Universitat de Girona
Escola Politècnica Superior

Treball Final de Màster

Realitzat i defensat a SIGMA-CLERMONT (nom universitat) de França (país)

Màster: Master's in Industrial Engineering

Títol: Conception des systèmes automatiques à bas coût

Document: Memòria i annexos

Alumne: Marc López Montenegro

Director/Tutor: Norbert Blanco Villaverde
Departament: Eng. Mecànica i de la Construcció Industrial

Convocatòria (mes/any): Juny 2017

FICHE D'IDENTIFICATION DU DOCUMENT

Rapport de Projet		
TITRE DU DOCUMENT :	Conception des systèmes automatiques à bas cout Design of low cost automatic systems	
AUTEUR(S) :	Marc López Montenegro	
Date du document	**Nº de**	**Référence du document**
28 juin 2017	87	Projecte final Lopez Montenegro MARC

ABSTRACT:

This project presents the experimentation and the analysis of automatic systems in two different robots, the *gyropode* and the *quadcopter*, from the set-up to the simulation. The aim is to implement cheaper means of development like low cost embedded cards and software (Arduino cards, IDE Arduino and Matlab/Simulink) to control and simulate these structures.

Keywords: Control, simulation, *gyropode*, *quadcopter*, Simulink, Matlab, Microduino, low cost

RESUME :

Ce projet présente l'expérimentation et l'analyse de systèmes automatiques dans deux robots différents, le gyropode et le quadcopter, de la mise au point à la simulation. L'objectif est la mise en œuvre d'outils de développement avec cartes embarquées à bas coût et logiciels (cartes Arduino, IDE Arduino et Matlab/Simulink) pour contrôler et simuler ces structures.

Mots clés : Contrôle, simulation, gyropode, quadcopter, Simulink, Matlab, Microduino, bas coût

Table des matières

Table des figures

1. INTRODUCTION

1.1. ANTÉCÉDENTS

Aujourd'hui l'automatisation et le contrôle des systèmes sont utilisés partout, tant au niveau domestique que industrielle, pour faciliter la vie des personnes, optimiser des processus, assurer la répétabilité, etc. Les principaux avantages d'avoir des objets ou machines régies par les contrôleurs sont que le processus est optimisé et le résultat est probablement le meilleur (comme par exemple le contrôleur qui régit la température d'un four et qui permet cuire les aliments à la température désiré), cela permet de créer des systèmes autonomes qui facilite aux personnes afin de faire d'autres actions et que c'est possible économiser puisqu'on utilise des machines au lieu des employés. De plus, les machines sont plus efficientes que les personnes et travaillent toujours au même rythme. Par contre, il faut les faire un maintien périodique pour garantir leur correct fonctionnement et il y a quelques fois qu'elles sont en panne. Un des principaux inconvénients c'est que des emplois sont perdus à cause de cette automatisation bien que des autres apparaissent car il faut dessiner, programmer et surveiller les machines. Un autre inconvénient c'est que la production s'arrête quand il y a quelque erreur dans le processus et qu'il faut faire des contrôles de qualité pour garantir que le produit final est conforme aux spécifications.

Dans ce projet, les systèmes automatiques avec lesquels on travaillera ne sont pas de type industriel car les robots acquis sont assez petits. De plus, ces robots ne sont pas très puissants car ils utilisent des petites batteries et la conception de ceux-ci n'est pas vraiment sophistiquée. Ci-dessous il y a des images des robots

Figure 1: Robots acquis a) Gyropode et b) Quadcopter

On pourrait dire que ces robots ne sont pas les plus professionnels mais ils sont les appropriés pour faire des expériences et comprendre son fonctionnement. En outre, le gyropode et le quadcopter acquis sont idéaux car ils utilisent des cartes embarquées à bas cout, concrètement du type Microduino.

1.2. OBJECTIF

L'objectif de ce projet final est la mise en œuvre d'outils de développement avec cartes embarquées à bas coût (concrètement un type de carte Arduino laquelle s'appelle Microduino) pour piloter diverses structures associés à la conception et fabrication d'un robot de mise en situation.

1.3. SPÉCIFICATIONS

Pour atteindre à cet objectif deux types de robot seront utilisés : Le gyropode, un petit robot avec seulement deux roues lequel peut rester verticalement et le quadcopter, un drone avec quatre rotors.

Ces robots seront étudiés pour pouvoir avoir des modèles mathématiques qui décrivent leurs comportements réels. Ensuite, ces modèles pourront être introduits dans un environnement de simulation et de contrôle et on pourra voir les mouvements des robots dans l'ordinateur. Ça permettra faire des changements sur les robots et vérifier la réponse sans la nécessité de changer le programme qui les fait fonctionner.

Les spécifications de ce projet final peuvent être regroupées en différentes catégories. Pour les connaitre avec plus de détail ci-dessous se trouve le Tableau avec ces spécifications, séparées par les catégories, et avec une colonne laquelle indique si est une spécification requise (R) ou complémentaire (C).

Tableau 1: Description des spécifications de ce projet

Catégorie	Description	R/C
Objectif	Mise en œuvre d'outils de développement avec cartes embarquées à bas coût pour piloter diverses structures associés à la conception et la fabrication d'un robot de mise en situation	R
Ressources	Microduino Self-Balancing Robot kit	R
	Microduino Quadcopter kit	R
	Microduino Joypad pour contrôler les robots	R
	Oscilloscope Tektronix TDS 210 pour connaitre des paramètres des moteurs	R
	Tachymètre pour lire la vitesse angulaire des hélices du quadcopter	R

	Voltmètre pour vérifier et réparer quelques parties des robots	C
Résultats	Mise en œuvre des produits fournis	R
	Analyser et documenter les codes de commande	C
	Construire des modèles de simulation du quadcopter sous Matlab/Simulink	R
	Changer parties du gyropode pour une réduction homothétique et faire une maquette	C
	Réalisation des différentes pièces en usinage ou impression 3D pour la réduction homothétique	C
	Construire sous Scilab/Xcos et MATLAB/Simulink les modèles de simulation du gyropode	R
	Étendre le développement de la commande à un gyropode échelle 1.	C
Date limite	Présentation du projet dans la convocation Juin 2017	R
Coûts	Minimiser le coût du matériel	C
Autres	Faciliter l'ordre et la compréhension des résultats pour une éventuelle reprise du projet	R

Bien que dans les spécifications de ce projet il y a certains aspects liés à la conception mécanique des robots, seulement seront développés ceux liés au contrôle et la simulation des systèmes. En cas d'avoir suffisant temps, les autres seront aussi traités. Ç'est la raison pour laquelle quelques spécifications sont notées comme complémentaires dans le Tableau 1.

2. GYROPODE

2.1. ASSEMBLAGE ET TEST

Cette partie du projet comprend la description et les résultats de la première expérience avec un des robots utilisés dans cette thèse, le *"self-balanced"* robot (connu aussi en France sous le nom de gyropode), de la procédure d'assemblage jusqu'au programme qui permet de le faire marcher. En outre, tous les changements faits pour un fonctionnement correct seront décrits.

2.1.1. Principe de fonctionnement

Le gyropode est un véhicule avec seulement deux roues, qui peut se tenir en équilibre dans une position verticale avec un contrôle approprié. Ce système est souvent représenté comme dans la Figure 2. Le robot peut bouger dans l'axe X et tourner autour de l'axe Y (plan X-Z).

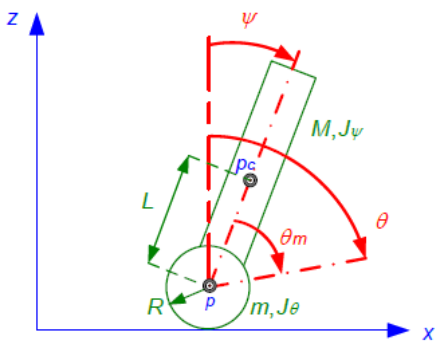


Figure 2: Dynamique du modèle du gyropode [1]

Quand l'angle ψ augmente, le gyropode doit se déplacer plus rapidement vers la même direction que la masse *M* pour le relever jusqu'au le *set point* ($\psi = 0$). La force qui fait bouger le robot est la force de frottement générée par le couple des deux moteurs pas à pas et le coefficient de frottement entre les pneus et la surface où ils sont placés.

Pour connaître la fonction de transfert du système, des équations dynamiques doivent être résolues. Ces équations sont la somme des forces dans chaque axe égal à la masse totale par accélération et la somme des moments égaux à l'inertie par l'accélération angulaire. Après avoir simplifié ces expressions, la transformation de Laplace peut être appliquée pour travailler dans le plan S et pour simplifier certaines autres opérations.

2.1.2. Le gyropode acquis

Le gyropode utilisé dans ce projet se compose d'un kit qui comprend toutes les pièces et tout le matériel nécessaire pour le faire fonctionner. Le kit a été acheté sur un site web [4] et a été construit en suivant un tutoriel sur une autre page web [5] qui comprend beaucoup d'informations sur les cartes Microduino et les robots. Cette page dispose aussi d'un code source ouvert qui permet au robot de s'auto-équilibrer et rester en position verticale.

Arduino est une plate-forme open-source, dans le domaine de l'électronique, conçue pour beaucoup de différentes applications. Elle a son propre langage de programmation, basé sur le code C ++ et sur libraires, et permet d'automatiser certains processus. Certains d'entre eux sont simples comme l'éclairage d'une LED (c'est l'équivalent du *"Hello world"* dans le monde de la programmation), mais il existe d'autres programmes, plus développés, qui prennent des informations sur les capteurs et donnent des instructions aux actionneurs, comme le gyropode le fait. Microduino est un type de carte Arduino qui peut fonctionner comme Arduino mais à une taille inférieure. Microduino dispose de nombreux modules qui peuvent être empilés pour effectuer différentes fonctions telles que la communication USB ou radiofréquence, mesurer l'accélération, etc.

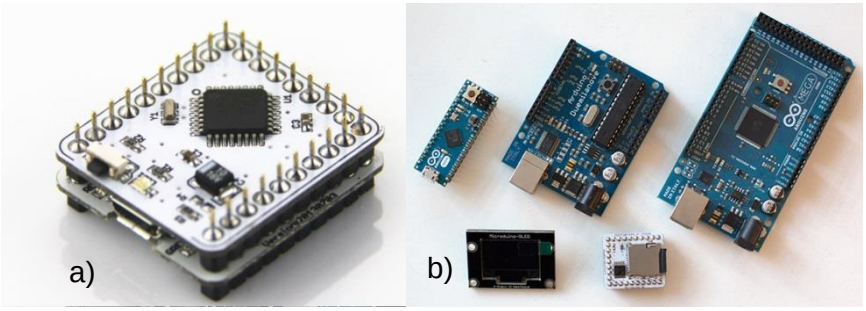


Figure 3: a) Modules Microduino assemblés ; b) Cartes Arduino et Microduino [6]

Ce produit est considéré comme un robot que tout le monde peut utiliser, car il s'agit uniquement d'assembler, de télécharger le programme et de le faire fonctionner. Comme il dispose d'un code 100% source ouvert, il est possible d'apporter des modifications au programme afin de faire des actions différentes et de l'expérimenter.

Bien que la page web et le tutoriel indiquent que, dans le kit il y a un module Core + et un module USBTTL. Pourtant, seul un module CoreUSB a été reçu car le Core + a besoin de l'USBTTL pour communiquer avec l'ordinateur, alors que et le CoreUSB peut

réaliser la même opération que ces deux modules mais en un seul. D'autres informations sur ces modules seront données au paragraphe 2.1.4 et à l'annexe A.

Tout le matériel inclus dans ce kit est indiqué dans le Tableau 2 et présenté à la Figure 4.

Tableau 2: Nomenclature du gyropode

Nº	Description	Quantité
1	Microduino-Module CoreUSB	1
2	Microduino-Module BT (Bluetooth)	1
3	Microduino-Module Motion	1
4	Microduino-Stepper	1
5	M2 Nylon vis, colonne et écrou	4, 4, 4
6	Colonne de cuivre	8
7	M4 Métal vis	2
8	2S Li-ion batterie (7.4V)	1
9	Lithium batterie balance charger	1
10	Micro-USB câble	1
11	Axis connecteur	2
12	Roues	2
13	Moteurs pas à pas	2
14	Stepper câble	2
15	Parties en bois	-
16	Tournevis	1

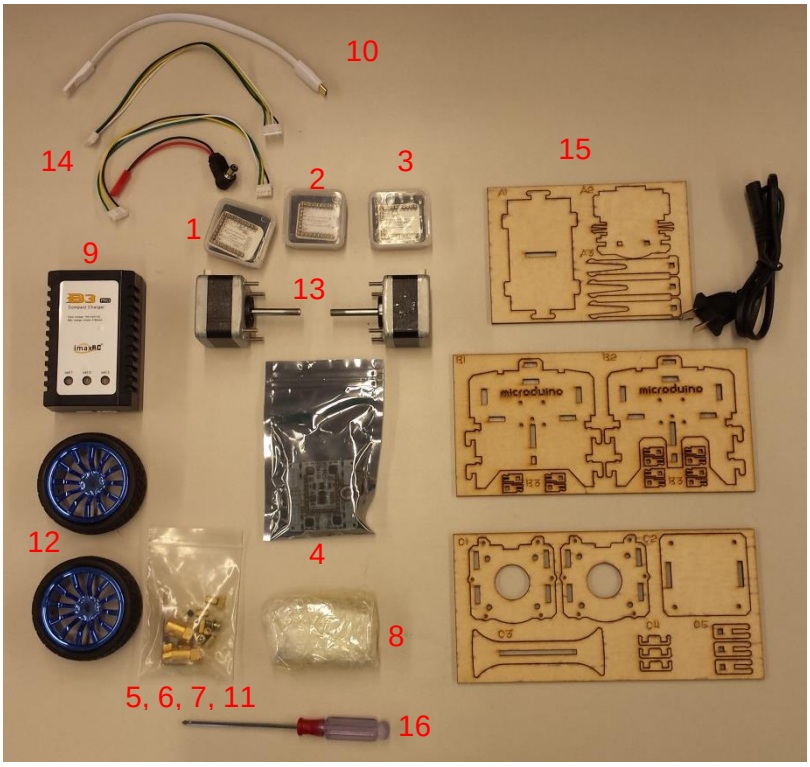


Figure 4: Image de tout le matériel inclús codifié comme au Tableau 2

2.1.3. Assemblage

La première étape était d'assembler, en suivant le tutoriel décrit antérieurement, toutes les parties du gyropode. Comme le tutoriel le montre, la difficulté est moyenne et il est possible de le terminer en deux heures. Avant de commencer à regrouper certaines pièces, le sticker qui protège le bois doit être enlevé afin de les assembler correctement. En somme, les étapes principales pour construire le gyropode étaient:

- Assemblage du châssis
- Fixer les deux moteurs pas à pas
- Fixer des roues
- Placer the batterie
- Fixer les autres parties en bois
- Empiler tous les Microduino modules
- Connecter tous les câbles
- Allumer le robot

Après avoir connecté la batterie au Microduino-stepper, il était possible de voir des étincelles dans le connecteur entre la batterie et le microprocesseur Microduino. En outre, certains plastiques avaient fondu et de la fumée en sortait. Cela s'est produit parce qu'il y avait un défaut de fabrication dans le connecteur. Les pôles plus et moins étaient en contact et ça a déclenché un court-circuit.

Pour résoudre ce problème, le connecteur a été ouvert et réparé au lieu d'en demander un nouveau au fournisseur, car cela prendrait quelques semaines. Comme le connecteur est devenu un peu délicat, un interrupteur a été soudé pour être utilisé au lieu du connecteur. Après quelques tests avec le voltmètre, la batterie a été allumée et tout fonctionnait correctement. Après tous ces changements, le gyropode a finalement été construit. La figure 4 montre le robot avec tous les modules Microduino empilés et le commutateur.

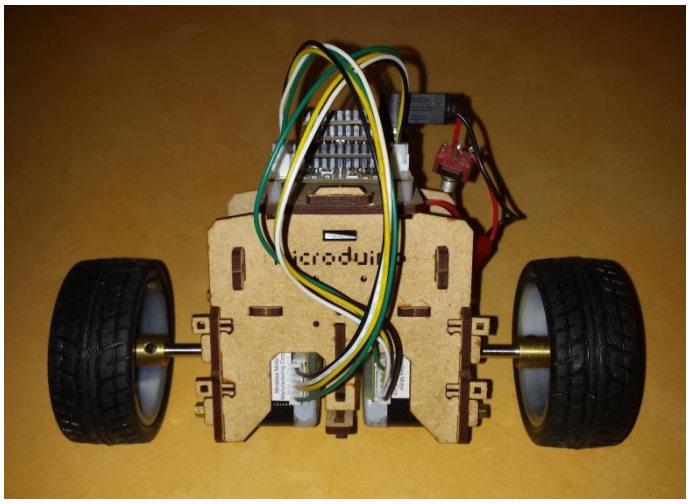


Figure 5: Gyropode fini

2.1.4. Le programme

En plus du processus d'assemblage, doit également être réalise le processus de programmation car le gyropode ne se maintient pas dans la position verticale à moins que le noyau du Microduino n'ait un programme approprié dans la mémoire. Tout d'abord, le logiciel Arduino for Microduino doit être téléchargé depuis la page web [7] et installé.

Comme dans un premier temps ce programme ne s'est pas ouvert (il est supposé qu'il ne fonctionne pas dans certains systèmes opératoires 64 bits), une machine virtuelle à 32 bits a été utilisée pour exécuter le logiciel avec succès. Deuxièmement, le Microduino-module CoreUSB doit être connecté à l'ordinateur et installé. Ensuite,

l'ordinateur reconnaîtra la carte Microduino et permet la connexion. Enfin, après avoir choisi la carte COM et la carte appropriée dans la barre de l'outil, le code peut être téléchargé sur le noyau.

Le robot est censé fonctionner sans apporter de modifications au programme, mais le fait d'utiliser le CoreUSB au lieu de Core + a fait que le programme était plus grand que la mémoire du noyau (le premier a une mémoire de 32 Ko et le second 64 Ko ou plus). Puisque demander la raison d'avoir un CoreUSB au lieu d'un Core + au fournisseur aurait pris quelques jours et l'achat d'un module Core + n'étant pas possible, la première option était d'essayer de simplifier le programme (ce n'était que 104% de la mémoire totale).

Le programme téléchargé comprend une fonction qui permet de communiquer avec le gyropode par radiofréquence. En effet, le but de cette première expérimentation était de savoir si le robot fonctionnait, et non pas de communiquer par radiofréquence (cette fonction ayant été supprimée du code, ainsi que toutes les variables liées à celle-ci). Lorsque cette modification a été appliquée, la taille totale du programme était alors un 72% de la mémoire totale et le programme pouvait être téléchargé.

Après avoir téléchargé le programme modifié, le logiciel Arduino for Microduino n'a envoyé aucun message d'erreur concernant la taille du programme. Cependant, le logiciel s'est bloqué pendant le téléchargement du code et il n'a pas pu être transféré sur le noyau puisque le robot n'a pas bougé quand il a été branché.

Pour savoir pourquoi la carte Microduino ou le logiciel ne fonctionnait pas, un autre type de Microduino Core, le module CoreRF, a été pris d'un autre robot. Puisque ce noyau n'a pas la prise pour le connecter à l'ordinateur, il était nécessaire d'utiliser le Module USBTTL, lequel a aussi été pris d'un autre robot, pour réaliser la connexion entre le noyau et l'ordinateur. Après avoir empilé et connecté ces modules à l'ordinateur, le code pouvait être transféré avec succès. Cela signifiait que le CoreUSB était défectueux et qu'il ne fonctionnait pas. Enfin, les modules ont été empilés sur le panneau Microduino-stepper du gyropode, il a été branché et il a commencé à s'équilibrer.

2.2. MODÉLISATION

Actuellement la simulation est un outil très important dans le monde de l'ingénierie. Utiliser la simulation a beaucoup d'avantages. Elle permet par exemple d'obtenir une connaissance des systèmes sans la nécessité de l'avoir ou de le construire. De plus, il

est simple et économique de faire des changements dans la conception, car cela évite d'acheter ou de fabriquer de nouveaux éléments jusqu'au dessin définitif. En revanche, il existe aussi des inconvénients puisqu'il est nécessaire d'acquérir des ordinateurs, parfois très puissants, et les licences des logiciels, qui sont souvent chères. Ces logiciels sont complexes à utiliser et la personne qui va réaliser des simulations doit connaitre leur fonctionnement et se familiariser avec eux pouvoir travailler plus efficacement. Un autre inconvénient est la nécessité de calculer un modèle mathématique pour pouvoir simuler correctement le comportement du système, or cela exige de nombreux calculs.

Dans cette partie du projet, nous trouverons les expressions mathématiques qui définissent le comportement du gyropode. L'équation de Lagrange sera utilisée pour définir le système et obtenir les expressions en fonction des coordonnées généralisées, malgré l'existence de nombreuses autres méthodes comme les travaux et puissances virtuels, les équations de la dynamique, etc. Cette méthode permet de calculer la trajectoire d'un objet si on connait l'énergie cinétique et la potentielle du même.

Avant de commencer le calcul des équations du mouvement, nous allons décrire tous les paramètres qui ont été utilisés pendant les opérations ainsi que les différentes hypothèses. Il faut aussi remarquer que la partie de l'information de la modélisation a été extraite et contrasté de la source [1].

Tableau 3 : Constantes et variables utilisées dans la modélisation

Paramètre	Valeur et unités	Description
g	9,81 m/s^2	Accélération de la pesanteur
m	0,03 kg	Masse d'une roue et du rotor d'un moteur
R	0,04 m	Rayon d'une roue
J_θ	3,4·10^{-3} kg·m^2	Inertie d'une roue et du rotor d'un moteur
M	0,6 kg	Masse du corps
L	0,072 m	Distance CdG corps / roues
J_ψ	0,0011 kg·m^2	Inertie du corps / axe y

Liste des hypothèses :

1) Le pneu n'agit pas comme un ressort
2) Le gyropode est symétrique dans les axes *x* et *y*
3) Le CdG est au centre du gyropode

Le schéma du gyropode, à partir duquel le calcul a été fait, est montré ci-dessous. Dans le dessin, nous trouvons les principaux paramètres pour avoir une meilleure idée.

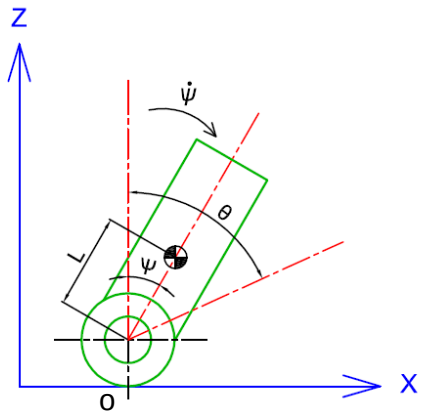


Figure 6 : Schéma du gyropode avec les paramètres

Le Lagrangien peut être exprimé comme la différence entre l'énergie cinétique *T* et la potentielle *V* du système. Pour cette raison, la première étape est de calculer ces énergies. Il faut calculer le corps et les roues. Le calcul est fait de façon paramétrique afin de pouvoir changer facilement les valeurs dans la simulation. Dans le cas où les roues pèsent beaucoup moins que le corps il est possible de négliger l'énergie de celles-ci si on dit que la masse et l'inertie sont égales à zéro.

Ci-dessous se trouvent les équations utilisées pour le calcul du Lagrangien et pour avoir le modèle mathématique. Le développement des expressions se trouve aussi dans le document [1] de la bibliographie.

L'énergie cinétique est formée par deux composantes, l'énergie cinétique de translation :

$$T_{Tr} = \frac{1}{2} \cdot m \cdot \dot{x}^2 \qquad \text{(Eq. 2.1)}$$

et celle de rotation :

$$T_{Ro} = \frac{1}{2} \cdot J_\theta \cdot \dot{\theta}^2 \qquad \text{(Eq. 2.2)}$$

Il est nécessaire de calculer l'énergie des roues et du corps, ce qui est plus difficile car cela nécessite de trouver la vitesse du centre de gravité (CdG) par un calcul vectoriel. L'énergie potentielle est celle ont des corps dû à la force de la pesanteur et c'est nécessaire l'exprimer par chaque partie du système qui a masse. Il existe aussi l'énergie potentielle élastique mais dans ce cas on ne considère pas les pneus comme des ressorts. Ces expressions sont, respectivement, les suivantes :

$$V_{grav} = m \cdot g \cdot z \qquad \text{(Eq. 2.3)}$$

$$V_{elas} = \frac{1}{2} \cdot k \cdot x^2 \qquad \text{(Eq. 2.4)}$$

Après avoir trouvé l'énergie cinétique et potentielle du système, il est possible d'écrire le Lagrangien :

$$L = T - V \qquad \text{(Eq. 2.5)}$$

Le Lagrangien (*L)* est utilisé pour pouvoir écrire l'équation de Lagrange. Avec cette expression il est possible de connaître le comportement du système en fonction des coordonnées généralisées qui sont, dans ce cas, θ et ψ. L'équation de Lagrange est la suivante :

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{q}_i}\right) - \left(\frac{\partial L}{\partial q_i}\right) \qquad \text{(Eq. 2.6)}$$

Si l'on remplace cette expression par le Lagrangien, la formule reste comme ci-dessous :

$$\frac{d}{dt}\left(\frac{\partial T}{\partial \dot{q}_i}\right) - \left(\frac{\partial T}{\partial q_i}\right) + \left(\frac{\partial V}{\partial q_i}\right) = 0 \qquad \text{(Eq. 2.7)}$$

Après avoir dérivé l'expression de Lagrange selon chaque coordonnée généralisée le système d'équations qui définit le comportement dans l'espace temporel est le suivant :

$$J_{11} \cdot \ddot{\theta} - J_{12} \cdot \dot{\psi}^2 \cdot \sin(\psi) + J_{12} \cdot \ddot{\psi} \cdot \cos(\psi) = 0 \qquad \text{(Eq. 2.8)}$$

$$J_{22} \cdot \ddot{\psi} + J_{12} \cdot \ddot{\theta} \cdot \cos(\psi) - M \cdot G \cdot L \cdot \sin(\psi) = 0 \qquad \text{(Eq. 2.9)}$$

Pour pouvoir travailler dans l'espace *S* il faut appliquer la transformé de Laplace sur ces expressions. Avant cela toutefois, il est nécessaire de linéariser le système, parce qu'il y a des fonctions non linéales comme le sinus et cosinus. Pour linéariser, il est nécessaire de considérer que l'angle du corps (ψ) est très petit. Cela signifie que, dans le fonctionnement réel, le gyropode ne peut pas dépasser ce petit angle, car il serait impossible de le contrôler. Avec cet angle ($\psi \approx 0$), il est possible d'écrire $\sin(\psi) = \psi$, $\cos(\psi) = 1$ et $\psi^2 = 0$

Après avoir appliqué ces simplifications le système d'équations devient :

$$\left.\begin{aligned} J_{11} \cdot \ddot{\theta} + J_{12} \cdot \ddot{\psi} = 0 \\ J_{22} \cdot \ddot{\psi} + J_{12} \cdot \ddot{\theta} - M \cdot g \cdot L \cdot \psi = 0 \end{aligned}\right\} \qquad \text{(Eq. 2.10)}$$

Pour simplifier les expressions suivantes, le terme J_{11} remplace $\left(2(m \cdot R^2 + J_\theta) + (M \cdot R^2)\right)$, le terme J_{12} remplace $(M \cdot R \cdot L)$ et le J_{22} remplace $\left(M \cdot L^2 + J_\psi\right)$.

De l'autre côté de l'égalité de l'expression de Lagrange, on doit écrire le travail des forces non conservatives, qui sont les dérivées du travail pour chaque coordonnée généralisée. Ces forces sont normalement dues aux frottements et à d'autres sollicitations et il est nécessaire de trouver une expression de la force en fonction de la commande. Dans ce cas, puisque la commande est la vitesse angulaire des roues, il n'était pas possible d'avoir une expression du type force par distance (travail ou couple) en fonction de cette commande. Cela signifie qu'on ne peut pas avoir la fonction de transfert en suivant cette méthode. Pour cette raison et puisque on voulait avoir des simulations, on a supposé que le gyropode a aussi les mêmes moteurs de courant continu que le rapport suivi. Avec cette considération, le modèle mathématique final est le suivant :

$$\left.\begin{aligned} J_{11} \cdot \ddot{\theta} + J_{12} \cdot \ddot{\psi} = -2 \cdot fm \cdot \dot{\theta} + 2 \cdot fm \cdot \dot{\psi} + 2 \cdot K_m \cdot u \\ J_{22} \cdot \ddot{\psi} + J_{12} \cdot \ddot{\theta} - M \cdot g \cdot L \cdot \psi = 2 \cdot fm \cdot \dot{\theta} - 2 \cdot fm \cdot \dot{\psi} - 2 \cdot K_m \cdot u \end{aligned}\right\}$$

Où K_m est la constant de proportion entre le couple et le voltage du moteur et *fm* le frottement visqueux.

Avec ce modèle écrit il est possible commencer à obtenir les matrices de l'espace d'état, ce qui nous permettra avoir les fonctions de transfert en fonction des entrées et sorties. Le gyropode est un système MIMO (*"multiple input multiple output* ") puisque on a deux entrées (le couple moteur *u* ce qui est proportionnel à la commande et le couple C_r, dû à le pilote sur le véhicule) et deux sorties (l'angle du corps ψ vers l'axe verticale et la position *x* de celui-ci).

Les équations d'état écrites dans le rapport [1] et sous forme matricielle sont les suivantes :

$$\begin{bmatrix} \dot{\theta} \\ \dot{\psi} \\ \ddot{\theta} \\ \ddot{\psi} \end{bmatrix} = \underbrace{\frac{1}{\Delta}\begin{bmatrix} 0 & 0 & \Delta & 0 \\ 0 & 0 & 0 & \Delta \\ 0 & -A_{32} & -A_{34} & A_{34} \\ 0 & A_{42} & A_{43} & -A_{43} \end{bmatrix}}_{\mathbf{A}} \cdot \begin{bmatrix} \theta \\ \psi \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} + \underbrace{\frac{1}{\Delta}\begin{bmatrix} 0 & 0 \\ 0 & 0 \\ B_3 & -J_{12} \\ -B_4 & J_{11} \end{bmatrix}}_{\mathbf{B}} \cdot \begin{bmatrix} u \\ cr \end{bmatrix}$$

$$\begin{bmatrix} x \\ \psi \end{bmatrix} = \underbrace{\begin{bmatrix} R & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}}_{\mathbf{C}} \cdot \begin{bmatrix} \theta \\ \psi \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} \cdot \begin{bmatrix} u \\ cr \end{bmatrix}$$

Figure 7: Équation d'état et équation de sortie

Ci-dessous se montrent la manière de calculer les valeurs correspondants a tous ces paramètres :

- $A_{32} = M \cdot g \cdot L \cdot J_{12}$
- $A_{34} = 2 \cdot fm \cdot (J_{22} + J_{12})$
- $A_{42} = M \cdot g \cdot L \cdot J_{11}$
- $A_{43} = 2 \cdot fm \cdot (J_{11} + J_{12})$
- $B_3 = 2 \cdot Km \cdot (J_{22} + J_{12})$
- $B_4 = 2 \cdot Km \cdot (J_{11} + J_{12})$
- $\Delta = J_{11} \cdot J_{22} - J_{12}^2$

Avec ces expressions il n'est pas encore possible connaitre les fonctions de transfert puisque il faut opérer avec ces deux pour n'avoir une plus facile sous forme vectorielle:

$$\dot{x}(t) = A \cdot x(t) + B \cdot u(t) \quad \text{(Eq. 2.11)}$$

$$S \cdot X(S) = A \cdot X(S) + B \cdot U(S) \quad \text{(Eq. 2.12)}$$

$$X(S) = (S \cdot I - A)^{-1} \cdot B \cdot U(S) \quad \text{(Eq. 2.13)}$$

$$y(t) = C \cdot x(t) + D \cdot u(t) \quad \text{(Eq. 2.14)}$$

$$Y(S) = C \cdot X(S) + D \cdot U(S) \quad \text{(Eq. 2.15)}$$

Si dans cette dernière équation on remplace X(S) pour l'expression Eq. 2.12 on a la formule des sorties :

$$Y = C \cdot (S \cdot I - A)^{-1} \cdot B \cdot U \quad \text{(Eq. 2.16)}$$

Ou Y est le vecteur des sorties (*x* et *ψ*) et U est le vecteur des entrées (*u* et *Cr*). Aussi, on appelle matrice de boucle ouvert à la multiplication $C \cdot (S \cdot I - A)^{-1} \cdot B$ et ça permettra avoir les fonctions de transfert de chaque sortie en fonction de chaque entrée, c'est-à-dire, quatre fonctions.

Finalement, pour connaitre rapidement ces fonctions de transfert, on utilise Matlab pour les avoir et épargner le calcul. Préalablement il faut définir chaque paramètre et calculer chaque matrice dans un script Matlab. Les fonctions de transfert sont obtenues automatiquement grâce à une commande qui permet lire chaque position de la matrice de boucle.

Ci-dessous se montrent toutes les fonctions de transfert obtenues après avoir opéré et les simplifier :

$$\frac{X(S)}{U(S)} = \frac{-2.72 \cdot S^2 + 78.76 \cdot S - 1989.32}{-S^4 - 22 \cdot S^3 + 273 \cdot S^2 + 1162 \cdot S} \quad \text{(Eq. 2.17)}$$

$$\frac{X(S)}{Cr(S)} = \frac{-6.912e-5 \cdot S^2 + 0.00019056 \cdot S + 0.00003152}{-S^4 - 22 \cdot S^3 + 273 \cdot S^2 + 1162 \cdot S} \quad \text{(Eq. 2.18)}$$

$$\frac{\psi(S)}{U(S)} = \frac{163 \cdot S + 65}{-S^3 - 22 \cdot S^2 + 273 \cdot S + 1162} \quad \text{(Eq. 2.19)}$$

$$\frac{\psi(S)}{Cr(S)} = \frac{-0.001124 \cdot S - 0.004764}{-S^3 - 22 \cdot S^2 + 273 \cdot S + 1162} \quad \text{(Eq. 2.20)}$$

Ces fonctions seront utilisées dans l'environnement de simulation pour avoir les réponses selon le modèle mathématique.

2.3. SIMULATION

Aujourd'hui il y a des logiciels pour simuler tout type de systèmes et processus et les plus connus sont ceux qui sont liés à l'ingénierie mécanique, électronique et automatique. Dans le domaine de la mécanique il y a des outils CA (de l'anglais *Computer Aided*) comme le CAD, CAM, CAE, etc. des logiciels d'éléments finis ainsi que des autres pour calculer des structures. En revanche, ceux qui sont utilisés pour l'électronique sont ceux qui aident à calculer et à construire des circuits et ceux qui permettent de simuler le contrôle de systèmes et processus.

Dans ce projet on emploiera Simulink, un environnement de programmation contenu dans le logiciel Matlab. L'objectif est de simuler le comportement du gyropode pour apprécier la réponse du système. Après avoir trouvé la meilleure configuration du régulateur PID, il sera possible de changer le code du gyropode pour le faire travailler comme dans la simulation.

2.3.1. Dessin du gyropode

Pour mieux pouvoir observer le comportement du gyropode pendant la simulation dans l'environnement Simulink, il était nécessaire de dessiner une approche du robot en créant des fonctions sous code Matlab. Avec ce logiciel, il est possible de dessiner quelques formes et de les remplir. Ensuite, en utilisant des boucles ou exécutions interminables et en changent des variables du robot, il est possible de regarder le mouvement de celui-ci.

La première étape était de trouver une fonction pour dessiner des cercles et pour les remplir. Deuxièmement, le code de la fonction qui dessine a été écrit. Ce code appelle la fonction du cercle et dessine chaque partie du gyropode en fonction de la position du robot (x et y), l'angle du corps du robot (ψ) et l'angle des roues (θ). Ci-dessous il y a

quelques parties de ces codes pour comprendre meilleur comment dessiner les parties et comment les faire tourner :

```
function h = cercle(center,r,N,color)
filledCircle(CENTER,R,N,COLOR)
coder.extrinsic('fill')
THETA=linspace(0,2*pi,N);
RHO=ones(1,N)*r;
[X,Y] = pol2cart(THETA,RHO);
X=X+center(1);
Y=Y+center(2);
h=fill(X,Y,color);
axis square;
```

Cette fonction a été téléchargée de la source [17] et permet dessiner un cercle en introduisant la position du centre (sous forme de vecteur), le rayon, le numéro de points qui forment le cercle et le couleur qu'on veut dans l'intérieur. Le code suivant montre comment appeler à cette fonction :

```
%Wheels' definition
cercle([x,y],6.5/2,1000,'k'); hold on
cercle([x,y],5.5/2,1000,[0 0 0.5]); hold on
cercle([x,y],4.5/2,1000,'k'); hold on
```

Cette première partie dessine la roue. La première fonction cercle dessine un cercle derrière (le pneu), après la jante de couleur bleu (code RGB de 0 à 1, ça veut dire que blanc serait [1 1 1] et noir [0 0 0]) et encore un cercle noir pour bien différer les rayons.

Pour dessiner un rectangle il faut définir un vecteur avec toutes las coordonnés X du rectangle et une autre avec les Y. Ces coordonnés contenant les paramètres correspondants à la position du centre et à l'angle du corps. Pour cette raison, quand la position du gyropode change, le dessin suive le changement. Pour connaitre les expressions il faut seulement faire un calcul trigonométrique avec un rayon incliné et avec les dimensions de celui (5/2 de largeur et 5.5 de longueur, par exemple). Après avoir fait le même pour chaque point (en X et en Y) les vecteurs sont bien définis et il manque seulement le remplir avec l'avant-dernière ligne. Il faut écrire les vecteurs, ensuite le couleur qu'on veut et l'épaisseur de la ligne qui définit la forme.

```
pxr1=[x-4.5/2*sin(theta)-0.2*cos(theta) x-
4.5/2*sin(theta)+0.2*cos(theta) x+4.5/2*sin(theta)+0.2*cos(theta)
x+4.5/2*sin(theta)-0.2*cos(theta)];

pyr1=[y+4.5/2*cos(theta)-0.2*sin(theta)
y+4.5/2*cos(theta)+0.2*sin(theta) y-4.5/2*cos(theta)+0.2*sin(theta) y-
4.5/2*cos(theta)-0.2*sin(theta)];
fill(pxr1,pyr1,[0 0 0.5],'LineWidth',1); hold on
```

Le résultat du dessin du rayon avec ces lignes de code est le suivant si on définit un angle de -15º, une "*x*" égale à 1,5 et une "*y*" égale à 1,2 :

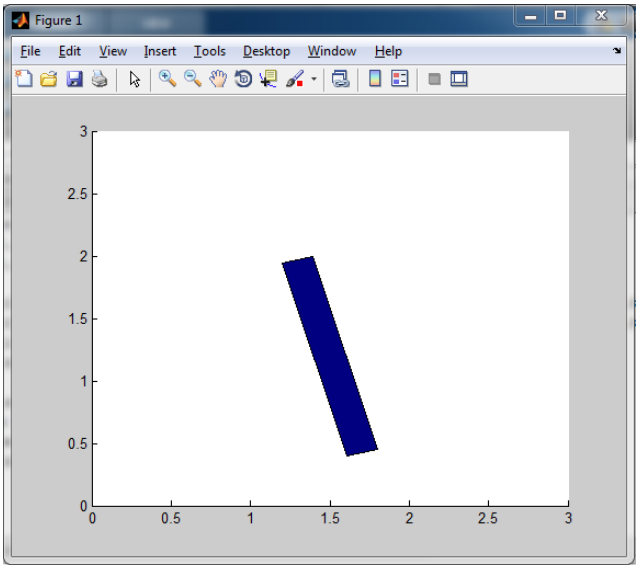


Figure 8: Dessin d'un rayon du gyropode

Finalement, la commande *hold on* est strictement nécessaire pour maintenir le dessin entre une boucle et la suivante et pouvoir regarder l'animation. Pour dessiner les autres rayons il faut seulement copier les vecteurs et ajouter +pi/4, +pi/2 et –pi/4 après l'angle *theta.* Définir des nouveaux noms pour les vecteurs n'est pas nécessaire dû à la commande *hold on*. Avec ces instructions c'est possible dessiner le gyropode.

Dans un premier temps, pour pouvoir vérifier que le dessin faisait le mouvement correctement, il était nécessaire de définir un nouveau script pour faire changer les coordonnées x, ψ et θ, et pour introduire les boucles qui permettent de faire l'animation. Ci-dessous se trouve un des boucles utilisés et le résultat de cette codification :

```
psi=0;
x=0;
r=1;
theta=-x/r;
a=1*2*pi/360;
for i=1:100
    y=1.2;
    psi=(15*2*pi/360)*sin(4*a);
    dessingyro(x,y,psi,theta)
    x=x+0.05;
    a=a+(1*2*pi/360);
    theta=-x/r;
end
```

Au début il faut initialiser quelques variables pour situer le gyropode dans une correcte position. La variable “*a*” est utilisé pour faire changer l’angle et pour changer les unités de dégrées à radians. L’amplitude du mouvement angulaire du corps est 15º mais écrit en radians. On observe que la variable "*x*" change sa valeur dans l’intérieur de la boucle et que la variable "*y*" est toujours la même. Il faut calculer *theta* pour faire tourner les rayons.

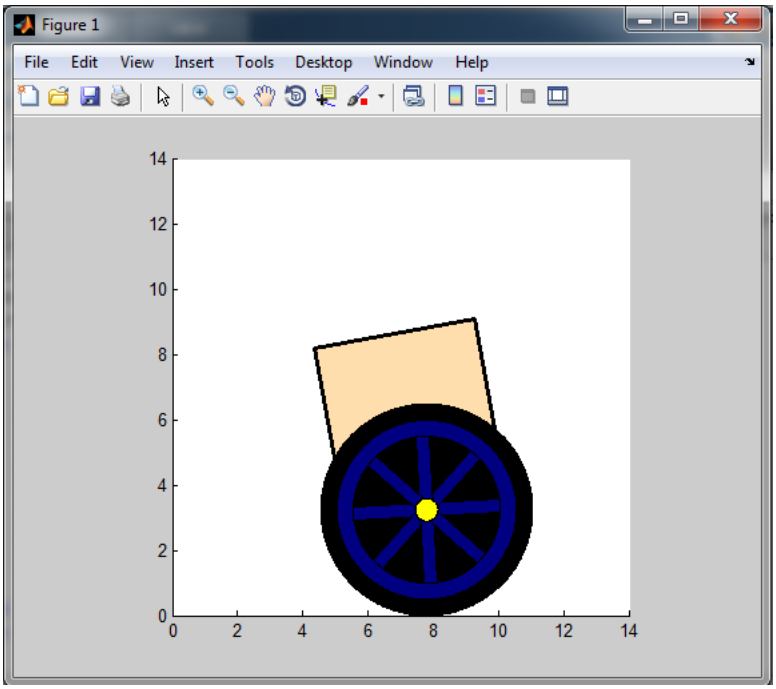


Figure 9: Dessin du gyropode sous Matlab

Puisque la simulation du gyropode est sous Simulink, il a fallu relier cet environnement aux fonctions qui dessinent le robot. Cela fut possible grâce à un bloc dénommé "*Matlab function"*. Avec ce bloc, il est effectivement possible d’ajouter les inputs en forme de constantes, sinus, rampe, etc.

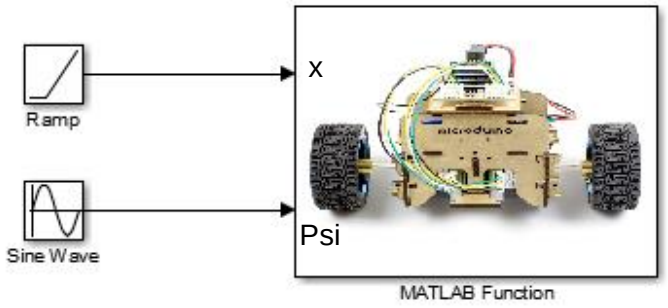


Figure 10: Schéma pour le dessin du gyropode sous Simulink

2.3.2. Schémas du modèle

Après avoir le modèle mathématique et toutes les fonctions de transfert, il est possible commencer à faire le schéma de control du gyropode sous Simulink. Celui doit respecter l'équation Eq. 2.16 pour bien linker chaque fonction de transfert avec les entrées correspondantes. De plus, on a mis des contrôleurs PD et P pour contrôler l'angle du corps du gyropode et aussi la position à laquelle il doit aller. Le schéma final avec tous les blocs connectés se montre à la Figure 11.

Dans ce cas, la source qui donne la valeur désiré de la position en fonction du temps n'est pas une fonction sinus sinon une "*repeating sequence interpolated*" car ça permet de faire les mouvements plus doucement et regarder mieux le mouvement du robot.

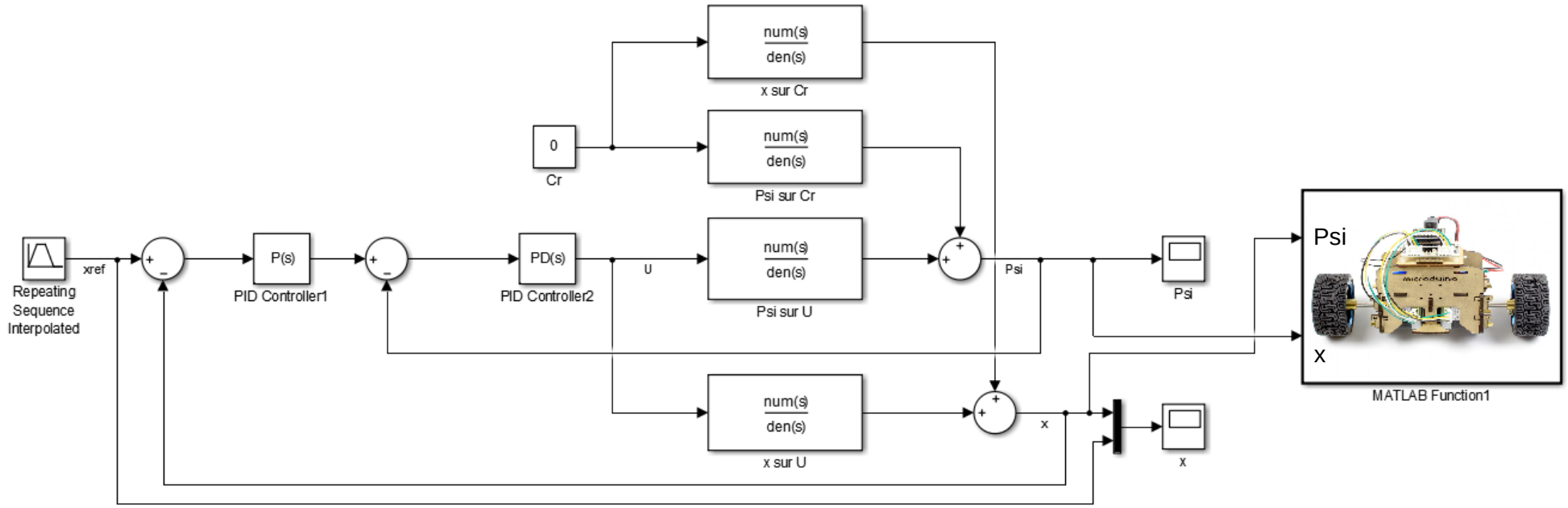


Figure 11: Schéma de simulation et contrôle du gyropode

2.3.3. Réglage des contrôleurs PD et P

Dans ce petit paragraphe il y aura la procédure et l'explication de comment connaitre les valeurs qui composent le régulateur PD et P de ce système. Dans ce cas, puisqu'on veut contrôler la position du gyropode et l'angle du corps pour qu'il ne tombe pas, il faudra régler les deux.

Pour avoir les valeurs du PD qui régule l'angle ψ on a fait le schéma avec seulement le boucle de celui en fonction de la commande "*U*". Ce schéma est le suivant :

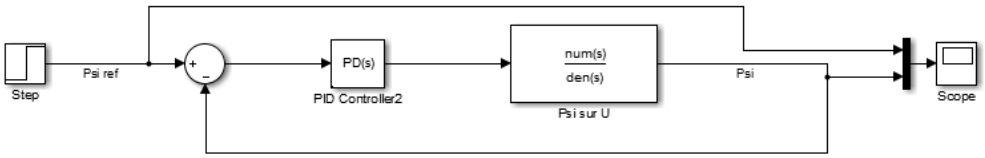


Figure 12: Schéma de la boucle de contrôle de l'angle "Psi"

Après avoir le schéma finit, on peut faire double clic sur le bloc PID et appuyer encore sur le bouton "*Tune*". Avec ça, Matlab calcule automatiquement les valeurs qui permettent avoir une réponse stable et qui suive la commande. Il faut ajouter que ces valeurs ne sont pas les optimales et qu'on peut jouer pour avoir une réponse plus rapide, plus précise, etc. Finalement, les valeurs choisis sont les suivants :

- *Proportional* (P) : -73,677
- *Derivative* (D) : -4,918
- *Filter coefficient* (N) : 9157,651

Pour régler le deuxième contrôleur n'était pas possible réalisée la même procédure car Matlab ne pouvait pas proposer des valeurs. Pour cette raison on a obtenu cette valeur par essai et erreur. Finalement, la valeur qui a fait suivre mieux la commande de position était *Proportional (P) :* -0,05.

Dans les deux cas, on n'a pas utilisé l'action intégrale parce que dans les fonctions de transfert il y a déjà l'action intégrale $\frac{1}{s}$.

2.3.4. Comment lancer une simulation

Dans ce paragraphe on trouve les indications à suivre pour connaitre tous les fichiers nécessaires pour avoir une simulation du gyropode. D'abord, il faut dire que tous ces fichiers doivent être dans le même dossier car sinon Matlab et Simulink présenteront des erreurs. Aussi, il faut avoir ce dossier ouvert sous Matlab.

Les fichiers qui dessinent le gyropode sont le *cercle.m* et *dessingyro.m*. Il n'est pas nécessaire de les ouvrir et exécuter, seulement être dans le dossier car Matlab les utilise automatiquement. Après, la photo *gyro.jpg* est utilisé pour changer le masque du bloc qui contient le modèle du gyropode.

Ensuite, il faut ouvrir le fichier *Gyro_fct_transfert.m* et le lancer. Ça permet d'avoir les valeurs des paramètres au *"workspace"* et les utiliser après sous Simulink. Finalement il faut ouvrir le fichier *controlgyro.slx* et commencer la simulation.

2.4. RÉSULTATS

Une fois les processus d'assemblage et de programmation terminés, et tous les changements appliqués, le robot a commencé à fonctionner. Avant de se lever, le robot est resté incliné quelques secondes, peut-être à cause d'un problème de réchauffement, puis s'est levé et maintenu en position verticale réglée. Le gyropode peut contrer les perturbations effectuées par une force externe ou des obstacles afin de maintenir la position décrite antérieurement.

Le robot auto-équilibré a deux points d'équilibre. Le premier est celui où l'angle est à 180º, comme il est montré dans la Figure 13, et le second apparaît lorsque l'angle est 0º. La différence entre ces deux points d'équilibre est que le premier est stable et le second a une stabilité critique. Cela signifie que, dans le premier cas, une force externe appliquée au gyropode va le faire tourner et il changera l'angle du corps, mais il retournera à 180º sans aucun contrôleur car le centre de gravité n'est pas dans l'arbre principal. Néanmoins, si la même force est appliquée au gyropode quand il est sur une position de 0º, il tombera dans la position stable qui est de 180º. La seule façon de le maintenir dans la position 0º est en utilisant un contrôleur PID.

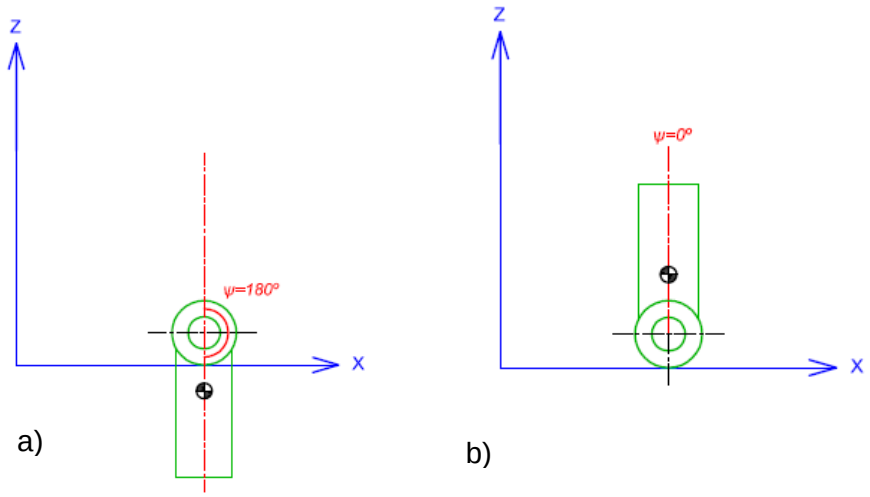


Figure 13: Le deux points d'équilibre a) Stable b) Stabilité critique

Quand le robot est en position verticale, il corrige sa position constamment et ne s'arrête jamais. La raison pour laquelle il est en mouvement constant est que le système n'est pas idéal. Si la table était parfaitement horizontale, s'il n'y avait pas d'air, si les deux moteurs pas à pas fonctionnaient de manière identique, etc., le robot resterait toujours dans une position verticale et avec un angle égal à zéro. La première différence entre l'erreur et le point de consigne est importante, et il doit la corriger le plus rapidement possible, c'est pourquoi il change d'angle brusquement. Cette action réduit la différence entre l'erreur et le point de consigne car la position actuelle a un petit angle, proche de zéro.

Après toute la procédure de modélisation et simulation, on pouvait avoir des résultats des simulations. Ceux-ci sont possibles après avoir bien réglé les contrôleurs PID. A continuation se montrent les graphiques obtenues après quelques seconds de simulation. Cette première figure montre la position "x" désiré (couleur violet) et la position réelle (ligne rouge). On peut regarder que le robot suive la position désiré mais avec un pic assez grand. Aussi, le robot montre un peu de retard puisque la ligne rouge est toujours derrière la violet.

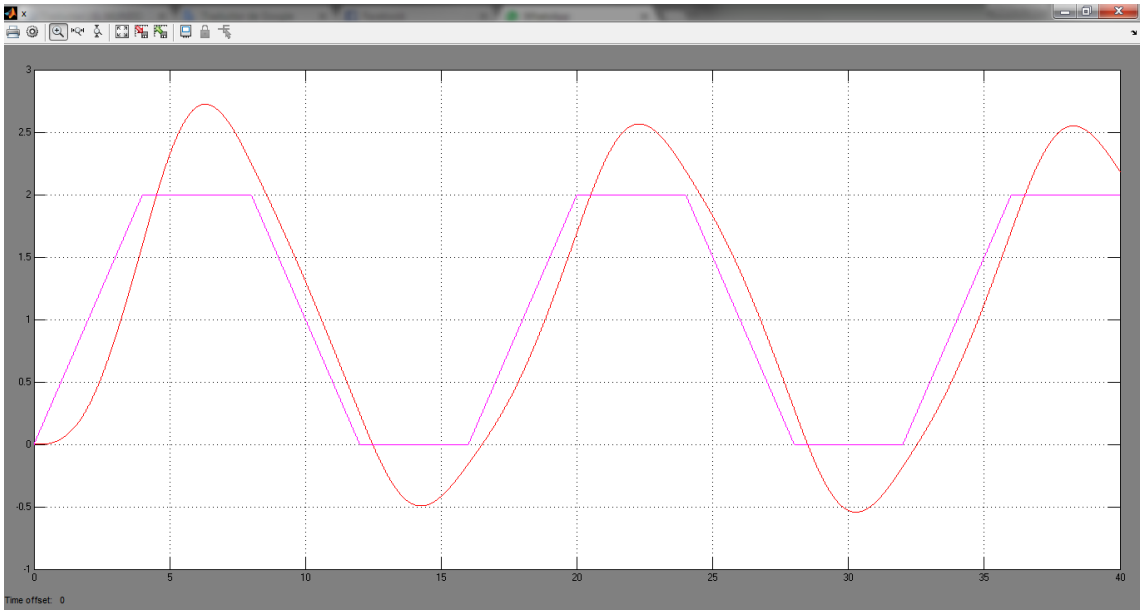


Figure 14: Position réelle (rouge) et désirée (violet) du gyropode

D'autre côté il y a la variation de l'angle du corps en fonction du temps. Cet angle change sa valeur parce que le robot doit changer sa position. Ça fait que le mouvement des roues pour arriver à la position désiré change l'angle ψ. La variation de cet angle en fonction du temps se montre ci-dessous :

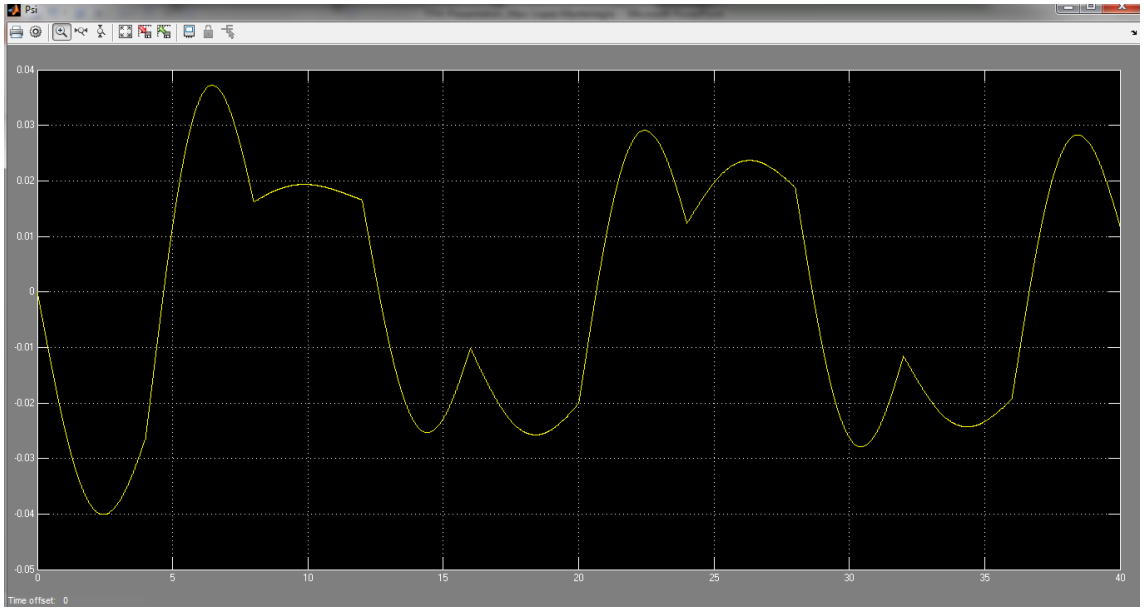


Figure 15: Évolution de l'angle du corps en fonction du temps

Avec cette graphique c'est possible regarder comment l'angle change au même temps que la position. Si on compare les deux il est possible comprendre que le changement de l'angle est dû à le changement de position.

2.5. CONCLUSION

Bien que faire fonctionner le gyropode aurait dû être facile, il y avait quelques problèmes qui ont retardé l'assemblage et les processus de programmation. Après avoir réfléchi et essayé certaines solutions possibles, ces problèmes ont été résolus et le robot a finalement marché. Comme expliqué aux paragraphes 2.1.3 et 2.1.4, ces solutions ont été les plus rapides puisque contacter le fournisseur afin de changer certaines parties aurait pris quelques jours ou peut-être des semaines. Voici, ci-dessous, un tableau avec tous les problèmes rencontrés pendant cette première expérience et les solutions appliquées.

Tableau 4: Liste des différents problèmes, des solutions prises et de l'état final

Nº	Problème	Solution retenue	État
1	Court-circuit lors de la connexion de la batterie	Ouverture du connecteur et réparation	**Résolu**
2	Taille du code supérieure à la mémoire du noyau	Simplification du code en supprimant une fonction non nécessaire	**Résolu**
3	Le code n'a pas pu être transféré au noyau	Utiliser un autre type de noyau extrait d'un autre robot (CoreRF)	**Résolu**

Le deuxième problème a été résolu, mais la solution n'était pas nécessaire car, finalement, un Core différent, avec 128KB au lieu de 32KB, a été utilisé. Cependant, le robot a finalement travaillé avec le code qui n'a pas la librairie RF; Cela signifie que le gyropode peut fonctionner avec un nouveau CoreUSB et sans cette librerie.

En dépit du fait que le robot oscille tout le temps, il a une réponse très adaptée et marche parfaitement. Il est capable d'absorber les perturbations et de retrouver la position attendue. On peut conclure qu'après avoir résolu tous les problèmes liés à la mise en œuvre du gyropode, celui-ci fonctionne correctement et qu'il n y a pas plus erreurs.

En ce qui concerne à la partie de modélisation et simulation il faut remarquer que le fait d'utiliser moteurs pas à pas pour le fonctionnement du gyropode a marqué plus de travail à faire puisque c'était impossible avoir la fonction de transfert directement et on devait chercher une alternative, comme par exemple essayer de connaitre les opérations qui calcule le PID dans le code du gyropode pour donner la meilleure réponse.

Finalement on a décidé faire les simulations selon le modèle avec les moteurs de courant continu et ça a marché assez bien. L'obtention des fonctions de transfert était rapide grâce à l'utilisation de Matlab et aussi le réglage des contrôleurs PID. Malgré ça, il est possible d'avoir un réglage plus efficient si on calcule manuellement les valeurs du PID mais le proposé par Simulink était suffisant pour regarder comment il se bouge bien et comment il suive la commande d'entrée.

Aussi il faut remarquer que le mouvement du gyropode pendant la simulation n'est pas totalement fluide sous Simulink. Pour résoudre ça il faut changer le "*sample time*" dans le bloc qui dessine. Après jouer avec ce temps, finalement la simulation marche presque au temps réel et on peut bien apprécier le mouvement linéal et angulaire.

3. QUADCOPTER

3.1. ASSEMBLAGE ET TEST

3.1.1. Introduction du quadcopter

Une des technologies qui aujourd'hui attire l'attention des personnes et qui est beaucoup utilisé c'est le drone. Drone c'est le mot plus commun pour décrire un aéronef sans humain à bord et aussi connu en anglais comme UAV (Unnamed Aerial Vehicle).

Bien que le drone soit une invention du siècle XIX, il n'a pas été sérieusement développé jusqu'au le début du XXème dû à la Première Guerre Mondiale. Le fait d'utiliser des drones permettait de ne pas risquer la vie des soldats. Actuellement il y a beaucoup d'autres applications, en plus de celles militaires, où les drones peuvent être utilisés. Ci-dessous quelques exemples :

- Agriculture
- Opérations de secours
- Recherche scientifique
- Messagerie
- Transport des personnes
- Photo et vidéo-reportage professionnel

En plus de tout cela, il y a aussi la possibilité d'avoir des drones moins chers pour les utiliser dans la vie domestique comme par exemple pour faire des photos, jouer, s'amuser, etc. Afin d'éviter que n'importe quelle personne puisse contrôler n'importe quel drone et pour garantir la sécurité des utilisateurs et des tierces personnes il y a une réglementation laquelle doit être satisfait. Il y a certains cas, dépendant du type de drone, pour lequel il faut avoir une license spécial.

Ce rapport inclut la description des kits Microduino acquis contenant toutes les pièces du quadcopter, la procédure d'assemblage, ainsi que des difficultés et d'autres aspects importants. L'objectif de ce document c'est expliquer les premiers pas réalisés sur le kit du Quadcopter et la première expérience.

3.1.2. Le quadcopter

Le quadcopter c'est un type de drone qui dispose de quatre rotors. À la différence du hélicoptère, qu'utilise le principal rotor pour monter ou descendre et l'autre dans la queue pour tourner, le quadcopter fait tourner deux hélices dans le sens des aiguilles d'une montre et les autres deux dans le sens inverse. Les quatre rotors sont distribués sous forme de croix ou de X comme il est vu dans la Figure 16. Selon le web page [8], que c'est le tutoriel du quadcopter, le mode X c'est plus difficile de contrôler mais c'est plus flexible.

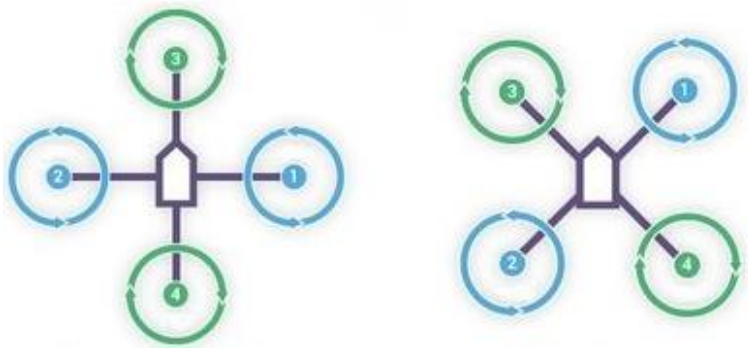


Figure 16: Distribution des hélices a) sous forme croix et b) sous forme X

Les rotors numéro 1 et 2 tournent dans le sens inverse des aiguilles d'une montre et le 3 et 4 dans le sens des aiguilles. Si tous les rotors ont la même vélocité de rotation, le robot est stable et tend à monter si cette vitesse est suffisamment grande.

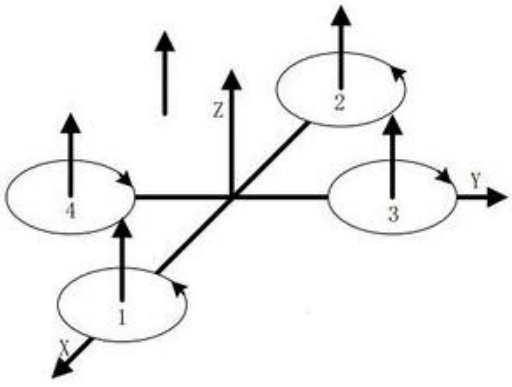


Figure 17 : Schéma du mouvement vertical

En outre, il y a aussi le mouvement frontal et le latéral. Le premier est possible quand la vélocité de l'arrière (rotor numéro 1) c'est supérieure à la de l'avant (rotor numéro 2). C'est nécessaire de maintenir la vitesse du 3 et 4 pour garantir la stabilité du robot. Par

contre, si la vélocité du 1 et 2 c'est la même et la du 4 augmente, le quadcopter va tomber a droit et initier le mouvement latéral.

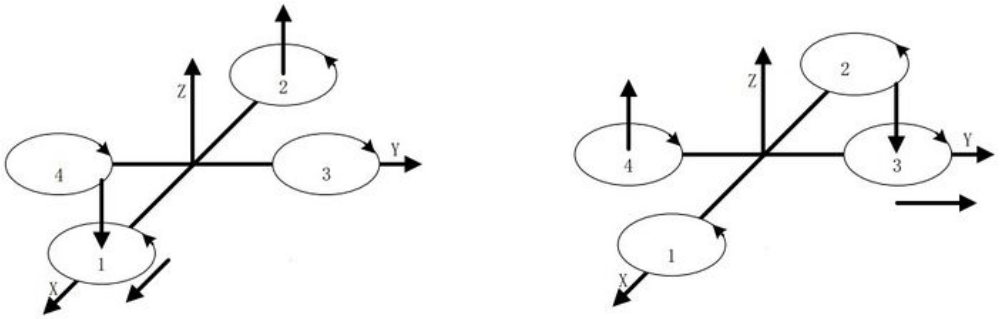


Figure 18 : Schémas du mouvement frontal et latéral respectivement

Ces trois mouvements : le vertical, le frontal et le latéral ; sont des mouvements linéaires qui suivent chacun des trois axes (Z, X et Y respectivement). Il y a aussi des mouvements de rotation sur chaque axe et ces sont : le *yaw*, le *roll* et le *pitch* en se référant encore aux les axes Z, X et Y. Ils sont aussi connus comme lacet, roulis et tangage en langage aéronautique.

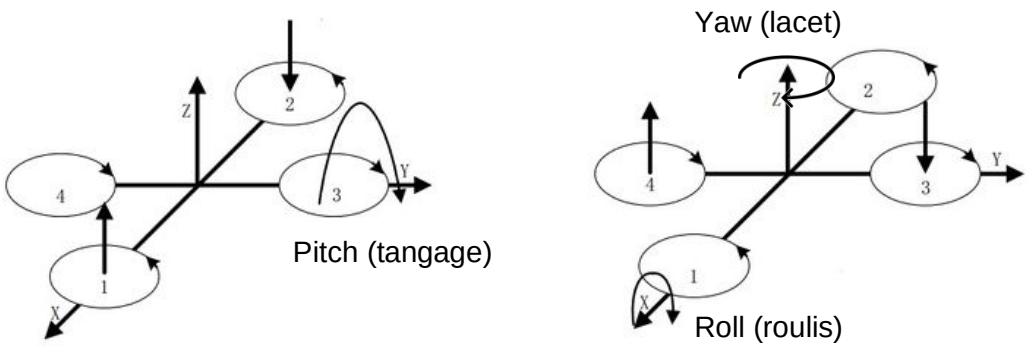


Figure 19 : Roll et pitch mouvements respectivement

Le quadcopter, de la même manière que le gyropode, utilise des différents Microduinos modules pour se communiquer et pour bien fonctionner. Comme il a été commenté dans le Rapport 1 du gyropode, Microduino c'est un type de carte Arduino mais d'une taille réduite. Tous ces modules, ainsi que toutes les autres parties comprises dans le quadcopter kit, se montrent dans le tableau 1 et la

Tableau 5: Liste des matériaux du quadcopter

Nº	Description	Quantité
1	Microduino-CoreRF	1
2	Microduino-USBTTL	1
3	Microduino-Motion	1
4	Microduino-QuadCopter	1
5	2.4G Antenne	1
6	USB Câble	1
7	Châssis	1
8	Batterie	1
9	Vis	1
10	Tournevis	1
11	Hélice	4

Figure 20: Matériaux du kit codifiés selon le Tableau 5

Avec toutes les parties c'était possible commencer à les assembler et construire le quadcopter. À la Figure 21Figure 20 se montre l'état du robot après du processus de montage

Figure 21: Quadcopter fini après le montage

3.1.3. Le joypad

Puisque le programme fourni par le web page du quadcopter ne permet pas l'auto stabilisation du robot, c'est nécessaire utiliser une télécommande pour se communiquer et donner des ordres au robot. C'est pour ça qu'on devait acheter un autre microduino kit : le joypad. Le joypad utilise aussi un Microduino-CoreRF et une antenne pour se communiquer avec le quadcopter. Ci-dessous se trouve le tableau avec tous les matériels et aussi une figure avec ces components.

Tableau 6 : Liste des matériaux du joypad

Nº	Description	Quantité
1	Microduino-CoreRF	1
2	Carte principale Joypad	1
3	Couvercle Joypad	1
4	Écran	1
5	Antenne	1
6	Vis	12
7	Tournevis	1
8	Câble de connection	1
9	Joysticks et bottons	2, 4

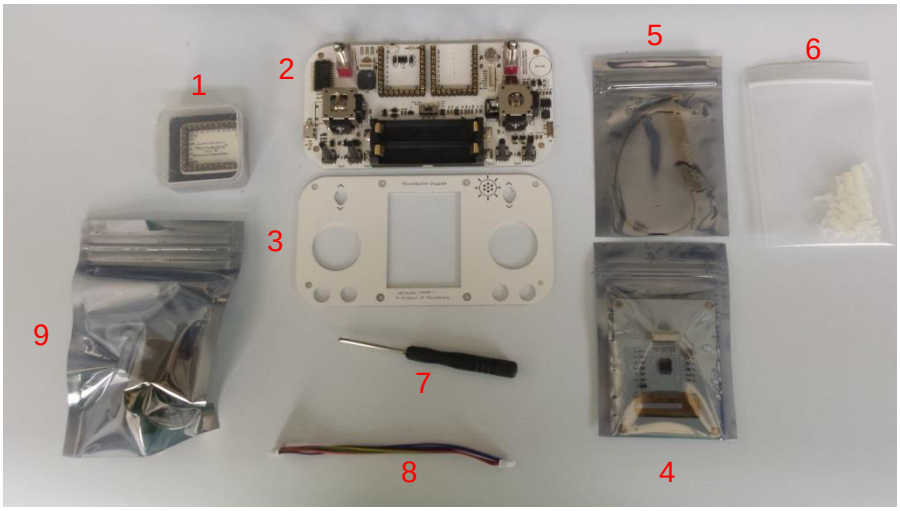


Figure 22: Matériaux du kit joypad codifiés selon le Tableau 6

Pour utiliser le joypad c'est nécessaire des piles ou une batterie. Selon le web page [9], c'est possible placer une batterie 1044 Li-ion (3,7V) ou deux piles AAA (1,5V). Dans ce cas, bien que la page recommande la batterie, on a utilisé des piles parce que c'était plus facile l'obtenir. Le tutoriel dit que le joypad marche aussi avec deux piles AAA mais remarque qu'il y aura un LED rouge au lieu d'un vert. Puisque il y a aussi une prise USB dans la carte principale du joypad, c'est possible l'alimenter avec un câble USB connecté à l'ordinateur pour charger la batterie. Le joypad fini se montre à la Figure 23.

Figure 23: Joypad fini sans alimentation

Selon le tutoriel, une fois connectés, l'écran devait s'éclairer et montrer l'option pour accéder à la configuration. Toutefois elle restait blanche sans la possibilité de ne faire rien. Ça n'était pas un problème du programme sinon du CoreRF car toujours c'est nécessaire d'appuyer le bouton de reset dans le CoreRF pour que ça marche. Finalement c'était possible de configurer le joypad selon le tutoriel. Ci-dessous se trouve une figure pour localiser le bouton reset.

Figure 24: Joypad vue depuis l'autre côté et localisation du bouton reset

Pour comprendre meilleur le fonctionnement du télécommande on trouvera ci-dessous un schéma des boutons et une petite description de chaque fonction.

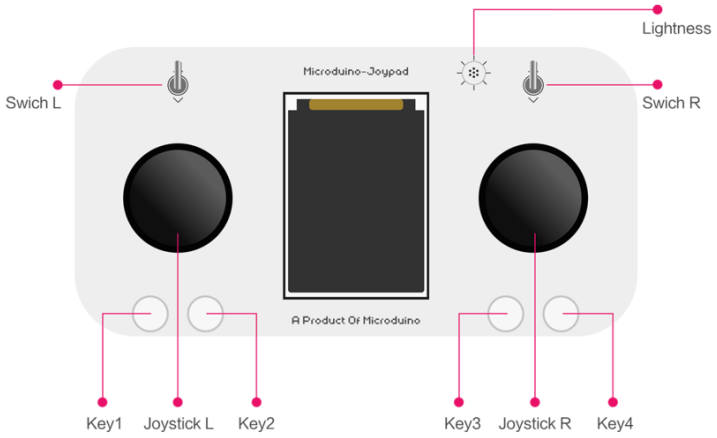


Figure 25: Schéma des principales contrôles du joypad

- Key1-key4 : AUX1 (canal 5) – AUX4 (canal 8)
- Joystick L : Contrôle de l'accélérateur et du *yaw*
- Joystic R : Contrôle du *roll* et du *pitch*
- Switch L : Blocage de l'accélérateur
- Switch R : « Small rudder » mode

3.2. MODÉLISATION

De la même manière que le gyropode, pour simuler le quadcopter c'est nécessaire avoir les équations qui définissent le comportement physique du robot pour que le programme puisse dessiner le mouvement du robot le plus réel possible. Dans ce cas, le document suivi pour avoir le modèle est encore le [2] et en cas de doute il est conseillé de le lire en détail car il y a plus d'information et calculs.

Le principe est le même, construire l'équation de Lagrange pour arriver à avoir trois équations lesquelles permettent obtenir la valeur des accélérations angulaires sur chaque axe en fonction de certaines vitesses et autres paramètres physiques liés au robot. Pour simplifier le calcul c'était nécessaire faire certaines hypothèses lesquelles se montrent ci-dessous :

- La structure du système est supposée rigide.
- La structure est supposée parfaitement symétrique donc la matrice d'inertie sera diagonale.
- La portance et la traînée de chaque moteur sont proportionnelles au carré de la vitesse ce qui se rapproche énormément du comportement aérodynamique du système réel.

Désormais, l'angle correspondant au lacet (*yaw)* serait la lettre grecque ψ, la du tangage (*pitch*) serait θ et la du roulis (*roll)* serait ∅. Pour mieux comprendre, le dessin ci-dessous montre ces angles et les lettres grecques correspondants :

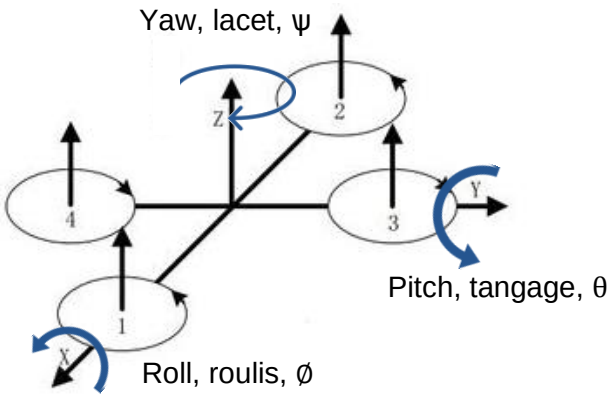


Figure 26: Nomenclature des angles de rotation

Ci-dessous se trouve partie du calcul réalisé dans la source [2] pour pouvoir comprendre comment il obtient le modèle final.

La première étape est écrire les trois matrices de rotation et les multiplier :

$$R(x,\phi)=\begin{pmatrix}1 & 0 & 0\\ \cos\phi & -\sin\phi & 0\\ \sin\phi & \cos\phi & 0\end{pmatrix} \quad R(y,\theta)=\begin{pmatrix}\cos\theta & 0 & \sin\theta\\ 0 & 1 & 0\\ -\sin\theta & 0 & \cos\theta\end{pmatrix} \quad R(z,\psi)=\begin{pmatrix}\sin\psi & -\sin\psi & 0\\ \sin\psi & \cos\psi & 0\\ 0 & 0 & 1\end{pmatrix}$$

$$R(\phi,\theta,\psi)=\begin{pmatrix}\cos(\psi)\cos(\theta) & \cos(\psi)\sin(\theta)\sin(\phi)-\sin(\psi)\cos(\phi) & \cos(\psi)\sin(\theta)\cos(\phi)+\sin(\psi)\sin(\phi)\\ \sin(\psi)\cos(\theta) & \sin(\psi)\sin(\theta)\sin(\phi)+\cos(\psi)\cos(\phi) & \sin(\psi)\sin(\theta)\cos(\phi)-\sin(\phi)\cos(\psi)\\ -\sin(\theta) & \cos(\theta)\sin(\phi) & \cos(\theta)\cos(\phi)\end{pmatrix} \quad \text{(Eq. 3.1)}$$

Soit $[\vec{X},\vec{Y},\vec{Z}]$ une base orthonormée constituant un repère fixe. Si le micro hélicoptère subit trois rotations successives selon les angles aéronautiques on a alors :

$$r_{x,y,z}(x,y,z)=R(\phi,\theta,\psi)\begin{pmatrix}x\\ y\\ z\end{pmatrix}$$

$$\begin{cases}r_x(x,y,z)=\cos(\psi)\cos(\theta)x+[\cos(\psi)\sin(\theta)\sin(\phi)-\sin(\psi)\cos(\phi)]y+[\cos(\psi)\sin(\phi)\cos(\phi)+\sin(\psi)\sin(\phi)]z\\ r_y(x,y,z)=\sin(\psi)\cos(\theta)x+[\sin(\psi)\sin(\theta)\sin(\phi)+\cos(\psi)\cos(\phi)]y+[\sin(\psi)\sin(\phi)\cos(\phi)-\sin(\psi)\cos(\phi)]z\\ r_z(x,y,z)=-\sin(\theta)x+\cos(\theta)\sin(\phi)y+\cos(\theta)\cos(\phi)z\end{cases} \quad \text{(Eq. 3.2)}$$

Ensuite, il faut dériver ces expressions selon le temps pour avoir les vitesses correspondantes :

$$V_x(x,y,z)=[-\dot{\theta}\sin(\theta)\cos(\psi)-\dot{\psi}\sin(\psi)\cos(\theta)]x+$$
$$[-\dot{\psi}\sin(\psi)\sin(\theta)\sin(\phi)+\dot{\theta}\cos(\psi)\cos(\theta)\sin(\phi)+\dot{\phi}\cos(\psi)\sin(\theta)\cos(\phi)-\dot{\psi}\cos(\psi)\cos(\phi)+\dot{\phi}\sin(\phi)\sin(\psi)]y$$
$$+[-\dot{\psi}\sin(\psi)\sin(\theta)\cos(\phi)+\dot{\theta}\cos(\psi)\cos(\theta)\cos(\phi)-\dot{\phi}\cos(\psi)\sin(\theta)\sin(\phi)+\dot{\psi}\cos(\psi)\sin(\phi)+\dot{\phi}\cos(\phi)\sin(\psi)]z$$

$$V_y(x,y,z)=[\dot{\psi}\cos(\psi)\cos(\theta)-\dot{\theta}\sin(\theta)\sin(\psi)]x+$$
$$[-\dot{\psi}\cos(\psi)\sin(\theta)\sin(\phi)+\dot{\theta}\cos(\theta)\sin(\psi)\sin(\phi)+\dot{\phi}\sin(\psi)\sin(\theta)\cos(\phi)-\dot{\psi}\sin(\psi)\cos(\phi)-\dot{\phi}\sin(\phi)\cos(\psi)]y+$$

$$[\dot{\psi}\cos(\psi)\sin(\theta)\cos(\phi)+\dot{\theta}\sin(\psi)\cos(\theta)\cos(\phi)-\dot{\phi}\sin(\psi)\sin(\theta)\sin(\phi)+\dot{\psi}\sin(\psi)\sin(\phi)+\dot{\phi}\cos(\phi)\cos(\psi)]z$$

$$\underset{z}{V}(x,y,z)=\dot{\theta}\cos(\theta)x+[\dot{\theta}\sin(\theta)\sin(\phi)+\dot{\phi}\cos(\theta)\cos(\phi)]y+[-\dot{\theta}\sin(\theta)\cos(\phi)-\dot{\phi}\cos(\theta)\sin(\phi)]z$$

Ces expressions sont très longues mais on peut les écrire sous la forme :

$$\underset{x}{v}(x,y,z)=\underset{xx}{v}x+\underset{xy}{v}y+\underset{xz}{v}z=(\underset{xx}{v}\quad\underset{xy}{v}\quad\underset{xz}{v})\begin{pmatrix}x\\y\\z\end{pmatrix} \qquad \text{(Eq. 3.3)}$$

$$\underset{y}{v}(x,y,z)=\underset{yx}{v}x+\underset{yy}{v}y+\underset{yz}{v}z=(\underset{yx}{v}\quad\underset{yy}{v}\quad\underset{yz}{v})\begin{pmatrix}x\\y\\z\end{pmatrix} \qquad \text{(Eq. 3.4)}$$

$$\underset{z}{v}(x,y,z)=\underset{zx}{v}x+\underset{zy}{v}y+\underset{zz}{v}z=(\underset{zx}{v}\quad\underset{zy}{v}\quad\underset{zz}{v})\begin{pmatrix}x\\y\\z\end{pmatrix} \qquad \text{(Eq. 3.5)}$$

Après, le module de ces vitesses est le carré :

$$v^2(x,y,z)=v_x^2+v_y^2+v_z^2 \qquad \text{(Eq. 3.6)}$$

$$v^2(x,y,z)=(v_{xx}\quad v_{xy}\quad v_{xz})A\begin{pmatrix}v_{xx}\\v_{xy}\\v_{xz}\end{pmatrix}+(v_{yx}\quad v_{yy}\quad v_{yz})A\begin{pmatrix}v_{yx}\\v_{yy}\\v_{yz}\end{pmatrix}+(v_{zx}\quad v_{zy}\quad v_{zz})A\begin{pmatrix}v_{zx}\\v_{zy}\\v_{zz}\end{pmatrix} \qquad \text{(Eq. 3.7)}$$

Où la matrice A est la suivante:

$$A=\begin{pmatrix}x^2 & xy & xz\\ xy & y^2 & yz\\ xz & yz & z^2\end{pmatrix} \qquad \text{(Eq. 3.8)}$$

Quand on a calculé les vitesses il est possible connaitre l'énergie cinétique car l'expression est la même que la utilisé dans la partie 2.2 (modélisation du gyropode). L'auteur calcule cette énergie selon le calcul ci-dessous :

$$T=\frac{1}{2}\int(y^2+z^2)dm[\dot{\psi}^2\sin^2(\theta)-2\dot{\phi}\dot{\psi}sin(\theta)+\dot{\phi}^2]$$

$$+\frac{1}{2}\int(x^2+y^2)dm[\dot{\psi}^2\cos^2(\phi)\cos^2(\theta)-2\dot{\theta}\dot{\psi}sin(\phi)\cos(\phi)\cos(\theta)+\sin^2(\phi)\dot{\theta}^2]$$

$$+\frac{1}{2}\int(x^2+z^2)dm[\dot{\psi}^2\sin^2(\phi)\cos^2(\theta)+2\dot{\theta}\dot{\psi}sin(\phi)\cos(\phi)\cos(\theta)+\cos^2(\phi)\dot{\theta}^2]$$

$$+\int xydm[\dot{\psi}^2\sin(\phi)\sin(\theta)\cos(\theta)+\dot{\psi}(\cos(\phi)\sin(\theta)\dot{\theta}-\sin(\phi)\cos(\theta)\dot{\phi})-\cos(\phi)\dot{\phi}\dot{\theta}]$$

$$+\int xzdm[\dot{\psi}^2\cos(\phi)\sin(\theta)\cos(\theta)+\dot{\psi}(-\cos(\phi)\cos(\theta)\dot{\phi}-\sin(\phi)\sin(\theta)\dot{\theta})+\sin(\phi)\dot{\phi}\dot{\theta}]$$

$$+\int yzdm[-\dot{\psi}^2\sin(\phi)\cos(\theta)\cos^2(\theta)+\dot{\psi}(\sin^2(\phi)\cos(\theta)\dot{\theta}-\cos^2(\phi)\cos(\theta)\dot{\theta})+\sin(\phi)\cos(\phi)\theta^2] \quad \text{(Eq. 3.9)}$$

Grâce à la supposition que le quadcopter est parfaitement symétrique, les produits d'inertie sont nuls et la matrice d'inertie du robot est diagonale. Ça permet simplifier l'équation de l'énergie cinétique laquelle devient :

$$T=\frac{1}{2}I_x(\dot{\phi}-\dot{\psi}\sin(\theta))^2+\frac{1}{2}I_y(\dot{\theta}\cos(\phi)+\dot{\psi}\sin(\phi)\cos(\theta))^2+\frac{1}{2}I_z(\dot{\theta}\sin(\phi)-\dot{\psi}\cos(\phi)\cos(\theta))^2 \quad \text{(Eq. 3.10)}$$

Les expressions qui définissent les inerties écrites sont:

$$I_x=\frac{1}{2}\int(y^2+z^2)dm \qquad I_y=\frac{1}{2}\int(x^2+z^2)dm \qquad I_z=\frac{1}{2}\int(x^2+y^2)dm$$

Ensuite, on doit calculer l'énergie potentielle :

$$V=g\int(-\sin\theta.x+\sin\phi\cos\theta.y+\cos\phi\cos\theta.z)dm \quad \text{(Eq. 3.11)}$$

$$V=g\int xdm.(-g\sin(\theta))+\int ydm.(g\sin(\phi)\cos(\theta))+\int zdm.(g\cos(\phi)\cos(\theta)) \quad \text{(Eq. 3.12)}$$

Avec tous ces facteurs connus c'est possible compléter l'équation de Lagrange (Eq. 2.6) en fonction du Lagrangien (Eq. 2.5). Postérieurement, il faut écrire et dériver l'équation de Lagrange selon chaque coordonnée généralisée. Cela signifie qu'on aura 3 équations car il y a 3 coordonnées.

Après avoir calculé les énergies cinétiques et potentielles du quadcopter il faut connaitre les forces non conservatives puisque c'est la partie laquelle va dans l'autre côté de l'égal dans l'équation de Lagrange. Ces forces sont dues à quatre effets :

1) La portance des moteurs crée en direction des axes X et Y des couples
2) La traînée des hélices crée un couple vertical
3) Les effets gyroscopiques sont dus aux hélices lors d'une rotation autour de l'axe X et Y
4) L'effet d'inertie du système est lié aux interactions des mouvements

Aussi, si on fait la somme des accélérations de la poussée, traînée, de l'effet d'inertie, de l'effet gyroscopique et en appliquant l'approximation de petits angles pour linéariser le système, on arrive aux trois équations suivantes :

$$\ddot{\psi} = \frac{d \cdot (\Omega_1^2 + \Omega_3^2 - \Omega_2^2 + \Omega_4^2)}{I_z} + \frac{I_x - I_y}{I_z} \cdot \dot{\theta} \cdot \dot{\phi} \qquad \text{(Eq. 3.13)}$$

$$\ddot{\phi} = \frac{I_{rot} \cdot \dot{\theta} \cdot (\Omega_1 + \Omega_3 - \Omega_2 - \Omega_4)}{I_x} + \frac{I_y - I_z}{I_x} \cdot \dot{\theta} \cdot \dot{\psi} + \frac{b \cdot l \cdot (\Omega_4^2 - \Omega_2^2)}{Ix} \qquad \text{(Eq. 3.14)}$$

$$\ddot{\theta} = \frac{-I_{rot} \cdot \dot{\phi} \cdot (\Omega_1 + \Omega_3 - \Omega_2 - \Omega_4)}{I_y} + \frac{I_z - I_x}{I_y} \cdot \dot{\phi} \cdot \dot{\psi} + \frac{b \cdot l \cdot (\Omega_3^2 - \Omega_1^2)}{Iy} \qquad \text{(Eq. 3.15)}$$

Les paramètres qui forment ces équations sont les suivants :

- d : C'est le coefficient de traînée et c'est possible le connaitre à partir des expériences
- b : Ce coefficient peut être calculé aussi expérimentalement
- I_x, I_y et I_z : Inerties du quadcopter selon chaque axe
- I_{rot} : Inertie du rotor d'un moteur
- Ω_i : Vitesse angulaire du moteur

Le calcul de ces paramètres ainsi que les expériences réalisées se trouve dans la section 3.3 SIMULATION.

De plus, il faut savoir la réponse des moteurs devant un changement de la commande. La fonction de transfert d'un moteur de courant continu est de deuxième ordre et dépend de certains paramètres du moteur comme la résistance interne, l'inductance, etc. mais si on néglige quelques paramètres, comme par exemple l'inductance car c'est négligeable devant la résistance du moteur, la fonction devient de premier ordre. L'auteur de ce rapport considère aussi que les frottements des moteurs sont négligeables devant l'inertie des rotors.

Après avoir appliqué ces simplifications la fonction de transfert d'un moteur est la suivante :

$$H(S) = \frac{k}{1 + \tau \cdot S} \qquad \text{(Eq. 3.16)}$$

Où k est le gain du moteur en rad/s/V et τ est la constante de temps du moteur exprimé en seconds. Il faudra faire des autres expériences pour pouvoir connaitre ces deux paramètres.

Le calcul et l'explication de tous les paramètres utilisés dans ces modèles (du quadctopter et du moteur) se trouve dans le paragraphe 3.3.2 Expériences et détermination des paramètres.

3.3. SIMULATION

La simulation du quadcopter sera réalisée aussi sous Simulink pour pouvoir avoir les résultats des deux robots dans un même environnement. Dans cette partie il y aura une explication des dessins du robot sous Matlab, l'explication et la procédure des expériences pour connaitre des paramètres et aussi tout ce qui concerne la simulation et les schémas sous Simulink.

3.3.1. Dessin du quadcopter

De la même manière que le gyropode, un code sous Matlab a été créé pour pouvoir observer le mouvement du robot pendant la simulation parce que sinon on pourrait seulement voir le résultat dans graphiques de l'oscilloscope. Le script qui dessine le quadcopter utilise aussi la fonction du cercle mais il y a plus de lignes car il y a plusieurs parties à dessiner. Une autre raison pour laquelle il y a plusieurs lignes est

que le quadcopter peut se bouger dans les trois dimensions. Cela signifie qu'on doit représenter différents points de vue en 2D pour pouvoir bien voir les mouvements dans chaque plan. Ci-dessous se montrent quelques lignes du code pour le mieux comprendre :

Tout d'abord il faut créer la fenêtre avec 4 figures différentes. La ligne que permet créer ça est la suivante :

```
subplot(2,2,1)
```

Ça veut dire qu'on va créer une fenêtre avec 4 *subplots* (2x2) et qu'on va dessiner dans le premier. Pour dessiner les parties du quadcopter il faut définir deux matrices avec les points X et Y. La différence c'est que dans ce cas, les points n'ont pas été calculés de manière trigonométrique sinon qu'on a multiplié chaque coordonné du point par la matrice de rotation suivante :

$$\begin{pmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{pmatrix}$$

Alors, l'expression qu'indique la position X et Y du point est la suivante :

$$\begin{pmatrix} x_1' \\ x_2' \end{pmatrix} = \begin{pmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{pmatrix} \cdot \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$$

Où x_i' et y_i' sont les points écrits en fonction de l'angle de rotation. Après connaitre ces expressions il est possible calculer la nouvelle matrice. Ci-dessous se montre une partie du code laquelle dessine les bras du quadcopter selon l'angle de rotation *theta*.

```
pxl1=[15*cos(theta)+0.5*sin(theta) 15*cos(theta)-0.5*sin(theta) -
15*cos(theta)-0.5*sin(theta) -15*cos(theta)+0.5*sin(theta)];

pyl1=[15*sin(theta)-0.5*cos(theta) 15*sin(theta)+0.5*cos(theta) -
15*sin(theta)+0.5*cos(theta) -15*sin(theta)-0.5*cos(theta)];

fill(pxl1,pyl1,[0.2 0.2 0.2],'LineWidth',1);
hold on
```

La figure suivante montre le dessin de ces lignes du code selon un angle "*theta*" égale à 15º aussi :

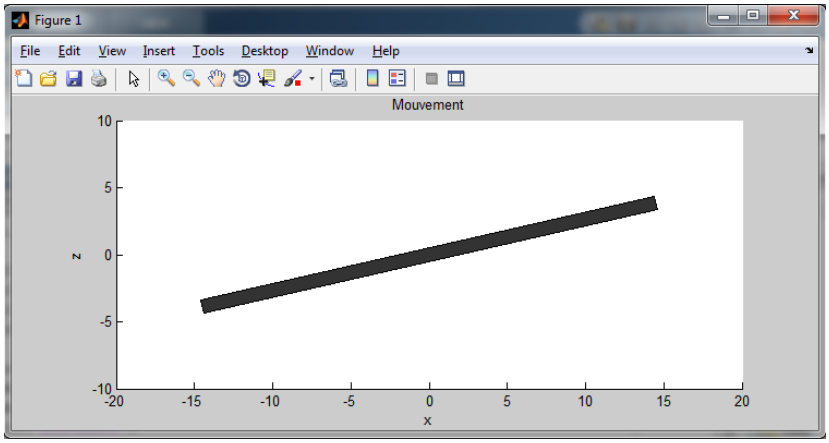


Figure 27: Dessin du bras du quadcopter

Avec les matrices des coordonnées complétées, il est possible de dessiner et de remplir la forme décrite selon ces matrices avec la même comande *fill*. La ligne *hold on* est aussi nécessaire pour maintenir le dessin et pouvoir regarder le mouvement du robot.

Au début, un script Matlab a été créé pour faire bouger le robot et vérifier le fonctionnement. Cette fonction, de la même façon que le cas du gyropode, change les valeurs des angles et de la position grâce aux boucles *for* et le résultat est le suivant :

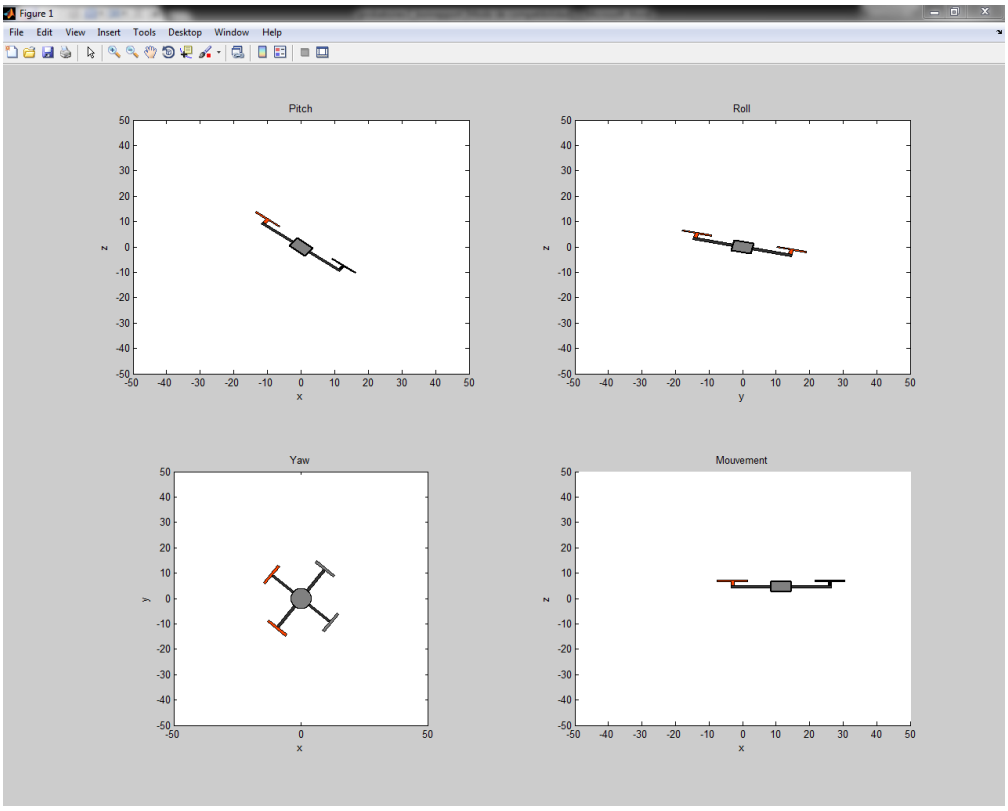


Figure 28: Représentation du quadcopter selon les angles et la position

Il faut remarquer que les hélices oranges sont l'avant du robot et les noires sont l'arrière.

L'étape suivante était réussir à lier ce dessin avec Simulink. Au début on a fait un schéma simple pour vérifier que ça marchait et celui est le suivant :

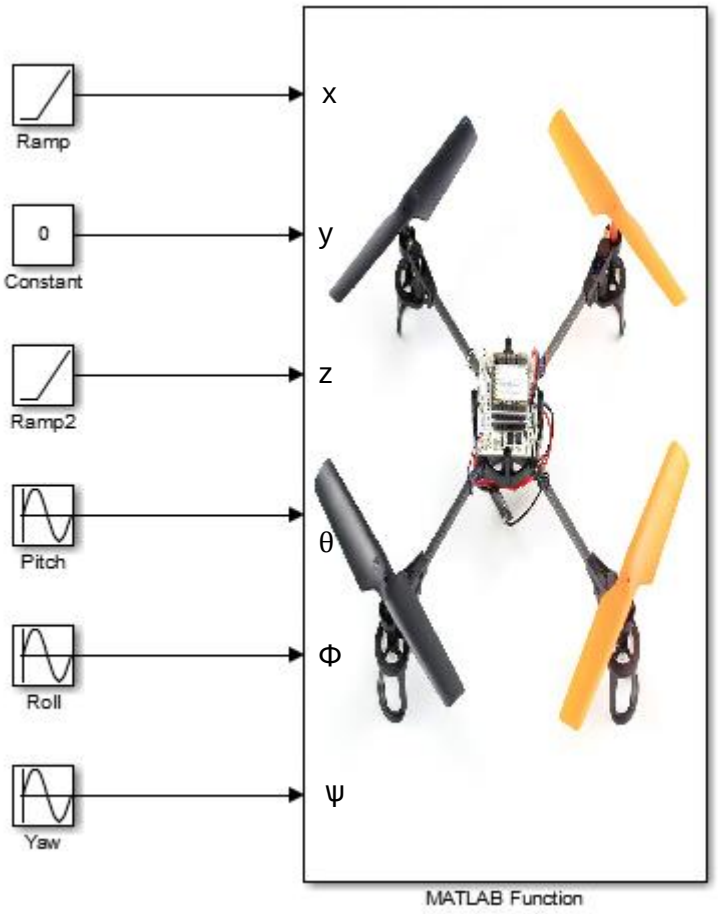


Figure 29: Schéma pour le dessin du quadcopter sous Simulink

3.3.2. Expériences et détermination des paramètres

Pour pouvoir avoir le modèle le plus similaire au le quadcopter utilisé dans ce projet, il faut connaitre quelques paramètres qui sont caractéristiques de ce robot. Après, avec ces valeurs, serait possible les introduire dans le correspondant bloc sous Simulink et commencer la simulation. Ces paramètres sont liés aux inerties du robot et des moteurs, d'autres aux moteurs et d'autres à la force du robot ou vitesse de rotation, par exemple. A continuation, la procédure et les résultats obtenus de ces expériences seront expliqués.

Calcul des inerties

Le calcul des inerties du robot selon chaque axe est nécessaire parce que ces paramètres sont utilisés pour connaitre certains effets du quadcopter. En dehors de ces résistances à la rotation du robot sur chaque axe, il faut aussi calculer l'inertie du rotor du moteur.

Dû à qu'on ne dispose pas d'une machine pour connaitre ces inerties, une approximation sera réalisée pour avoir une valeur indicatif. Cette approche consiste à mesurer la masse de quelques parties du quadcopter, supposer que ces masses sont des cylindres et calculer ses inerties en appliquant le théorème de Huygens-Steiner. Après avoir démonté partiellement le robot, la lecture des masses a été la suivante :

- Masse du moteur, hélice et bras : 15g chaque un
- Masse du corps : 40g

Toutes les parties moteur-hélice-bras pèsent le même : 15g. Pour cette raison on peut dire que le poids total du quadcopter c'est 100g approximativement. Le dessin à partir duquel les inerties seront calculées est le suivant :

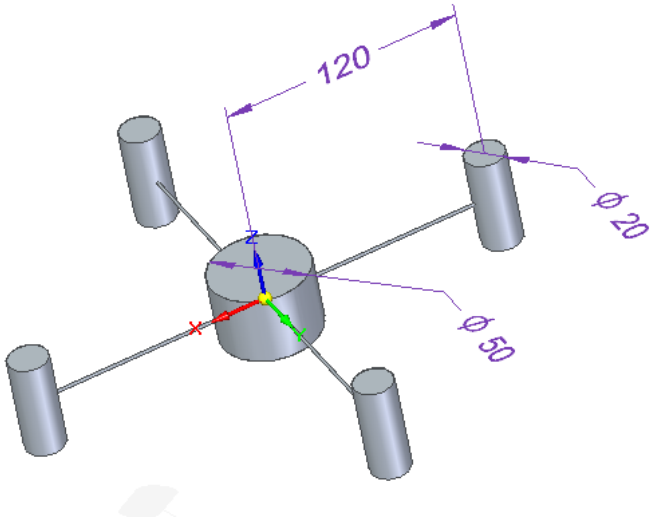


Figure 30: Croquis de la distribution de masses du robot (cotes en millimètres)

Il faut ajouter que la masse des bras qui unissent le cylindre du corps avec les autres 4 est considérée dans ces derniers.

Les expressions que seront utilisés dans le calcul des inerties sont les suivantes :

$$I_{cyl} = \frac{1}{2} \cdot M \cdot R^2 \qquad \text{(Eq. 3.4)}$$

Théorème de Huygens-Steiner :

$$I_{//} = I + M \cdot r^2 \qquad \text{(Eq. 3.5)}$$

Où $I_{//}$ c'est le moment d'inertie du corps selon un axe parallèle, *I* c'est le moment d'inertie du corps, *M* la masse de l'élément et r^2 la distance parallèle entre ces 2 axes.

Pour connaitre les moments d'inertie selon les axes *x* et *y* il faut appliquer aussi le théorème des axes perpendiculaires qui est écrit ainsi :

$$I_z = I_x + I_y \qquad \text{(Eq. 3.6)}$$

Puisque le quadcopter est considéré symétrique, les inerties I_x et I_y prennent la même valeur et ça simplifie un peu le calcul.

Avec ces expressions c'est possible connaitre les inerties selon chaque axe. Il faut appliquer le changement d'axe (parallèle et perpendiculaire) puisque les cylindres des moteurs ne sont pas au centre du robot. Ci-dessous se montrent les calculs réalises et les valeurs finals :

$$I_z = 4 \cdot \left(\frac{1}{2} \cdot M_1 \cdot R_1^2 + M_1 \cdot r^2\right) + \frac{1}{2} \cdot M_2 \cdot R_2^2$$

$$I_z = 8{,}093 \cdot 10^{-4}\ kg \cdot m^2$$

$$I_x = I_y = 2 \cdot \left(\frac{1}{2} \cdot M_1 \cdot R_1^2 \cdot \frac{1}{2}\right) + 2 \cdot \left(\frac{1}{2} \cdot m1 \cdot R1^2 \cdot \frac{1}{2} + M_1 \cdot r^2\right) + \frac{1}{2} \cdot M_2 \cdot R_2^2 \cdot \frac{1}{2}$$

$$I_x = I_y = 7{,}157 \cdot 10^{-6}\ kg \cdot m^2$$

- M_1 et M_2 sont les masses des moteurs et du corps respectivement (15g et 40g)
- R_1 et R_2 sont les rayons supposés des cylindres des moteurs et du corps respectivement (10mm et 25mm)
- r est la distance entre le centre du cylindre des moteurs au centre du corps (120mm)

Il faut introduire toutes les valeurs en unités du système international (kg et m) pour avoir l'inertie en kg·m^2.

Pour le calcul de l'inertie de l'hélice et le moteur on va utiliser l'expression suivante laquelle unit l'inertie de l'hélice et l'axe du moteur avec la du rotor avec la relation de transmission.

$$I_{Hel+mot} = I_{Hel+axe} + {i_{red}}^2 \cdot I_{rot} \qquad \text{(Eq. 3.7)}$$

La relation *i* peut être calculée avec les nombres de dents des deux engrenages :

$$Z_1 = 11 \quad et \quad Z_2 = 66$$

$$i_{red} = \frac{1}{6} = 0.167$$

Puisque la relation de transmission est très petite, il faut seulement calculer l'inertie de l'hélice et l'axe car le carré de cette est encore plus petit et on peut considérer égal à zéro. Cette inertie est calculée à partir du dessin suivant :

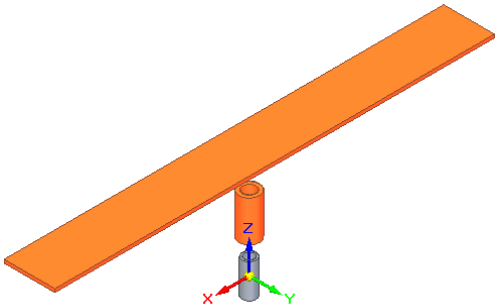


Figure 31: Dessin des parties du moteur-hélice qui ont inertie

Pour calculer l'inertie totale il faut calculer la de chaque partie. Ces objets sont simples, deux tubes et une plaque. Ses inerties sont respectivement les suivantes :

$$I_{tube} = \frac{1}{2} \cdot M \cdot (R_1^2 + R_2^2) \qquad \text{(Eq. 3.8)}$$

$$I_{plaque} = \frac{1}{12} \cdot M \cdot (a^2 + b^2) \qquad \text{(Eq. 3.9)}$$

Les dimensions de la plaque sont 130x15x1mm, les du cylindre de l'hélice Ø6xØ4x12mm et les du cylindre de l'axe sont Ø4xØ3x10mm. Avec les dimensions et la densité du polypropylène (946 kg/m^3) et du acier inoxydable 304 (8000 kg/m^3) est possible connaitre la masse *M* et calculer les inerties. La somme de l'inertie de chaque partie c'est l'inertie totale et les transformer n'est pas nécessaire parce que dans ce cas ces objets tournent au tour de son centre. L'inertie totale est donc :

$$I_{Hel+mot} = 2{,}636 \cdot 10^{-6}\ kg \cdot m^2$$

Calcul du paramètre "*b*"

Le paramètre "b" s'utilise dans les équations du modèle mathématique et c'est la constante qui unisse la poussée et la vitesse de rotation d'un moteur. Selon la source [2], la manière pour le calculer est en faisant des expériences. Ci-dessous se trouve l'expression pour le calculer :

$$F_i = b \cdot \Omega^2 \qquad \text{(Eq. 3.10)}$$

F_i est la force par l'ensemble hélice-moteur exprimée en N et Ω est la vitesse angulaire de l'hélice. D'un côté, pour savoir la valeur de la vitesse, on a mis quelques autocollants blanches dans les hélices pour que le tachymètre la puisse lire. Ci-dessous une image d'une hélice avec une partie de l'autocollant.

Figure 32: Hélice avec l'autocollant blanche

De l'autre, pour avoir la valeur de la force on utilisera une balance. On mettra le quadcopter sur la balance, avec un des bras fixé à la surface (avec seulement ce moteur connecté), on donnera de puissance et on va lire le poids négatif indiqué dans la balance. Ensuite, si on multiplie ce poids par la pesanteur on aura la valeur de la force en Newton. A chaque instant qu'on lit la valeur du poids il faut lire aussi la vitesse angulaire de l'hélice. Avec plusieurs valeurs on pourra faire une représentation graphique et avoir une expression d'une variable en fonction de l'autre.

Dans une première expérimentation le poids indiqué par la balance était 7g. Ça n'est pas possible car le poids total du quadcopter est de 100g et chaque bras doit générer une force de 25g, au moins, pour pouvoir soutenir le robot dans l'air. Après avoir essayé quelques autres fois, le moteur n'avait pas suffisante puissance dû à la batterie

faible et on devait la charger. Ensuite, quand on l'a bien chargé (avec un chargeur spécial qui n'était pas inclus dans le kit), le moteur a fonctionné parfaitement et avec beaucoup de puissance et de manière constante. L'autre problème était que la même force de l'air puisse sur la balance comme un poids positif, contraire à la force qu'on veut mesurer. Alors, pour pouvoir prendre une lecture correcte du poids négatif, on a utilisé un carton pour faire que l'aire ne touche pas la surface de la balance. Ci-dessous une image du montage expérimental :

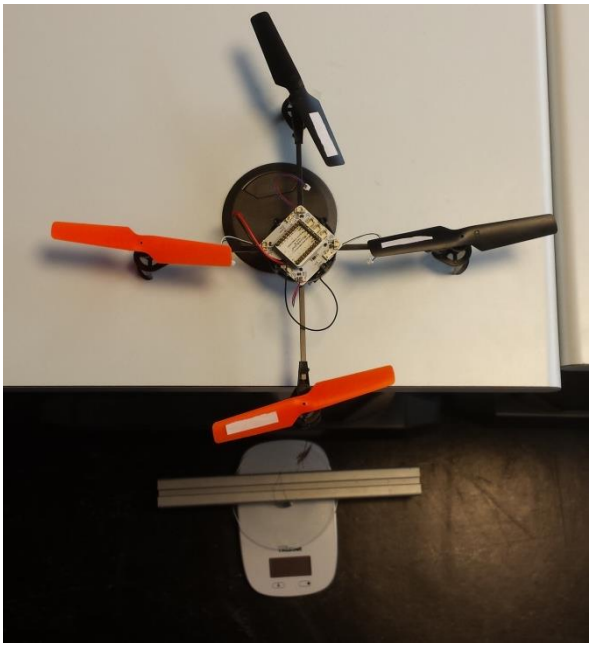


Figure 33: Montage expérimental pour déterminer le paramètre "d"

Calcul du paramètre "*d*"

Ce paramètre est aussi nécessaire pour l'introduire dans le modèle sous Simulink et arriver à avoir une simulation. Dans ce cas, *d* est une constante reliant la traînée et la vitesse d'un moteur. Pour connaitre la valeur de cette constant il faut aussi faire une expérience laquelle consistent à faire tourner le drone un quart de tour en mesurant la vitesse angulaire d'une hélice et le temps. Malgré on a essayé de soutenir le quadcopter avec un fil fixé au plafond pour supprimer le frottement, l'expérience n'était pas possible car le robot bougeait dans toutes directions. Il faut avoir un support avec un roulement pour fixer le quadcopter et le permettre tourner seulement autour de l'axe vertical (lacet).

Selon la source [2], pour calculer cette variable on doit partir de l'expression suivante laquelle relie l'angle de lacet et la vitesse des moteurs. Il faut ajouter que pour cette

expérience on va utiliser seulement 2 moteurs, le 1 et 2 ou le 3 et 4, car ils sont le couple de moteurs qui font tourner le robot dans le sens des aiguilles d'un montre ou dans l'inverse.

$$\ddot{\psi} = \frac{d(\Omega_1^2 + \Omega_3^2 - \Omega_2^2 - \Omega_4^2)}{Iz} \qquad \text{(Eq. 3.11)}$$

Grâce à que le quadcopter seulement peux autour de l'axe Z, il est possible d'écrire la suivante rélation :

$$\frac{(Ix - Iy)}{Iz}\dot{\theta}\dot{\Phi} = 0 \qquad \text{(Eq. 3.12)}$$

Ensuite, il faut intégrer cette fonction à deux reprises pour avoir la loi de l'angle de lacet :

$$\psi = \frac{d(\Omega_1^2 + \Omega_3^2 - \Omega_2^2 - \Omega_4^2)}{Iz}\frac{t^2}{2} \qquad \text{(Eq. 3.13)}$$

En considérant que la vitesse du couple de moteurs est la même et que l'angle est 90º exprimé en radians, finalement on a la suivante expression :

$$d = \frac{\pi \cdot I_z}{2 \cdot \Omega^2 \cdot t^2} \qquad \text{(Eq. 3.11)}$$

I_z est la inertie déjà calculé du quadcopter autour de l'axe Z, Ω est la vitesse angulaire de l'hélice et *t* est le temps de faire ce quart de tour.

Pour éviter le frottement entre parties du quadcopter et la table, on a pendu le robot au plafond avec fil à coudre pour le soutenir en l'air.

Acquisition des paramètres "*k*" et "*τ*" du moteur

Ces deux valeurs sont liées aux caractéristiques internes du moteur de courant continu. Pour les connaitre on va accélérer un moteur et, avec l'aide d'un oscilloscope, on va avoir la courbe de sa réponse à partir de laquelle on pourra extraire ces constantes. Le problème est qu'on ne connait pas exactement la valeur de la commande quand on mouve le joystick du joypad et ça ne permet pas de connaitre le gain du moteur. Pour cette raison et parce que les expériences sont difficiles à réaliser,

on va supposer ces valeurs en considérant que la constante de temps doit être petite et on va prendre la valeur du gain en fonction des autres moteurs de CC du marché.

3.3.3. Schémas du modèle

Le but de cette partie est construire et expliquer les différents schémas qui permettent calculer des paramètres et suivre le modèle mathématique du robot (Eq. 3.1-Eq. 3.3). Puisque dans les trois équations il y a des dérivées des angles et dû à qu'on parte des accélérations angulaires il faudra intégrer ces signaux pour les utiliser dans autres opérations.

Tous les schémas ci-dessous ont été dessines en suivant le rapport [2]. Les schémas suivants ne sont pas exactement égaux aux de ce document car la organisation des blocs et connecteurs est diffèrent et parce qu'il y avait quelques erreurs. En supposant que le modèle a été bien calculé, les figures suivantes montrent toutes les connections et tous les blocs nécessaires pour construire les équations du modèle.

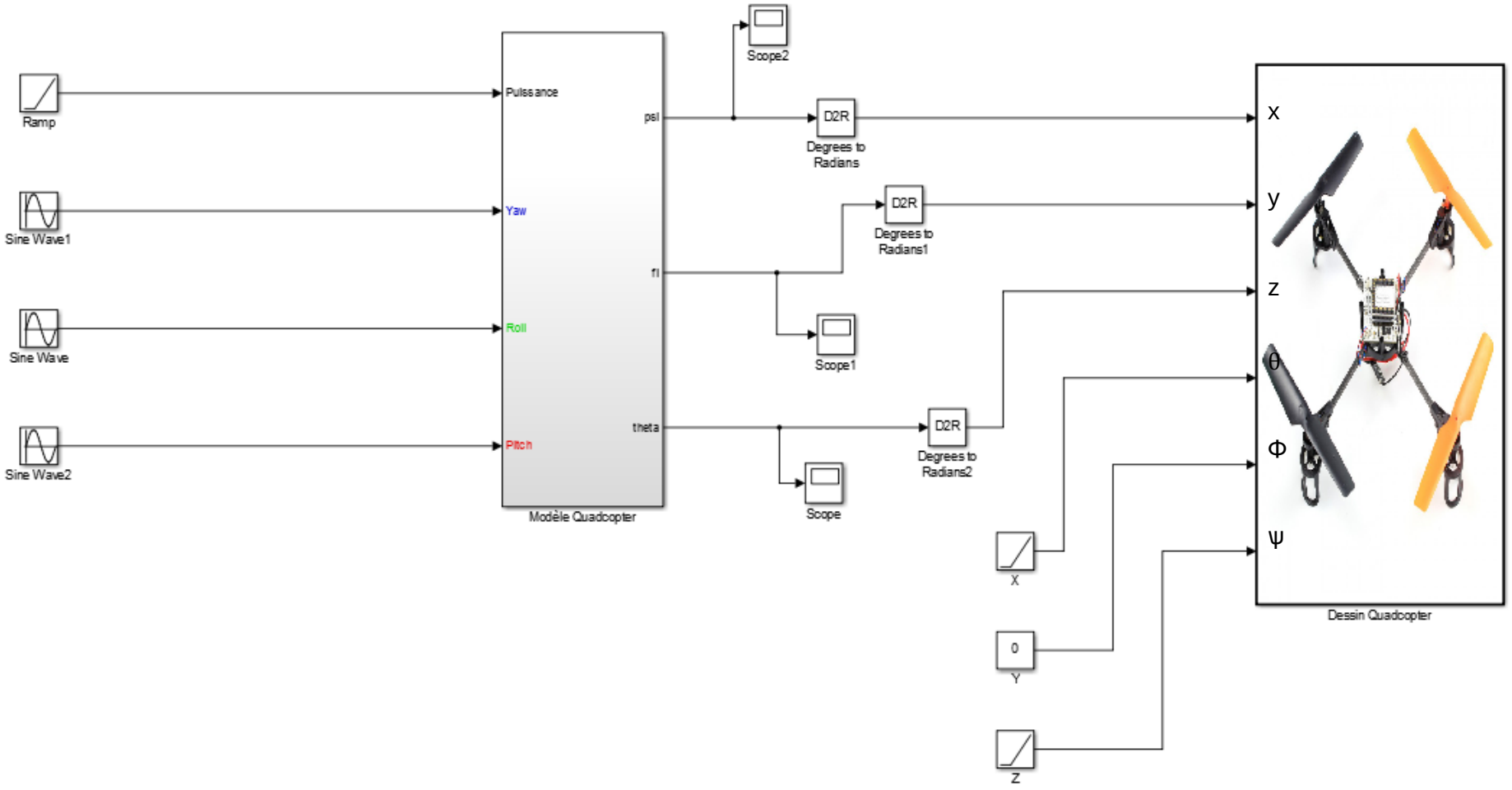


Figure 34: Schéma de contrôle et dessin des 3 angles

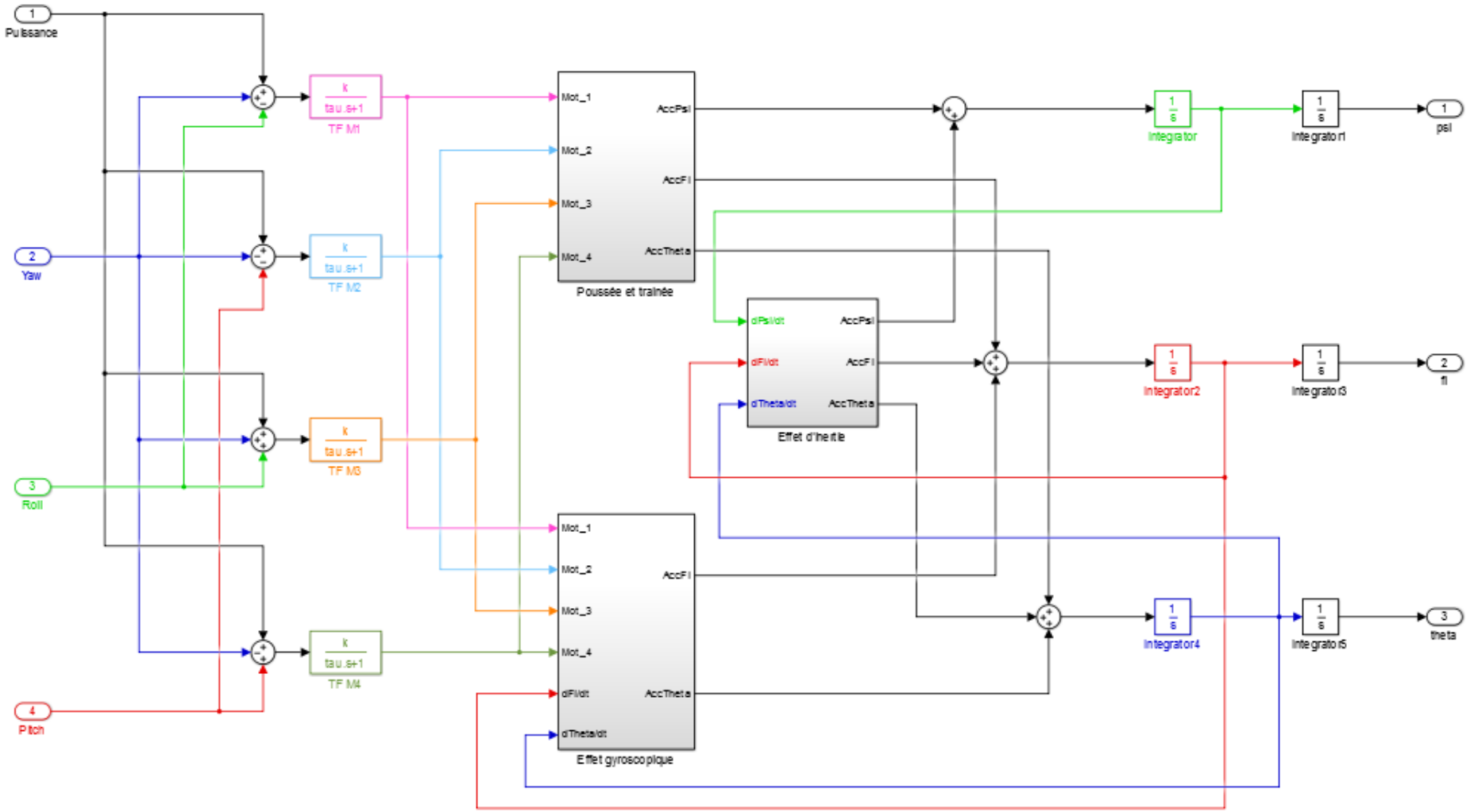


Figure 35: Schéma du modèle du quadcopter

Dans la Figure 34, les paramètres x, y et z ne sont pas liés au modèle mathématique puisque on n'a pas trouvé une relation entre ces variables et les angles du robot ou les commandes. Pour cette raison, le changement de ces paramètres est réalisé encore par fonctions sinus, rampes, constantes, etc. Il faut ajouter que la puissance est une rampe de pente 10.

Le schéma ci-dessous représente le calcul des termes de poussée et trainée contenus dans les 3 équations du modèle. Après avoir la valeur Mot_i dans le schéma du modèle, c'est possible connaitre la vitesse car on a supposé que cela au carré est proportionnel à la traînée et à la poussée.

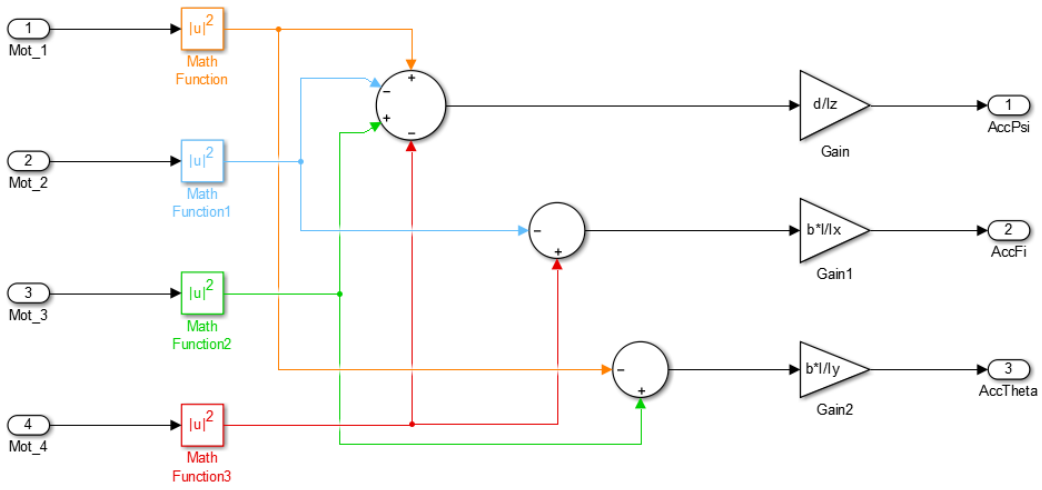


Figure 36: Schéma du calcul de la poussée et trainée

Un autre schéma nécessaire pour pouvoir compléter le modèle est le suivant. Avec ces connections c'est possible avoir les termes gyroscopiques qui figurant dans les 3 équations du modèle. Il y a seulement deux sorties parce que dans l'équation du deuxième dérivée de psi il n'y a pas aucun terme correspondant à l'effet gyroscopique.

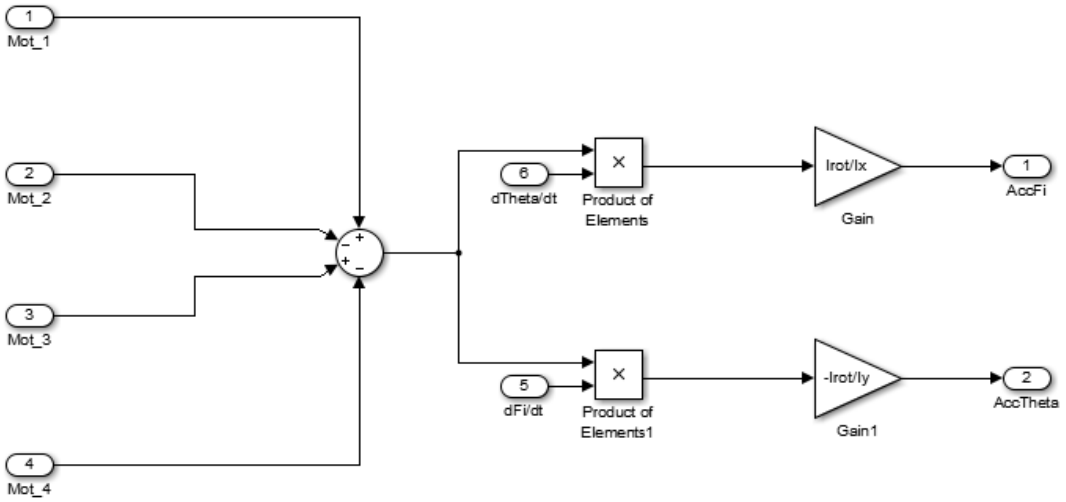


Figure 37: Schéma du calcul de l'effet gyroscopique

Par rapport à l'effet d'inertie, ci-dessous se trouvent les différents blocs connectés pour simuler cet effet. Il faut remarquer que ce schéma est assez différent de celui qui apparaît dans le rapport [2]. Ce schéma est bon car toutes les connections forment la partie de l'équation correctement.

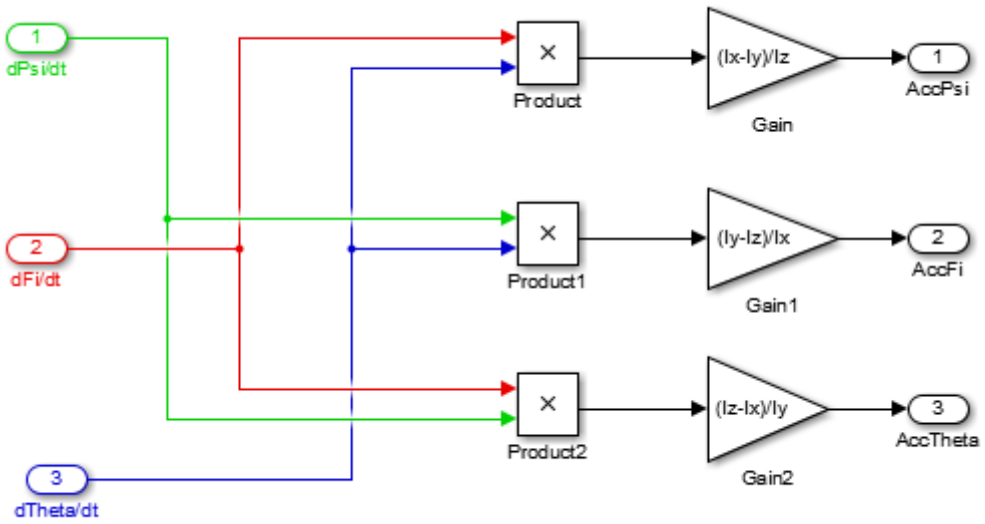


Figure 38: Calcul de l'effet d'inertie

Les valeurs des paramètres utilisés dans tous ces schémas sont introduites dans un script Matlab lequel doit être exécuté avant de la simulation pour garder les valeurs sur le "*workspace*" de Matlab. Ci-dessous les lignes de code et les valeurs de ces variables :

```
%Definition des paramètres du QUADCOPTER

Iz=8.093e-4;
Ix=7.157e-6;
Iy=Ix;
Irot=2.636e-6;
b=(35e-3*9.81)/(5240^2);
d=pi*Iz/(2*350^2*2^2);
k=5;
tau=0.05;
l=0.12;
```

Il convient de rappeler que les valeurs "*b*", "*d*", "*k*", et "*tau*" ont été supposés et ajustés selon les résultats des simulations car la réalisation des expériences n'était pas précise et dans quelques cas n'était pas possible de réaliser.

3.3.4. Comment lancer une simulation

Dans ce paragraphe on trouve les indications à suivre pour connaitre tous les fichiers nécessaires pour avoir une simulation du quadcopter. D'abord, il faut dire que tous ces fichiers doivent être dans le même dossier car sinon Matlab et Simulink présenteront des erreurs. Aussi, il faut avoir ce dossier ouvert sous Matlab.

Les fichiers qui dessinent le quadcopter sont le *cercle.m* et *dessinquad.m*. Pour la même raison que le gyropode, il faut y avoir la photo du quadcopter pour le masque laquelle s'appelle *quadcopter.jpg*.

Ensuite, il faut ouvrir le fichier *definitionquad.m* et le lancer pour avoir les valeurs des paramètres du robot et les utiliser après sous Simulink. Après tout ça, il est possible ouvrir le fichier *controlquad.slx* et lancer la simulation sans aucun erreur.

3.4. RÉSULTATS

Après avoir assemblé les deux kits et après avoir testé le bien fonctionnement du joypad c'était possible le communiquer avec le quadcopter. Avant de commencer à utiliser le robot c'est nécessaire suivre procédure indiqué dans le tutoriel pour garantir la sécurité de l'exécution. Les pas à suivre sont les suivants :

1. Joypad L switch vers le bas et R switch vers le haut
2. Placer le robot dans une surface horizontale et l'allumer
3. Attendre jusqu'à le LED bleu arrête clignoter
4. Tourner le L Joystick à droite (le maximum) et attendre deux seconds jusqu'à le LED bleu reste allumé (ça veut dire que le quadcopter est débloqué)

Après il faut seulement débloquer l'accélérateur (quand il est en position abaissée) et commencer à faire tourner les hélices lentement. Sans trop accélération, c'est aussi possible tester que le robot peut tourner. Grâce à cette première mise en œuvre, vérifier que toute marche correctement a été possible.

Finalement, après avoir le modèle sous Simulink, tous les paramètres estimés et tout le schéma bien connecté c'était possible avoir la simulation du quadcopter. Il faut remarquer que dans un premier instant la simulation était très lente dû au grand nombre de calculs que Matlab et Simulink doivent faire. Pour résoudre ce problème on a changé le "*sample time*" (click droit > *Block Parameters* (*subsystem*)) de la fonction qui dessine le quadcopter car cela c'est qui fait marcher le programme très lentement. Avec une entrée sinusoïdale de l'angle pitch les résultats étaient les suivants :

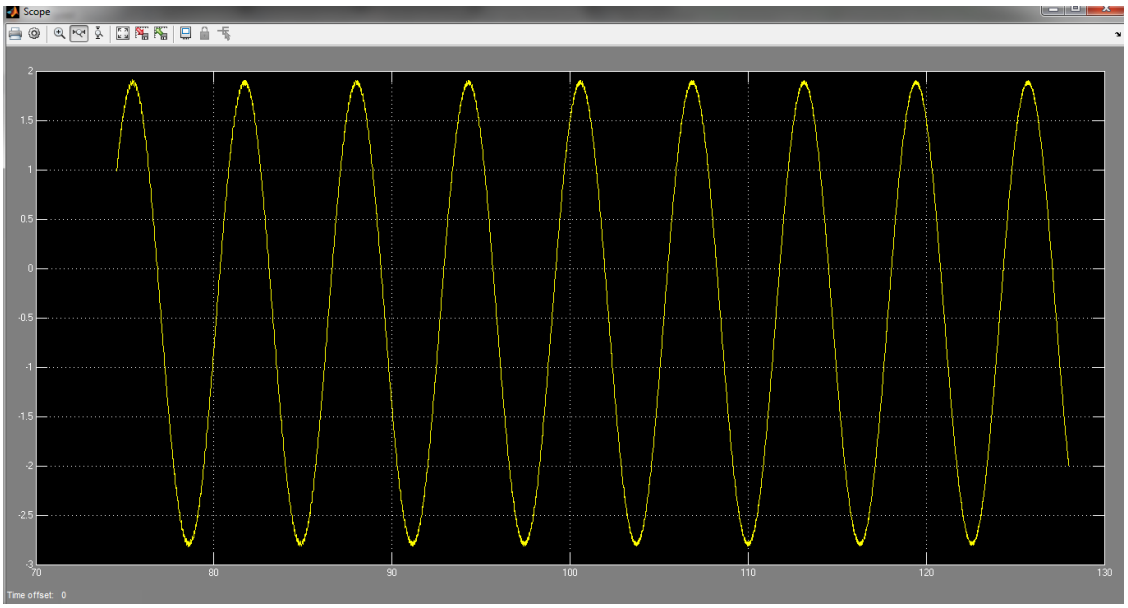


Figure 39: Angle pitch à la sortie oscillant selon l'entrée

L'angle pitch oscille puisque la commande est une fonction sinus d'amplitude 10. On peut regarder que cette fonction sinus n'est pas centrée au zéro car la valeur plus positive est autour de 2 et le plus négatif est plus grand que -2,5. Malgré cela, le quadcopter se bouge bien constamment et le mouvement de l'animation est fluide.

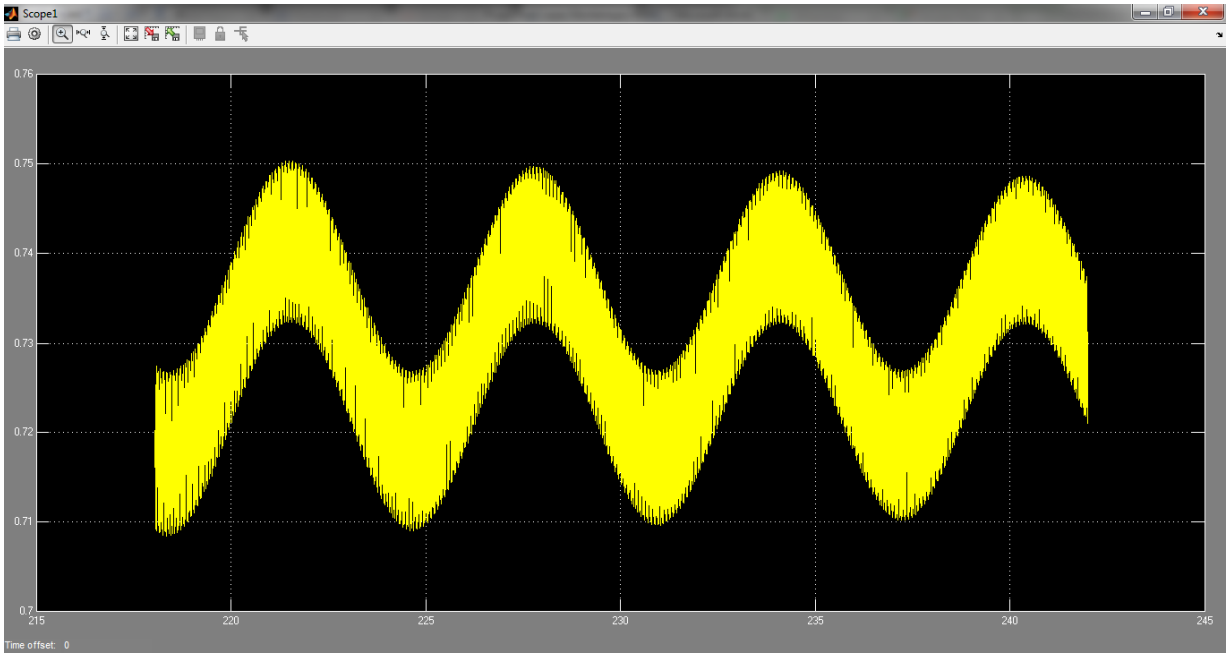


Figure 40: Angle roll très petit

Concernant l'angle du roulis, il faut dire que en dépit de que la commande de cet angle est zéro il y a quelque variation de celui. On peut apprécier sur la Figure 40 que l'angle est toujours petit et qu'il maintient l'amplitude.

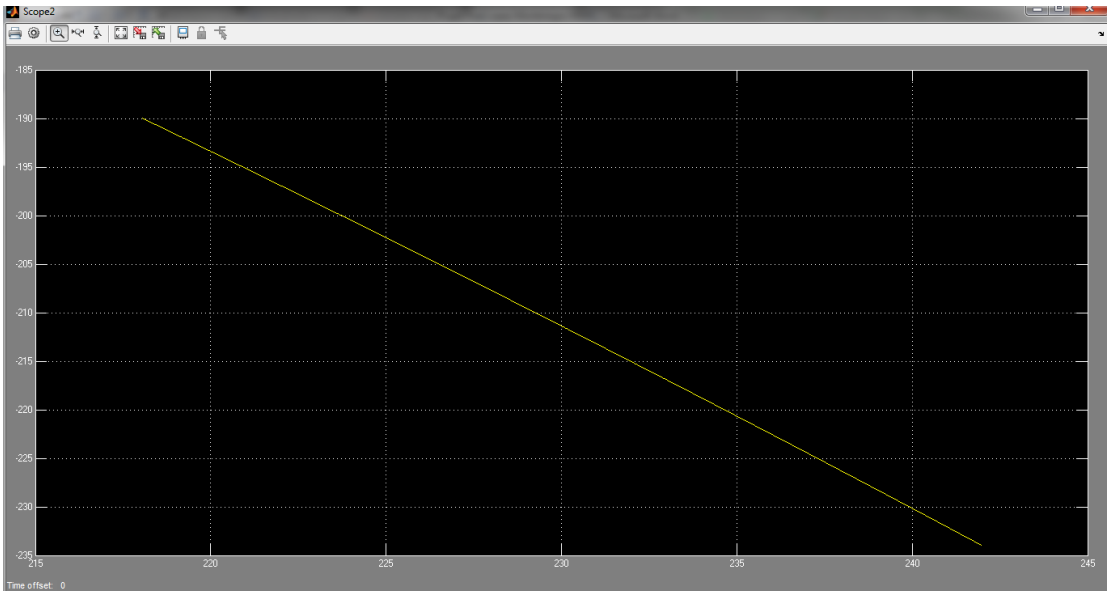


Figure 41: Angle yaw décroissant

Cette dernière graphique montre l'évolution de l'angle "*yaw*" en fonction du temps. La première chose à remarquer c'est que la valeur de celui décroît selon le temps de manière linéal. Ça n'est pas un comportement normal car la commande de l'angle yaw

reste à zéro aussi et cela est, peut-être, dû à que l'angle roll est toujours positif et la combinaison de ceux deux ("*pitch*" et "*roll*") fait changer la valeur de l'angle lacet.

Finalement, ci-dessous se trouve une photo des dessins du quadcopter prise pendant la simulation. L'angle "*pitch*" est vraiment petit (autour de $\pm 2º$) et c'est difficile d'apprécier alors que le "*roll*" reste toujours presque à zéro.

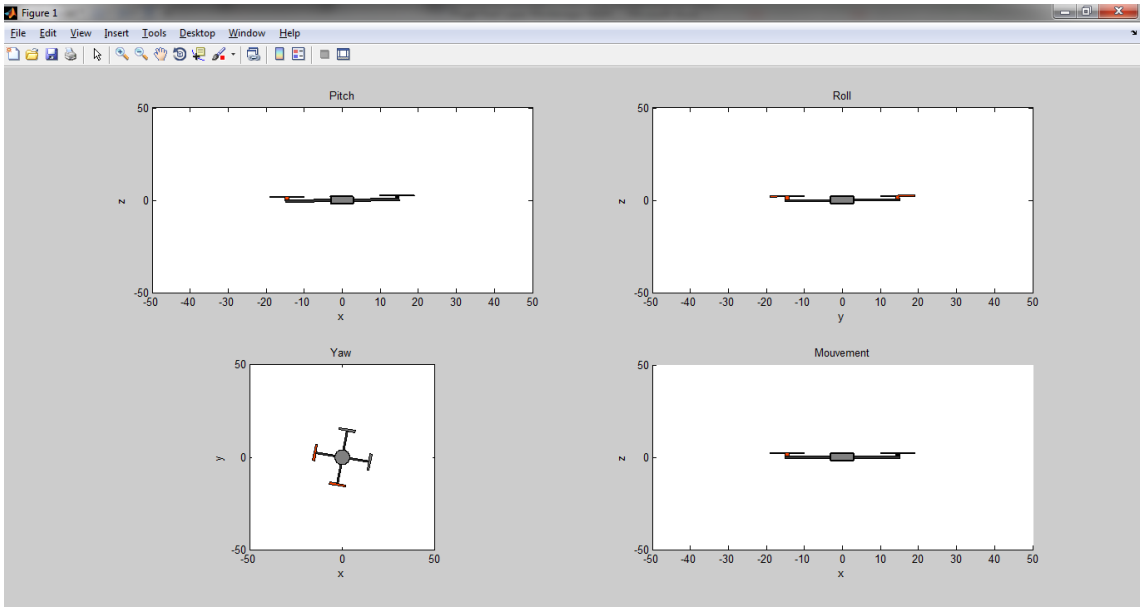


Figure 42: Mouvements du quadcopter selon la simulation

Il faut aussi ajouter que pour avoir un bon résultat et une bonne simulation l'amplitude de la commande Pitch ou Roll ne peut pas être très grand car on a linéarisé le modèle en supposant les angles petits. Si celui est trop grand, le mouvement du quadcopter devient instable.

Pour mieux apprécier le fonctionnement de cette simulation, ci-dessous se trouve un autre exemple avec la commande de l'angle "*yaw*" au lieu de "*pitch*" :

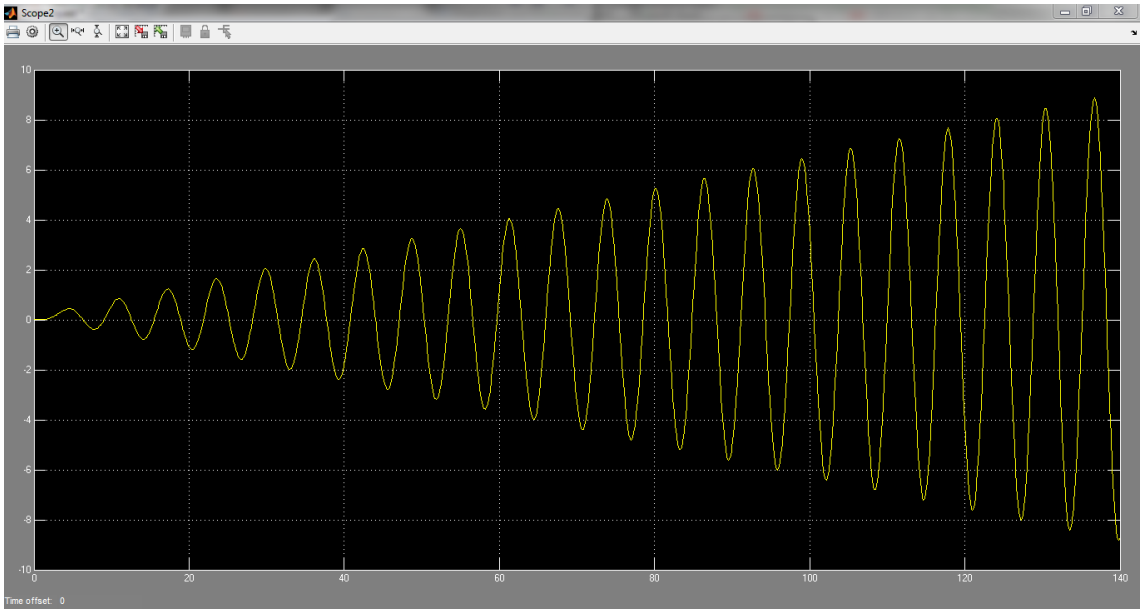


Figure 43: Angle "yaw" selon l'entrée sinusoïdale

On peut regarder que le comportement de cet angle tend à croître assez rapidement. La raison pour laquelle le quadcopter devient instable selon ces conditions d'entrée est que la puissance augmente sous forme de rampe. Ça fait qu'à chaque cycle le robot tourne plus fortement et que l'angle soit plus grand. Dans ce cas, les sorties des angles "*pitch*" et "*roll*" sont constantes et égales à zéro :

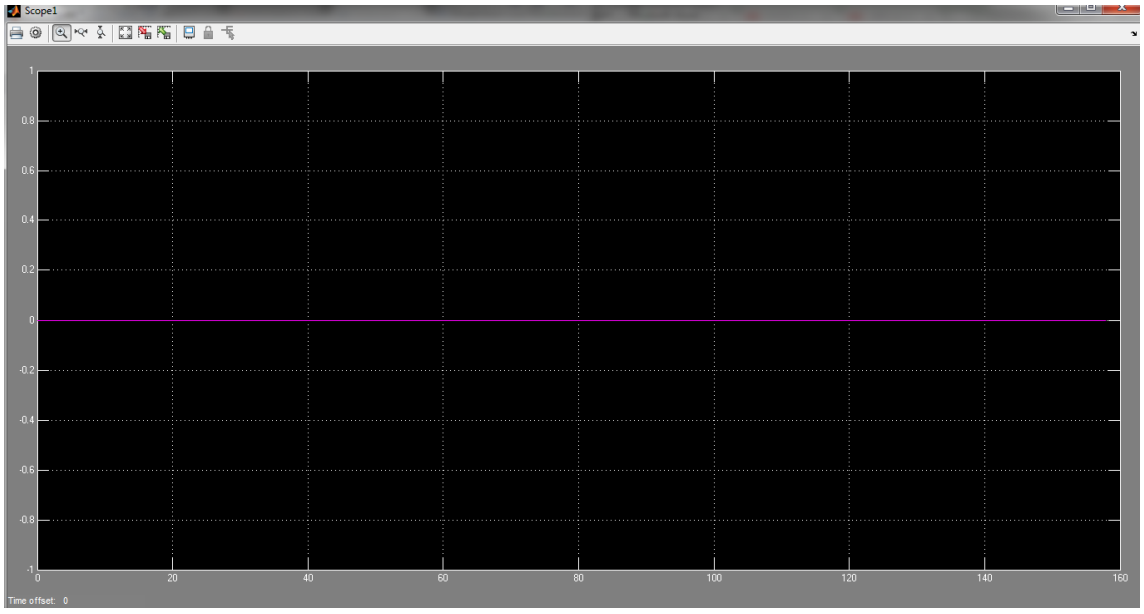


Figure 44: Angles "pitch" (jaune) et "roll" (violet) égaux à zéro

On ne peut pas regarder la ligne de l'angle "*pitch*" parce qu'elle a la même valeur que le "*roll*" et les deux graphiques sont superposées.

3.5. CONCLUSION

L'assemblage des parties et la construction, tant du quadcopter que du joypad, a été assez facile puisque il n'y avait pas aucun problème. Le seul obstacle, qu'il ha fait retarder le processus d'assemblage du joypad, était la réinitialisation du CoreRF car il a fait penser que peut être quelque component, comme le CoreRF ou l'écran, ne fonctionnaient pas. Par le moment le quadcopter et le joypad fonctionnent correctement et il n'y a pas des problèmes avec la connexion.

Pendant la réalisation des expériences la vitesse des hélices a diminué jusqu'à arriver à l'arrêt. Cela n'était un problème considérable puisque c'était seulement que la batterie du robot était déchargée et il a fallu la recharger. Après quelques heures de chargement le quadcopter a fonctionné parfaitement et avec toute la puissance disponible.

Ecrire le code sous Matlab pour arriver à dessiner le quadcopter vu depuis toutes les directions était encore plus difficile que le cas du gyropode parce que il y a plus de variables (angles et directions) et parce qu'il y a plus de parties (hélices, corps et structure) lesquelles changent sa position linéal et angulaire. Ainsi que dans le cas du gyropode, l'animation du quadcopter est plus fluide sous Matlab, probablement aussi dû à la manque de mémoire RAM.

Un autre aspect à remarquer c'est que le quadcopter du modèle mathématique suivi (source [2]) a des hélices sous forme de croix et pas sous X. Ça veut dire que la simulation est selon l'autre modèle proposé et il peut y avoir quelques différences avec le modèle réel du quadcopter utilisé dans ce projet.

Le processus d'obtention des paramètres était un peu dur, car on a essayé de faire des expériences sans succès, mais finalement on pouvait avoir des valeurs supposés qui ont permis avoir une bonne simulation et des bons résultats.

Finalement, et après avoir réalisé des simulations, on peut conclure que le quadcopter suive un mouvement assez réel dans l'animation et qu'il reste oscillant selon la commande d'entrée. Malheureusement, on ne pouvait pas simuler le mouvement linéal du robot car on ne connait pas la relation entre les commandes et la position x, y et z de celui.

4. RÉSUME DE L'ÉTUDE DE COÛTS

En considérant les correspondants dépenses de matériel et de main-d'œuvre, ainsi que les coûts indirects pour la conception des systèmes automatiques à bas coût, le budget total s'élève à 9537.77€ (NEUF MILLE CINQ CENTS TRENTE-SEPT EUROS ET SOIXANTE-DIX-SEPT CENTIMES).

5. CONCLUSION

Cette partie du projet présente les conclusions à lesquelles on a arrivé après avoir analysé et expérimenté avec les deux robots. On va comparer certains aspects entre les deux pour connaitre les différences entre eux et pour s'il y a quelque chose en commun, on va raconter les différents aspects liés à la utilisation d'outils à bas coût dans ce projet et on parlera aussi des spécifications et des possibles projets futurs pour suivre ce travail.

Les tableaux suivants montrent les principaux aspects à souligner pour chaque robot, qu'il s'agisse de positifs ou de négatifs. La première fait référence au gyropode :

Tableau 7: Caractéristiques générales résumées du gyropode

Description	Commentaires du gyropode
Processus d'assemblage	La procédure a été assez facile. L'unique problème était le court-circuit dans le connecteur de la batterie. Aussi, une partie en bois très fragile du gyropode s'est cassée mais il peut fonctionner également puisque sa fonction est seulement éviter que le robot s'allonge totalement.
Lancement	Faire fonctionner le gyropode était plutôt compliqué. La première raison était que le programme proposé pour le tutoriel occupait plus que la mémoire du CoreUSB. Deuxièmement, ce Core était défectueux et il ne permettait pas le transfert de données. Finalement ces problèmes ont été résolus, le gyropode a bien marché mais il a fallu dédier trop de temps pour trouver la solution.
Modélisation	La modélisation du gyropode était facile grâce à le document suivi contenu dans la bibliographie. Le fait que le gyropode acquis utilise des moteurs pas à pas au lieu de courant continu a beaucoup compliqué cette étape et pour suivre le projet on a utilisé un modèle avec moteurs de courant continu. Les fonctions de transfert ont été calculées avec l'aide de Matlab.

Dessin sous Matlab	Écrire le code pour le dessin était laborieux mais finalement il est assez proche au réel et on peut regarder le mouvement du corps et des roues correctement. Le dessin est un peu moins fluide sous Simulink.
Simulation	Pour pouvoir simuler le comportement du gyropode on a supposé qu'il utilise des moteurs de courant continu égaux aux utilisés dans le document [1]. Pour avoir les fonctions de transfert on a utilisé l'espace d'état pour avoir le modèle et l'introduire sous Simulink. Finalement la simulation était possible et les résultats sont corrects malgré on peut avoir une réponse meilleure.
Autres	Après avoir résolu tous les problèmes, le gyropode a marchait très bien sans aucun erreur. La batterie de celui-ci est mieux car à la fin du projet elle n'est pas encore déchargé (elle a deux celles au lieu d'une).

Et la deuxième concerne le quadcopter :

Tableau 8: Caractéristiques générales résumées du quadcopter

Description	Commentaires du quadcopter
Processus d'assemblage	Ce processus était plus simple que dans le cas du gyropode car beaucoup de parties étaient déjà assemblés. On devait prêter attention à mettre tous les bras d'accord avec le schéma des moteurs (hélices A ou B et oranges ou noires en fonction de la position des moteurs 1, 2 3 et 4). La construction du joypad était aussi facile mais la structure se bouge dû aux vis.
Lancement	Le lancement du quadcopter était aussi plus facile que le gyropode. Le problème était que le joypad ne fonctionnait pas car l'écran restait blanc. La solution était très simple (pousser le bouton Reset du CoreRF) mais ce n'était pas facile à trouver parce qu'on pensait que c'était dû à l'alimentation. Le fonctionnement du quadcopter est correct.

Modélisation	La modélisation du quadcopter n'est pas 100% comme le robot réel car on a supposé qu'il fonctionne sous forme de croix. Pour pouvoir introduire le modèle sous Simulink était nécessaire suivre la modélisation de la source [2] laquelle est un peu différente pour la même raison. Assi, les paramètres introduits dans Matlab ne correspondent pas 100% avec les réels car on ne pouvait pas faire des expériences de manière précise.
Dessin sous Matlab	Le code écrit pour dessiner les mouvements du quadcopter était plus difficile car il y a plus formes qui tournent. Une remarque est qu'on ne peut pas regarder tous les mouvements linéaux dans une même figure parce que le dessin en 3D sous Matlab est très compliqué.
Simulation	Après avoir réalisé tous les schémas sous Simulink et après avoir défini tous les paramètres liés au robot était possible lancer des simulations et avoir des résultats. Le quadcopter fait des mouvements selon la commande d'entrée et ceux-ci sont raisonnables.
Autres	La batterie est trop petite et il faut la charger très souvent. Une des hélices tourne moins vite que les autres dû à un décalage dans le joypad.

Après avoir commenté et résumé les principales caractéristiques des deux robots on peut conclure que l'utilisation de ces deux robots à bas coût a provoqué certains problèmes pendant la mise en œuvre car quelques éléments étaient défectueux. Aussi, la batterie du quadcopter est trop petite, les chargeurs fournis ne marchent pas avec les prises européennes et le service après-vente peut être n'est pas très efficient car on a préféré résoudre les problèmes au lieu de retourner le produit. Par contre, expérimenter avec ce type de robots est mieux qu'utiliser les plus professionnels et chers puisque c'est facile de les casser quand on ne connait pas son fonctionnement ou quand on fait certains changements sur le code qui les fait bien marcher.

Par rapport à la simulation il faut dire que c'était dur arriver à avoir des résultats et que les modèles utilisés ne correspondent pas complètement aux robots réels. Grâce à

avoir les schémas sous Simulink en fonction des paramètres du quadcopter ou du gyropode, on pourra utiliser les mêmes fonctions et fichiers pour avoir des simulations sur autres robots différents juste changeant les variables caractéristiques comme la masse, la distance du CdG, etc. On peut conclure aussi qu'utiliser et expérimenter avec ces robots pas chers pour avoir un programme qui permet simuler et comprendre le comportement d'autres plus chers était peut-être la meilleure option car on préfère prendre les risques aux qui sont à bas coût. Il faut remarquer que la difficulté d'avoir le modèle et la simulation ne sont pas liés aux outils à bas coût car finalement on a beaucoup simplifié le système pour faciliter les calculs et les opérations.

Pour reprendre ce projet et pour le développer plus à l'avenir, on propose améliorer les modèles pour qu'ils se ressemblent plus aux robots réels. Aussi serait intéressant ajouter une antenne Arduino au quadcopter pour pouvoir avoir des valeurs des angles en temps réel dans l'ordinateur, ce qui s'appelle télémétrie. Car on ne pouvait pas satisfaire toutes les spécifications, on propose aussi faire des changements sur le gyropode et dessiner nouvelles parties pour modifier l'échelle.

Finalement, et come a clôture du projet, ci-dessous se trouve le tableau de spécifications montré au paragraphe 1.3 avec les indications de si on les a complété ou pas.

Tableau 9: Tableau de spécifications actualisé

Catégorie	Description	R/C
Objectif	Mise en œuvre d'outils de développement avec cartes embarquées à bas coût pour piloter diverses structures associés à la conception et la fabrication d'un robot de mise en situation	✓
Ressources	Microduino Self-Balancing Robot kit	✓
	Microduino Quadcopter kit	✓
	Microduino Joypad pour contrôler les robots	✓
	Oscilloscope Tektronix TDS 210 pour connaitre des paramètres des moteurs et pour résoudre problèmes	✓
	Tachymètre pour lire la vitesse angulaire des hélices du quadcopter	✓
	Voltmètre pour vérifier et réparer quelques parties des robots	✓
Résultats	Mise en œuvre des produits fournis	✓
	Analyser et documenter les codes de commande	-
	Construire des modèles de simulation du quadcopter sous Matlab/Simulink	✓

	Changer parties du gyropode pour une réduction homothétique et faire une maquette	✗
	Réalisation des différentes pièces en usinage ou impression 3D pour la réduction homothétique	✗
	Construire sous Scilab/Xcos et MATLAB/Simulink les modèles de simulation du gyropode	✓
	Étendre le développement de la commande à un gyropode échelle 1.	✗
Date limite	Présentation du projet dans la convocation Juin 2017	✓
Coûts	Minimiser le coût du matériel	✓
Autres	Faciliter l'ordre et la compréhension des résultats pour une éventuelle reprise du projet	✓

6. BIBLIOGRAPHIE

[1] Modélisation pour l'Automatique Numérique / Modélisation d'un gyropode - IFMA (18/12/2014)

[2] Modélisation et stabilisation d'un Micro Hélicoptère à Quatre rotors – Rémi Cazzaro (03/07/2007)

[3] Simulación de un péndulo invertido – Jose Luís Beltrán Alonso (06/12/2010)

[4] http://www.robotshop.com
Consulté le 27 Février, 2017

[5] https://wiki.microduino.cc/index.php/Open_Source_Self-balance_Robot_System
Consulté le 6 Février, 2017

[6] http://www.notcot.com/archives/2014/01/microduino-up-close.php
Consulté le 3 Mars, 2017

[7] https://www.microduino.cc/download
Consulté le 6 Février, 2017

[8] https://wiki.microduino.cc/index.php/Microduino-Quadcopter_Tutorial
Consulté le 27 Février, 2017

[9] https://wiki.microduino.cc/index.php/Microduino-Joypad_Getting_started
Consulté le 27 Février, 2017

[10] https://wiki.microduino.cc/index.php/Microduino-Module_BLE
Consulté le 6 Mars, 2017

[11] https://wiki.microduino.cc/index.php/Microduino-Module_Motion
Consulté le 6 Mars, 2017

[12] https://wiki.microduino.cc/index.php/Microduino-Shield_Stepper
Consulté le 6 Mars, 2017

[13] https://wiki.microduino.cc/index.php/Microduino-Module_CoreUSB
Consulté le 6 Mars, 2017

[14] https://wiki.microduino.cc/index.php/Microduino-Module_CoreRF

Consulté le 6 Mai, 2017

[15] https://wiki.microduino.cc/index.php/Microduino-Module_USBTTL

Consulté le 6 Mai, 2017

[16] https://wiki.microduino.cc/index.php/Microduino-Shield_QuadCopter

Consulté le 24 Mai, 2017

[17] http://fr.mathworks.com/matlabcentral/fileexchange/27703-draw-a-filled-circle?focused=5153742&tab=function

Consulté le 2 Mai, 2017

ANNEXES

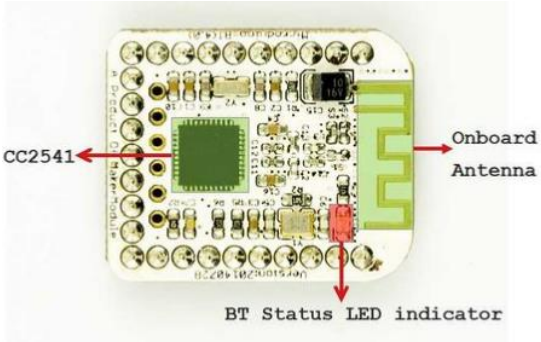
A. FICHE TECHNIQUE DES MODULES

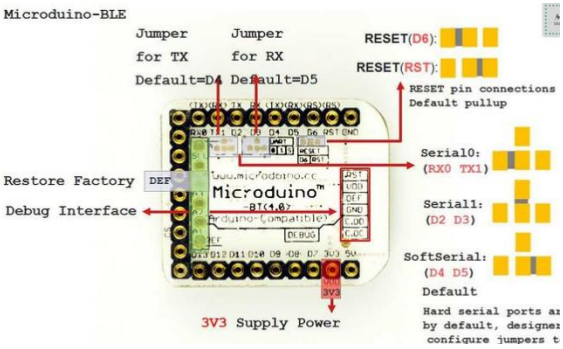
Cette annexe inclut des spécifications des modules Microduino utilisés pour faire fonctionner le gyropode, le quadcopter et le joypad. Pour plus d'information c'est possible visiter les sites web [10]-[16].

A.1. Microduino-module BT

Source [10]

"Microduino-BT is a BLE serial transparent transmission module based on CC2541 chip. It is custom-made for U-shaped 27PIN standard interface of Microduino. "

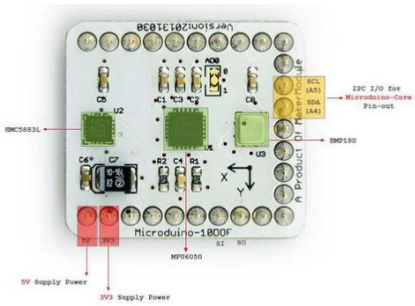




A.2. Microduino-module Motion

Source [11]

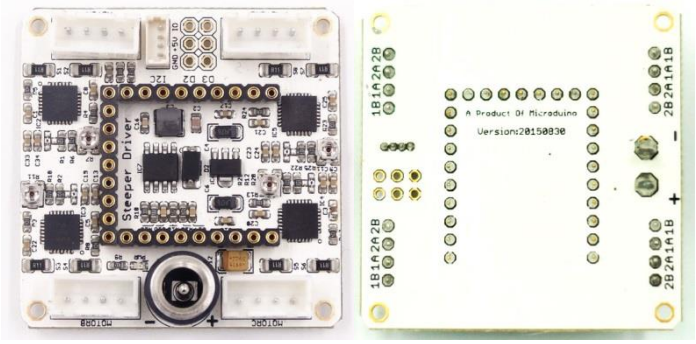
"Microduino-Motion(Microduino-10DOF) integrates four sensors including one sensor of 3-axis gyroscope and 3-axis accelerometer(MPU6050), one magnetic field strength sensor(HMC5883L) and a digital barometer sensor(BMP180). Adopting I2C interface, it can be widely used in automation control, such as aero modeling and self-balancing car."



A.3. Microduino-module Stepper

Source [12]

“Microduino-Shield Stepper is a DMOS microstep driving plate with a transverter and overcurrent protection, which can drive a two phase stepping motor under the full, half, 1/4, 1/8 and 1/16 stepping mode, and can operate four motors at most at the same time.”



A.4. Microduino-module CoreUSB

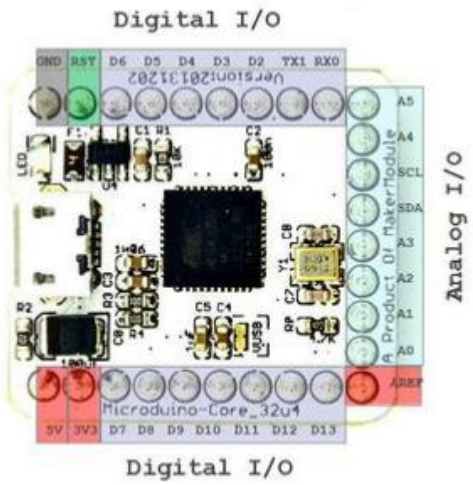
Source [13]

“Microduino-CoreUSB 8-bit microcontroller development board with the ATMEGA32U4 series as the core,and it is an open-source controller module compatible with Arduino Leonardo.

The difference between Microduino-CoreUSB and Microduino-core and Microduino-core+ is that the Microduino-CoreUSB contains a microcontroller and USB communication, equivalent to （Microduino-core + Microduino-USBTTL）, and its pins conform to the Microduino specification. Microduino uses java , the development environment of C language, same with Arduino. Players can use Arduino IDE, cooperating with software such as Flash or Processing, with Microduino and other electronic components,modules and sensors to make many funny interactive works.”

Specification

Flash	32 KB (ATMEGA32U4) . 4 KB during it is used to guide the program.
SRAM	2.5 KB (ATMEGA32U4)
EEPROM	1 KB (ATMEGA32U4)
Clock speed	16 MHz



Pin	Original Pin Name	Map Pin Name	Digital Pin	Analog Pin	interrupt	PWM	Serial	SPI	I2C	Power
1	VCC	+5V								+5V
2	VCC	+3V3								+3.3V
3	(OC0A/OC1C/#RTS)PB7	D7	D7			yes				
4	(OC1B/0C4B/ADC13)PB6	D8	D8	A8		yes				
5	(OC1A/#OC4B/ADC12)PB5	D9	D9	A9		yes				
6	(SS)PB0	D10	D10					SS		
7	(PDI/MOSI)PB2	D11	D11					MOSI		
8	(PDO/MISO)PB3	D12	D12					MISO		
9	(SCK)PB1	D13	D13					SCK		
10	AREF	AREF								
11	(ADC7/TDI)PF7	A0	D14	A0						
12	(ADC6/TDO)PF6	A1	D15	A1						
13	(ADC5/TMS)PF5	A2	D16	A2						
14	(ADC4/TCK)PF4	A3	D17	A3						
15	(SDA/INT1)PD1	SDA	D18		1				SDA	
16	(OC0B/SCL/INT0)PD0	SCL	D19		0	yes			SCL	
17	(ADC1)PF1	A6	D20	A6						
18	(ADC0)PF0	A7	D21	A7						
19	(RXD1/AIN1/INT2)PD2	D0	D0		2		1(RX)			
20	(TXD1/INT3)PD3	D1	D1		3		1(TX)			
21	(INT6/AIN0)PE6	D2	D2		4					
22	(T1/#OC4D/ADC9)PD6	D3	D3	A10		yes				
23	(T0/OC4D/ADC10)PD7	D4	D4	A11		yes				
24	(OC3A/#OC4A)PC6	D5	D5			yes				
25	(ICP3/CLK0/OC4A)PC7	D6	D6			yes				
26	RESET	RST								
27	GND	GND								GND

A.5. Microduino-module CoreRF

Source [14]

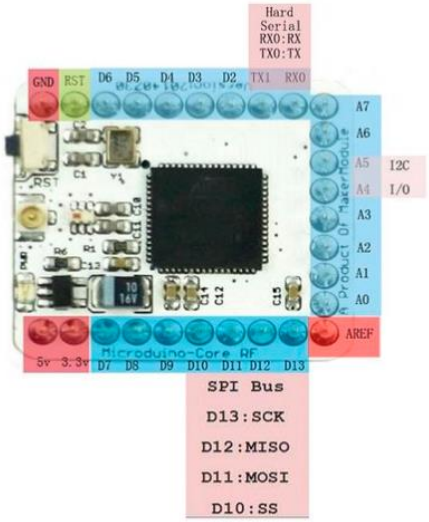
"Microduino CoreRF is an AVR core board with 802.15.4 Wireless Protocol integrated. It supports any wireless modules based on 802.15.4 Protocol, including Zigbee, MAC/6LoWPAN and RF4CE."

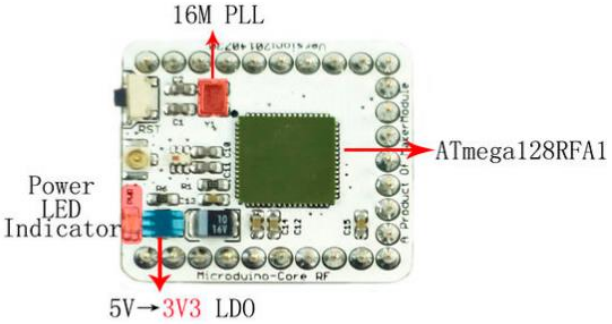
Specification

Adopt ATmega128RFA1 core chip

Power supply	3.3 V
Flash	128 KB
SRAM	16 KB
EEPROM	4 KB
Frequency of the time clock	16 MHz

Pin	Original Pin Name	Map Pin Name	Digital Pin	Analog Pin	interrupt	PWM	Serial	SPI	I2C	Power
1	VCC	+5V								+5V
2	VCC	+3V3								+3.3V
3	(OC0A/OC1C)PB7	D7	D7			yes				
4	(OC1B)PB6	D8	D8			yes				
5	(OC1A)PB5	D9	D9			yes				
6	(OC2A/SS)PB4	D10	D10			yes		SS		
7	(PDI/MOSI)PB2	D11	D11					MOSI		
8	(PDO/MISO)PB3	D12	D12					MISO		
9	(SCK)PB1	D13	D13					SCK		
10	AREF	AREF								
11	(ADC7)PF7	A0	D14	A0						
12	(ADC6)PF6	A1	D15	A1						
13	(ADC5)PF5	A2	D16	A2						
14	(ADC4)PF4	A3	D17	A3						
15	(SDA)PD1	SDA	D18		1				SDA	
16	(SCL)PD0	SCL	D19		0				SCL	
17	(ADC3)PF3	A6	D20	A6						
18	(ADC2)PF2	A7	D21	A7						
19	(RXD0)PE0	RX0	D0				0(RX)			
20	(TXD0)PE1	TX1	D1				0(RX)			
21	(RXD1)PD2	D2	D2		2		1(RX)			
22	(TXD1)PD3	D3	D3		3		1(TX)			
23	(OC3A)PE3	D4	D4			yes				
24	(OC3B)PE4	D5	D5		4	yes				
25	(OC3C)PE5	D6	D6		5	yes				
26	RESET	RST								
27	GND	GND								GND

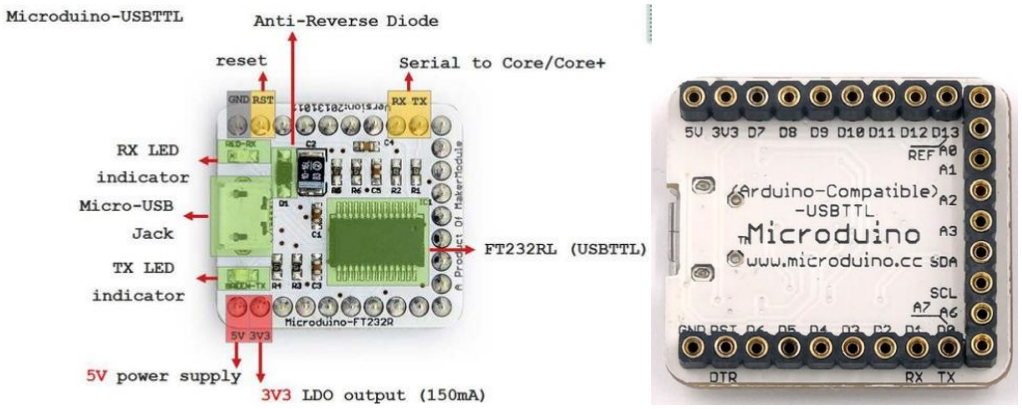




A.6. Microduino-module USBTTL

Source [15]

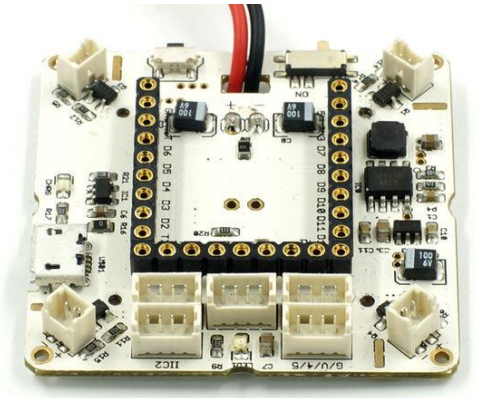
"Microduino-USBTTL is based on FDTI FT232RL chip (Arduino embedded driver). It can be stacked with Microduino-Core or Microduino-Core+, enable Microduino core modules to communicate with PC."



A.7. Microduino-Shield Quadcopter

Référence [16]

"Microduino-QuadCopter drive control board is designed especially for the quadcopter with a motor drive on the board. So, the only thing you need to is stack it with Microduino modules."



Specification

- Electrical Specification
  - 5V step-up: Adopt PT1301 step-up scheme
  - 3.3V stabilivolt: Adopt LP2985 step-down scheme
  - Power supply management: Adopt TP4056 power management scheme and MicroUSB power supply interface
  - Onboard power switch
  - Battery interface
- Motor drive: Adopt BSS123 field effect tube
  - Pins needed: D3,D9,D10,D11;
- D13 connected to status connection indicator.
- Onboard reset key
- Upin27 base board
- Wireless transmission extension interface

B. ÉTUDE DES COÛTS

B.1 Introduction

Cette annexe présente les coûts associés à la réalisation de ce projet de fin d'études, concrètement à la Conception des systèmes automatiques à bas coût. Dans ce document il y aura en détail les montants correspondants aux matériaux et aussi à la main-d'œuvre. Il faut souligner que les prix unitaires indiqués dans ce budget correspondent au mois de Juin du 2017 et qu'ils peuvent être modifiés. Ci-dessous il y a aussi les coûts indirects additionnels exprimés en pourcentage du coût total du budget.

B.2 Coûts des matériels

Les dépenses correspondantes au matériel utilisé dans ce projet de fin d'études sont présentées en détail dans le Tableau 10.

Tableau 10: Dépenses de matériel

Description du matériel	Quantité	Prix unitaire (€)	Subtotal (€)
Microduino *Self-Balancing robot kit* (gyropode)	1	108,45	108,45
Microduino Quadcopter kit (quadcopter + joypad)	1	145,61	145,61
Piles Alcalines AAA (4 x 1,5V)	1	5,95	5,95
Câble européen alimentation chargeur	1	4,99	4,99
Chargeur Tenergy TLP4000	1	21,00	21,00
		TOTAL	286,00€

B.3 Coûts de la main-d'œuvre

Les montants correspondants à la main-d'œuvre sont inclus dans le Tableau 11 avec une petite description du concept, la quantité et le prix unitaire.

Tableau 11: Dépenses de la main d'œuvre

Description	Quantité	Prix unitaire (€)	Subtotal (€)
Recherche bibliographique	40,00	12,00	480,00
Mise en œuvre des produits fournis	30,00	6,00	180,00
Analyse des robots et des programmes	35,00	30,00	1.050,00
Obtention des modèles mathématiques	50,00	30,00	1.500,00
Mise en œuvre des simulations	45,00	40,00	1.800,00
Expérimentation	20,00	12,00	240,00
Rédaction du projet	80,00	20,00	1.600,00
Réunions	20,00	20,00	400,00
		TOTAL	7.250,00

B.4 Coûts indirects

Les coûts indirects considérés pour une exécution correcte du projet sont un 16% du coût total de celui-ci. Les coûts indirects sont attribués à dépenses d'électricité, eau, transport, etc.

Tableau 12: Pourcentage et total des dépenses indirectes

Description	Pourcentage appliqué	Quantité	Subtotal
Dépenses indirectes	16.00%	7.536,00 €	1.205,76 €

B.5 Coût Total

En considérant les correspondants dépenses de matériel et de main-d'œuvre, ainsi que les coûts indirects pour la conception des systèmes automatiques à bas coût, le budget total s'élève à 8.741,76€ (HUIT MILLE SEPT CENTS QUARANTE ET UN EUROS ET SOIXANTE-SEIZE CENTIMES).

Aubière, 20 Juin du 2017

Marc López Montenegro