le répertoire : « rep\ » (par défaut « rep\ » = « C:\Program Files\MicroPilot »). Il sera fréquemment nécessaire d'ouvrir des fichiers codés en texte. Nous conseillons l'utilisation de Crimson editor ou de tout autre éditeur capable de reconnaître le langage utilisé (éditeur de code). Cela permet d'avoir un meilleur affichage et d'éviter des erreurs.

1ère utilisation :

Copier le contenu du dossier « **rep \XTENDER3\examples\userCode** »
vers le nouveau dossier « rep\ XTENDER3\examples\Casper »

Ouvrir « **rep\XTENDER3\examples\userCode\userCode.c »** et le sauvegarder sous le nom qui sera utilisé plus tard. Nous utiliserons « **Casper_userCode.c** ».

Modifier le fichier de la manière suivante:
ligne 72, remplacer « **int** » par « **long** »
ligne 34, insérer: « **#include "usercode.h »**
le sauver sous « **Casper_userCode.c** »

Implémenter votre propre code, le sauver.

A l'aide d'un éditeur de texte modifier le fichier :
« rep\XTENDER3\examples\Casper\gdb.bat » Changer le répertoire
« %MicroPilotSDK3%/examples/userCode/gnuc/ » par:
« %MicroPilotSDK3%/examples/Casper/gnuc/»

Saver «**gdb.bat**»

Ouvrir le fichier « **rep\XTENDER3\examples\Casper\ makefile** » à l'aide d'un éditeur de texte. A la ligne 42: remplacer « exampleUserCode.c » par « **Casper_userCode.c** ». Cela doit permettre de s'assurer de la bonne compilation du code.

A effectuer à chaque nouvelle modification du code :

Compilation de "**Casper_userCode.c**":
Ouvrir une invite :"cd rep\MicroPilot\XTENDER3\examples\Casper". Executer "**make-usercode.bat**" pour compiler le code. Vérifier qu'il n'y ait pas d'erreur.

Exécuter "**appendUser.bat**". Cette commande permet de créer "**mflyanduser.bin**" dans le fichier "**rep\MicroPilot\XTENDER3\examples\Casper\gnuc".** Vérifier la création de ce fichier (regarder la date) afin d'être sûr que cela soit la dernière version du fichier.

Copier "rep\MicroPilot\XTENDER3\examples\Casper\gnuc\mflyanduser.bin" dans le dossier "rep\MicroPilot\XTENDER3\bin\".

Relier la MP2028g à l'aide du câble au port série de l'ordinateur.

Se placer dans le répertoire "**rep\XTENDER3\bin"** et exécuter le batch "**flashld-mflyanduser.bat**"

Mettre la carte sous tension comme indiqué par l'invite de commande : une tension de 9V est recommandée au minimum. Il faut patienter un peu, environ 3 minutes.

Lorsque le code est actualisé dans la carte (invite de commande en pause), mettre la carte hors tension et fermer l'invite de commande.

Ouvrir une session Hyperterminal

Mettre la carte MP2028 sous tension, et patienter le temps de l'initialisation.

Lors de l'initialisation, il est possible de voir si le GPS est activé ou non. Lors d'essais en intérieur, il est recommandé de le désactiver. En effet, la détection risque de ne pas avoir lieu et donc l'initialisation peut échouer. Comme la procédure d'initialisation du GPS est bloquante, la carte restera en l'état et il ne sera pas possible de poursuivre. Pour désactiver l'étalonnage du GPS presser "**ffff**".

Accéder au menu en pressant "**qqqq**".

Changer la variable n° **588** en tapant "**p**", entrer le numéro de variable :"**588**", et changer sa valeur de 0 à 2.(**Enable user defined Code)** mettre à jour le code en appuyant sur "**w**".

Lorsque le code est actualisé, mettre la carte hors tension pendant quelques secondes, puis rallumer la carte et laisser l'initialisation se poursuivre.

Eteindre l'hyperterminal et lancer Horizon. Pour pouvoir communiquer avec la carte appuyer sur l'icône « Connect ». Horizon doit alors communiquer avec la carte, l'icône de statut « ready » apparaît rouge. L'initialisation du GPS peut être longue, afin de passer outre il faut double cliquer sur l'icône de statut « GPS » (pouvant apparaître « OSAT »), elle doit passer au jaune. Lorsque « READY » est vert alors tout est prêt pour utiliser le code utilisateur.

B. Fonctions Matlab utilisées

Dans cette section sont regroupés l'essentiel des codes Matlab qui nous furent utiles pour réaliser notre asservissement. Ces derniers sont précédés du nom du fichier auxquels ils font référence.

Fichier Matlab : RegulationPlacement.m

Ce fichier calcul un correcteur par la méthode des placements des pôles. A partir de la dynamique en boucle ouverte du système, on récupère les polynômes S,R et T.

```
%Spécification du système en boucle ouverte: q^-d.B/A : a1+a2.z^-1+...+an.z^-n

%Note: Pour matlab l'ordre le plus important est à gauche :a0^n a1^2...an^0
%mais nous travaillons avec : a0+a1^-1+a2^-2...
%pour la convolution c'est la même chose, mais pas pour la deconvolution !!!

%Système
A0=[1 -.5175]
%Ajout des deux intégrateurs
A=conv(A0,[1 -2 1]);
 B=[0.00514];
d=0;

%Spécification des performances en régulation:
Xsi=.9;
w0=1;
fnat=w0/(2*pi);
Te=0.2;  %temps d'echantillionnage

%Ajout de pôles particuliers
Pf=[1]

%choix de parties fixes
Hr=[1 1];
Hs=[1 -1];

%Spécification des performances en poursuite
pXsi=.9;
pw0=2;
pfnat=pw0/(2*pi);

[Bm,Am]=fd2pol(pfnat,pXsi,Te);

%calcul de la fonction de transfert numérique
[regnum,regden]=fd2pol(fnat,Xsi,Te);

Pd=regden;
P=conv(Pd,Pf);

%résolution de P=A.S'.Hr+q-d.B.R'.Hr
[Rp,Sp,nrp,nsp]=bezoutd(A,B,Hs,Hr,P);


%détermination de T:
if (sum(B)==0)          %sum(B)=B(z^-1=1)
   G=1;
else
   G=1/sum(B);
end

S=conv(Hs,Sp);
R=conv(Hr,Rp);
T=G*P;

%[R2,R1]=deconv(R,[1 -1]);
[R2,R1]=deconv(fliplr(R),[-1 1]);
R1=fliplr(R1);
%R1=R1(1);   %réduction de R1
R2=fliplr(R2);
%suppression des zéros inutiles dans R2
while (R1(end)==0 && length(R1)>1)
   R1=R1(1:end-1);
end
```

Fichier Matlab : CorrecteurRetourEtat.m

```
%Calcul d'un régulateur à retour d'état.. Ce fichier fournit le vecteur K sous la forme K1 ;
K2 et K3 ainsi que le coefficient K0 permettant d'annuler l'erreur statique à un échelon.
%Le schéma du régulateur doit être le suivant:
%
%        ----------  y(t)  ---
% u(t)-->| N(p)/D(p |----->|1/p|---->  Y(t)
%        ----------        ---
%
%


%modèle discret du système
%D=[1 -.5175];
%N=[0.00514];


%modèle LTI du système
A=[0 1 0;0 0 1;0 -1.095 -2.604];
B=[0;0;1];
C=[5.472 1.047 0.064];
D=[0];
sys_dc=ss(A,B,C,D);

%Modèle continu
sys_tf = tf(sys_dc);

%Récupération du numérateur du
système, classement par décroissance de
l'ordre
den_bo=sys_tf.den{1};
num_bo=sys_tf.num{1};
%Performances en boucle fermée désirée
:num_bf=[]   par décroissance de
%l'ordre
%ici ksi=0.9 w=1 ,
%pole secondaire arbitrairement à -6 afin
de ne pas influer sur la dynamique
den_bf=conv([1 1.8 1],[1 6]);

%Remise dans l ordre croissant des
coefficients
k2=fliplr(den_bf-den_bo);
k2=k2(1:end-1);
k=k2;
K2=k2;

%Mise en place du gain unitaire : K0=
N(0)/(K(0)+D(0)) donc on * par
%l'inverse, attention au classement des
ordres différents pour sys et pour
%K.

K1=(K2(1)+den_bo(end))/num_bo(end
);
```

Fichier Matlab : bezoud.m

```
function [Rp,Sp,nrp,nsp]=bezoutd(A,B,Hs,Hr,P)
%function [Rp,Sp,nrp,nsp]=bezoutd(A,B,Hs,Hr,P)
%solves AHsSp+BHrRp=P by coefficient comparison. Delay and discretization delay
need to be integrated in B
%(zeros at the begining).
%
```

```
%inputs:
%A=[a0 a1 ... aNa] ... vector of model denominator coefficients A=a0 + a1z^(-1) +
a2z^(-2) +...+ aNaz^(-Na)
%B=[b0 b1 ... bNb] ... vector of model numerator coefficients B=b0 + b1z^(-1) + b2z^(-
2) +...+ bNbz^(-Nb)
%Hs=[hs0 hs1 ... hsNhs] ... vector of controller denominator fixed part Hs=hs0 +
hs1z^(-1) +...+ hsNhsz^(-Nhs)
%Hr=[hr0 hr1 ... hrNhr] ... vector of controller denominator fixed part Hr=hr0 + hr1z^(-
1) +...+ hrNhrz^(-Nhr)
%P=[p0 p1 ... pNp] ... vector of desired polynomial coefficients P=p0 + p1z^(-1) +
p2z^(-2) +...+ pNpz^(-Np)
%outputs:
%Rp=[rp0 rp1 rp2 ...] ... vector of coefficients for resulted controller numerator
%Sp=[sp0 sp1 sp2 ...] ... vector of coefficients for resulted controller denominator
%nrp ... order of Rp
%nsp ... order of Sp

%
%written by: J. Langer, I.D. Landau, H. Prochazka
%7th june 2002


PRECISION=1e-16;

D=size(A);
if D(1)>1, A=A'; end;
D=size(B);
if D(1)>1, B=B'; end;
D=size(Hs);
if D(1)>1, Hs=Hs'; end;
if D(1)==0, Hs=1; end;
D=size(Hr);
if D(1)>1, Hr=Hr'; end;
if D(1)==0, Hr=1;end;
D=size(P);
if D(1)>1, P=P'; end;

na=length(A)-1;
nb=length(B)-1;
np=length(P)-1;
nhs=length(Hs)-1;
nhr=length(Hr)-1;

if (nhs>0), Ah=real(conv(A,Hs)); else
Ah=A*Hs; end; % Ah = A * Hs
nah=length(Ah) -1;

if (nhr>0), Bh=real(conv(B,Hr)); else
Bh=B*Hr; end;
nbh=length(Bh) -1;
if (np>nah+nbh-1), disp('Bezout error:
too many poles');end;


% increase size of P using zeros if
necessary
if (np<nah+nbh-1),
%       P=[P zeros(1,nah+nbh-1-np)];
% set remaining poles onto a circle with
radius rmin
        rootsPdes=roots(P);
        nextra=nah+nbh-1-np;
        rmin=1e-16;
        angle=[0:nextra-1]'/nextra*2*pi;
        j=sqrt(-1);
        rootsPextra=rmin*exp(j*angle);
        P=poly([rootsPdes;rootsPextra]);
        np=nah+nbh-1;
end;
P,

nsp=nbh-1;
```

```
nrp=nah-1;

%matrix is smaller than vector PD
if (np>nah+nbh-1),
   disp('Order of model denominator is
too low! Add a polynom of higher order
to Hs or Hr. ');
end;

% ns=nsp+nhs
% nr=nrp+nhr

M=[];
for j=1:nsp+1,
        V=[];
        if (j>1), V=[V ; zeros(j-1,1)];
end;% zeros in front of Ah
        V=[V ; Ah'];% Ah
        if (j<=nsp), V=[V ; zeros(nsp+1-
j,1)]; end;% zeros behind Ah
        if (length(V)~=nah+nbh),
disp('bezoutb: error V'); end;
        M=[M V]; % add one column to
M
end;

for j=1:nrp+1,
        V=[];
        if (j>1), V=[V ; zeros(j-1,1)]; end;
        V=[V ; Bh'];
        if (j<=nrp), V=[V ; zeros(nrp+1-
j,1)]; end;
   if (length(V)~=nah+nbh),
disp('bezoutb: error V'); end;
        M=[M V];
end;

D=size(M);
if (D(1)~=nah+nbh), disp('bezoutb:
error size M row'); end;
if (D(2)~=nah+nbh), disp('bezoutb:
error size M column'); end;

% make P column vector
P=P';

global M1;
M1=M;

X= M\P;
% coefficients are real values
X=real(X);

% make X row vector
X=X';
Sp=X(1:nsp+1);
Rp=X(nsp+2:nsp+nrp+2);
```

Fichier Matlab : cont2disc.m

```
function [N_d,D_d]=cont2disc(N_c,D_c,Ts)
%function [N_d,D_d]=cont2disc(N_c,D_c,Ts)
%
%the function realize the conversion of a continues transfer function (in s):
%          bms^m + ... + b2s^2 + b1s + b0
%   G(s)= --------------------------------   ; m<=n
%          ans^n + ... + a2s^2 + a1s + a0
%to its
%discret form (in z):
%          g0 + g1z^-1 + g2z^-2 + ... + gks^-k
%   G(z)= -----------------------------------
%          h0 + h1z^-1 + h2z^-2 + ... + hls^-l
%using zero order hold.
%inputs:
%N_c = [bm ... b2 b1 b0] ... vector of continues numerator coefficients
%D_c = [an ... a2 a1 a0] ... vector of continues denominator coefficients
%Ts ... desired sampling time for discret transfer function
%outputs:
%N_d = [g0 g1 g2 ... gk] ... vector of discret numerator coefficients
%D_d = [h0 h1 h2 ... hl] ... vector of discret denominator coefficients
%
%written by: H. Prochazka, I.D. Landau
%7th june 2002

sys_c=tf(N_c,D_c);                  %continue system
sys_d=c2d(sys_c,Ts,'zoh');          %conversion to a discret form (in z) with 0 order hold
set(sys_d,'variable','z^-1');       %conversion to a discret form in z^-1

N_d=get(sys_d,'num');               %extraction and
N_d=cat(1,N_d{:});                  %conversion to obtain a vector of numerator
coefficients
D_d=get(sys_d,'den');               %extraction and
D_d=cat(1,D_d{:});                  %conversion to obtain a vector of denominator
                                    coefficients
```

Fonction Matlab : fd2pol

```
function [num,den]=fd2pol(fnat,dpm,Ts);
%function [num,den]=fd2pol(fnat,dpm,Ts);
%computes numerator num and denominator den of 2nd order discret transfer function
%from natural frequency fnat and damping dmp of its continues equivalent.
%The continues 2nd order transfer function is of the form:
%                    2.pi.fnat
%   G2(s)= ---------------------------------------
%         s^2 + 2.dmp.2.pi.fnat.s + (2.pi.fnat)^2
%
%The discret transfer function is:
%           b1z^-1 + b2z^-2
%   Gd(z)= -------------------
%         1 + a1z^-1 + a2z^-1
%
%inputs:
%fnat ... natural frequency of continues 2nd order system
%dmp ... damping of continues 2nd order system
%Ts ... desired sampling time
%outputs:
%num=[0 b1 b2] ... vector of discret numerator coefficients
%den=[1 a1 a2] ... vector of discret denominator coefficients
%
%written by:  H. Prochazka, I.D. Landau
%7th june 2002


w=fnat*2*pi;
omega=w*sqrt(1-dpm^2);
alpha=exp(-dpm*w*Ts);
beta=cos(omega*Ts);
delta=sin(omega*Ts);
A1=-2*alpha*beta;
A2=alpha*alpha;
B1=1-alpha*(beta+(dpm*w*delta/omega));
B2=alpha^2+alpha*((dpm*w*delta/omega)-beta);
den=[1 A1 A2];
num=[0 B1 B2];
```

C. Code implémenté dans la carte MP2028

Il n'est décrit ici que la partie propre à l'assevissement en lacet du drone

```
//*****************************************************************************
//***  userPID8_1   Start of userPID8
//***  Essai d'un correcteur à retour d'états pour le lacet uniquement au 24/05/2005

   case USER_PID8 :                    //PID loop works with 5/30 Hz


//*****************************************************************************
//***  userPID8_2   Adjustment of coefficients for polynomials R and S
                Rp0=*Rp0_raw*10000/9113;        //adjust final gain to unit
                Rp1=*Rp1_raw*10000/9113;        //angular feed-back
                Rp2=*Rp2_raw*10000/9113;        //angular velocity feed-back
                Rp3=*Rp3_raw*10000/9113;        //angular acceleration feed-back


//*****************************************************************************
//***  userPID8_3   PD Controller for Yaw axis

                        Hdg_ref=*ref_ident_raw;


//*****************************************************************************
//***  userPID8_3_3         Reparation of owerflow problem of commanded signal, signal must be between 0 and 360 degrees

        if(Hdg_ref < 0)
                {
                        Hdg_ref=Hdg_ref+36000;
                }
                else
                {       if(Hdg_ref > 36000)
                        {
                                Hdg_ref=Hdg_ref-36000;
                        }
                }
      *disp1=Hdg_ref;                   //monitoring Hdg_ref (variabe 8004 is used, sent by telemetry too)
```

```
//*****************************************************************************
//*** userPID8_3_4        Calculation of angular acceleration (yaw) (and shifting angular velocity)

        yaw_p1=yaw_p0;
        yaw_p0=*iYaw_p;

        yaw_pp0=(yaw_p0-yaw_p1)/3; //divide by 3 instead of 30 to ensure the use of "long"

   *disp2=yaw_pp0;          //monitoring yaw_pp0 (variabe 8004 is used)

//*****************************************************************************
//*** userPID8_3_5        Calculation of regulation error (Yaw axis)

        Hdg_err=Hdg_ref - *Hdg;

        *disp3=Hdg_err;             //monitoring yaw_pp0 (variabe 8004 is used)


//*****************************************************************************
//*** userPID8_3_6        Adjustment of regulation error (Yaw axis)

                if(Hdg_err > 18000)
        {
                        Hdg_err=Hdg_err-36000;
                }
                else
                {       if(Hdg_err < -18000)
                        {
                                Hdg_err=Hdg_err+36000;
                        }
                        else
                        {
                                //nothing to do
                        }
                }

//*****************************************************************************

        *disp4=Hdg_err;             //monitoring Hdg_ref_m (variabe 8004 is used)


//*****************************************************************************
//*** userPID8_3_7        Calculation of signal for servo actuator (Yaw axis)
```

```
		Yaw_mix = (Rp0 * Hdg_ref) +(Rp1 * *Hdg) - (Rp2 * *iYaw_p) -(Rp3 *
yaw_pp0 / 10);


	*disp6= (Hdg_err * (- Rp0));
	*disp7= -(Rp1 * *iYaw_p);
	*disp8= -(Rp2 * yaw_pp0 / 10);



//*****************************************************************************
//***  userPID8_3_8         Adjustment of decimals of Yaw_mix

		Yaw_mix = Yaw_mix/100;

		*disp9=Yaw_mix;



//*****************************************************************************
//***  userPID8_4   Adjustment of servo command to actual thrust level by applying
ILUT

		thrust=*ref_Thrust_raw*1000;

		if(*ref_Thrust_raw <= 2600)
		{
			t=(thrust-2500000)*(1052-1391);
	   t=t/(2600000-2500000);
	   t=1391+t;
	}
	else
	{ if( (*ref_Thrust_raw > 2600) && (*ref_Thrust_raw < 2700) )
	  {
				t=(thrust-2600000)*(799-1052);
		t=t/(2700000-2600000);
	     t=1052+t;
	  }
	  else
	  {	if(*ref_Thrust_raw > 2700 && *ref_Thrust_raw < 2750)
		{
				t=(thrust-2700000)*(750-799);
			t=t/(2750000-2700000);
		   t=799+t;
		}
		else
		{	if(*ref_Thrust_raw > 2750 && *ref_Thrust_raw < 2800)
```

```
                {
                                t=(thrust-2750000)*(750-710);
                        t=t/(2800000-2750000);
                    t=710+t;
                }
                else
                {       if(*ref_Thrust_raw > 2800 && *ref_Thrust_raw < 2850)
                        {
                                        t=(thrust-2800000)*(669-710);
                                t=t/(2850000-2800000);
                            t=710+t;
                        }
                        else
                        {       if(*ref_Thrust_raw > 2850)
                                {
                                                t=(thrust-2850000)*(625-669);
                                        t=t/(2900000-2850000);
                                    t=669+t;
                                }
                                        }
                                }
                        }
                }
                *disp10=t;     //monitoring t (variabe 8010 is used)
        if(*Radio_raw==1)
        {
                Yaw_mix=Yaw_mix*t;

//*** end of userPID8_8
//*************************************************************************


//*************************************************************************
//***  userPID8_5   Correction of decimals servo command

                Yaw_mix=Yaw_mix/1000;

//*************************************************************************
        }

//*************************************************************************
//***  userPID8_4   Servo mixing
                *fServo5 = - Yaw_mix;
                *fServo6 = + Yaw_mix;
                *fServo7 = - Yaw_mix;
                *fServo8 = + Yaw_mix;
```

```
//*****************************************************************************
//***  userPID8_5   End of userPID2
                return USER_RETURN_REPLACE;
```

D. Table des figures

E. Bibliographie

[1] *Commande des systèmes linéaires*, notes de cours ENSICA

[2] D.Arzelier, *Représentation et analyse des systèmes linéaires*, notes de cours ENSICA

[3] I.D.Landau, *Commande de Systèmes - Conception, Identification et Mise œuvre*

[4] Micropilot, *XTENDER Progammer's guide*, datasheet

[5] Micropilot, *MP2028g Installation and operations,* datasheet

[6] Andréas Goërmer, *Flight dynamics of a mini UAV*, rapport de stage ENSICA

[4] Site Internet : http://www-lag.ensieg.inpg.fr/landau/bookIC/