République Algérienne Démocratique et Populaire

Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

ECOLE NATIONALE POLYTECHNIQUE

DÉPARTEMENT D'ÉLECTRONIQUE

Mémoire de Master

En électronique

Thème :

Développement et implémentation d'un algorithme de Navigation en Robotique Mobile

Encadré par :

Dr. R.SADOUN

Réalisé par :

Mr.DIOUANI Sidali

Promotion : Juin 2015

Dédicaces

Je dédie ce modeste travail à :

? mes très chers parents, qu'ils trouvent ici l'expression de ma plus profonde gratitude,

? mon grand-père,

? mes frères et sœurs,

? tous ceux qui ont fait partie de mon parcours académique.

Sidali

Remerciements

En premier lieu et avant toute chose, Louange et Action de grâce à Allah le tout puissant pour toutes ses grâces et ses faveurs, pour m'avoir accordé la chance d'étudier à l'École Nationale Polytechnique et m'avoir donné le courage et la force d'accomplir ce modeste travail.

Je tiens à remercier Monsieur R.SADOUN, aussi Monsieur Z.MOUSSAOUI ainsi que tous les membres de l'équipe CYBORG, (Mr. Mahfoud BELDELMI, Mr Mohamed ACHOUR) pour nous avoir ouvert les portes de leurs laboratoire et fournis le matériel, mais aussi pour leur aide lors des phases d'implémentation.

Enfin, je tiens également à remercier les membres de notre jury, Professeur Mhania GUERTI pour l'honneur qu'elle nous fait de le présider, Monsieur Mourad ADNANE et Monsieur Ahmed BOUZID pour l'intérêt qu'ils ont porté à notre travail en acceptant d'être examinateurs, leurs remarques et suggestions nous ont permis d'améliorer la qualité de notre travail.

ملخص

يتناول هذا العمل تنفيذ وحدة تحكم غامض للقيادة في مجال الروبوات المتنقلة.
تم عرض العموميات على الروبوات المتنقلة ثم وضعت القضايا المتعلقة بالقيادة.
يتناول الجزء الأول الطريقة المختارة لأداء الملاحة ثم يتم هذا التنفيذ، ومنهجية التنفيذ.
تم كسر نهج أسفل إلى مرحلتين، كانت الخطوة الأولى لمحاكاة الروبوت المتحرك مع وحدة القيادة، والخطوة الثانية للنظر في تحقيق النموذج من خلال تنفيذ الخوارزمية على الآلة.
الكلمات المفتاحية : الروبوت المتحرك, خوارزمية القيادة.

Résumé

Ce travail traite de l'implémentation d'un régulateur floue pour la navigation en robotique mobile, des généralités sur la robotique mobile sont présentés puis les questions relatives à la navigation sont développés.
La première partie traite de la méthode choisie pour effectuer la navigation puis ça mise en œuvre est effectuée, la méthodologie d'implémentation entreprise est celle du Model-Based Design.

L'approche suivie s'est décomposée en deux phases, une première étape à été de simuler un robot mobile avec le module de navigation (synthèse du régulateur), la seconde étape à viser en la concrétisation du modèle via l'implémentation de l'algorithme sur un robot mobile réel développé par Cyborg.

Mots clés : Robotique, Navigation, Logique Floue.

Abstract

This work deals with the implementation of a fuzzy controller for navigation in mobile robotics, generalities on mobile robotics are presented and issues related to navigation are developed.
The first part deals with the method chosen to perform the navigation then that implementation is made, the implementation methodology is that of "Model-Based Design".

The approach was broken down into two phases, the first step was to simulate a mobile robot with the navigation module (synthesis of the regulator), the second step to consider in the realization of the model through the implementation of the algorithm on a real mobile robot developed by Cyborg.

Keywords : Robotics, Navigation, Fuzzy Logic.

Table des matières

Table des figures

Introduction générale

Le développement interdisciplinaire actuel a permis l'évolution des robots mobiles et l'accroissement de leur usage pour des applications divers, pour citer l'assistance aux personnes handicapées, aide aux tâches d'entretien et de maintenance, etc.

L'évolution autonome de tels systèmes dans un environnement extérieur nécessite au minimum des tâches de base qui sont la localisation, la planification et la navigation.

Ce travail a pour but la mise en œuvre d'un régulateur flou pour concevoir un système de navigation répondant au besoin des compétitions "Eurobot" en robotique mobile.

Le premier chapitre présentera les généralités sur la robotique mobile, on entamera dès le deuxième chapitre les parties développement de la commande par logique floue, ou des simulation sur un modèle virtuel du robot seront effectués.

La dernière étape étant l'implémentation du système de navigation sur un robot réel ("Robucup" de chez Cyborg) et enfin les résultats des tests seront exposés.

Chapitre 1

Robotique Mobile

Nous aborderons dans ce chapitre les généralités et notions sur la robotique mobile, d'abord les classes de robots, puis les questions relatives à la localisation et la navigation en robotique mobile, enfin nous nous intéresserons à la modélisation du robot du type différentiel de vitesse.

1.1 Définition de la robotique mobile

Bien souvent, quand on parle de robotique mobile, on sous entend robots mobiles à roues. Ce sont en effet les systèmes les plus étudiés, parce qu'ils sont plus simples à réaliser que les autres types de robots mobiles, ce qui permet d'en venir plus rapidement à l'étude de leur navigation. Ce type de robots est notamment très souvent utilisé pour l'étude des systèmes autonomes.

L'appellation "robot mobile" désigne généralement un véhicule équipé de capacités de perception, de décision et d'action qui lui permettent d'agir de manière autonome ou semi-autonome dans un environnement complexe, parfois évolutif, partiellement connu ou inconnu, et d'exécuter les tâches programmées sans intervention humaine ou avec une intervention réduite.

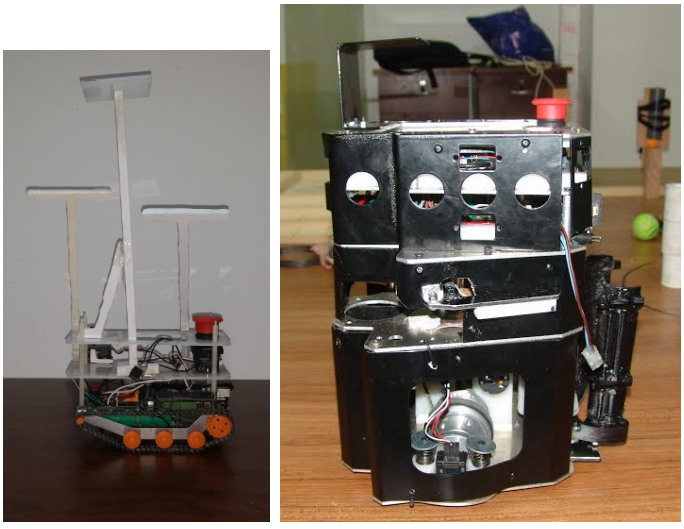


FIGURE 1.1: Exemple de robots mobiles (Cyborg)

Il existe deux principaux modes de fonctionnement pour un robot mobile : télé-opéré et autonome. En mode télé-opéré, une personne pilote le robot à distance. Elle donne ses ordres via une interface de commande (joystick, clavier. . .), et ceux-ci sont envoyés au robot via un

lien de communication (internet, satellite . . .). Le robot doit donc obéir aux ordres de l'opérateur qui perçoit l'environnement autour du robot, par différents moyens (camera, radar. . .), de manière à donner des ordres adaptés au robot. A l'inverse, en mode autonome le robot doit prendre ses propres décisions. Cela signifie qu'il doit être capable à la fois de percevoir correctement son environnement, mais également de savoir comment réagir en conséquence, suivant le niveau d'autonomie. C'est à lui d'envisager son parcours et de déterminer avec quels mouvements il va atteindre son objectif.

1.2 Classes des robots mobiles à roues

Pour déplacer un robot mobile sur une surface, il faut au moins deux degrés de liberté, donc deux moteurs. Et c'est aussi la combinaison du choix des roues et de leur disposition qui confère à un robot son mode de locomotion propre, on rencontre principalement trois types de robot :

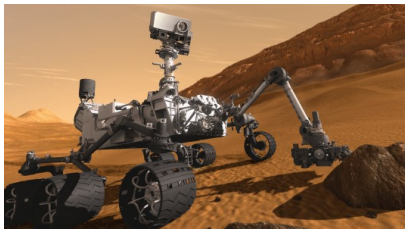


FIGURE 1.2: Curiosity sur Mars

— Robot uni-cycle : Actionné par deux roues indépendantes et possédant éventuellement un certain nombre de roues folles assurant sa stabilité ;
— Robot tricycle (et de type voiture qui partagent des propriétés cinématiques proches) : Constitué de deux roues fixes de même axe et d'une roue centrée orientable placée sur l'axe longitudinal du robot. Le mouvement est conféré au robot par deux actions (la vitesse longitudinale et l'orientation de la roue orientable) ;
— Robot omnidirectionnel : Un robot mobile est dit omnidirectionnel si l'on peut agir indépendamment sur les vitesses (vitesse de translation selon les axes x et y et vitesse de rotation autour de z).

1.3 Localisation en robotique

Parmi les différentes méthodes et technologies de mesure de mouvement nous nous intéresserons dans cette partie aux codeurs optiques (odomètres). Les systèmes optiques sont les plus répandus en robotique mobile. Le codeur optique rotatif fonctionne selon le principe du balayage photoélectrique d'un disque sur lequel a été déposé un réseau radial plus ou moins complexe de trous (ou de traits opaques).

Le codeur incrémental est le capteur le plus utilisé pour mesurer les variations de position et de l'orientation d'un robot mobile à roues car il est peu onéreux et facile à interfacer. Monté sur l'axe du moteur ou sur l'axe de la roue, il délivre des informations de rotation élémentaire qui, par intégration, donnent une mesure du mouvement global

1.3.1 Techniques de localisation du robot

La localisation du robot est une question crutiale lorsqu'il sagit de commander un robot mobile dans l'état de l'art divers techniques existent, on cite la localisation par triangulation, la Localisation odométrique, inertielle, à partir d'images de profondeur ou encore celle à partir de la vision.

La méthode choisit pour la suite de l'implémentation est la localisation odométrique, les raisons d'un tel choix sont multiples (facilité de mise en oeuvre, matériel peu couteux ...).

1.3.2 Odométrie

A priori utilisable dès que la structure de locomotion est à roues, la localisation odométrique est extrêmement répandue dans le contexte des environnements intérieurs, où les robots évoluent sur un plan. Elle est par contre moins utilisée en environnements naturels, d'une part parce que le robot évolue dans les trois dimensions de l'espace, ce qui nécessite l'utilisation d'au moins un inclinomètre pour mesurer l'angle de site du robot, et d'autre part parce que les principaux problèmes liés à l'odométrie (impossibilité de mesurer les glissements et dérapages) sont bien plus fréquents et importants qu'en environnements intérieurs.

L'odométrie est une technique permettant d'estimer la position d'un mobile à partir de données sur le déplacement des roues fournis généralement par des capteurs.

1.4 Navigation en Robotique Mobile

La programmation d'un robot pour la navigation mais aussi l'évitement utilisant les méthodes traditionnelles deviens rapidement complexe et fastidieux si l'on veut traiter tous les cas ou faire évoluer le programme pour rajouter des capteurs ou si l'on change de robot.

1.4.1 Etat de l'art en Navigation

Dans la littérature il existe différentes approches au problème de navigation et/ou évitement d'obstacle on cite l'utilisation de techniques :

— Neuro-flou [1]
— algorithme génétique [2]
— VFH [3]
— Le Champs de potentiels [4]

Enfin d'autres approches existent aussi tels les approches via les méthodes stochastiques, cela dit elle sont lourdes et requirent bien plus de connaissances une méthode assez simple est l'approche logique floue, dans notre cas on s'intéresse précisément à l'élaboration d'une loi de commande en utilisant la logique floue.

1.4.2 Logique Floue

À l'inverse de l'algèbre de Boole, la logique floue autorise la valeur de vérité d'une condition à parcourir un autre domaine que la paire vrai,faux. En logique floue, il y a des degrés

dans la satisfaction d'une condition.

Un ensemble flou est un ensemble pour lequel la fonction d'appartenance prend des valeurs dans tout l'intervalle [0,1] et pas seulement les valeurs 0 et 1. La fonction d'appartenance va définir le degré d'appartenance de l'élément considéré à l'ensemble dont on parle.

Pour raisonner avec les représentations numériques/linguistiques des variables d'entrée et de sortie, le système à base de règles floues opère selon trois étapes successives : A partir des données numériques issues des capteurs sont calculés pour chaque entrée les degrés d'appartenance aux classes utilisées.

Un système a base de logique flou opére en trois étapes :

— Définition des fonctions d'appartenances
— Définition des règles entre entrées et sorties
— Deffuzification (centre de gravité)

Fuzzification

— Evaluation des conditions : chaque condition est de la forme : SI variable = Valeur-Linguistique. Le niveau de vérité d'une condition correspond au degré d'appartenance de la valeur numérique de la variable à la classe linguistique considérée.
— Agrégation des conditions : plusieurs méthodes sont possibles, une des plus utilisées étant celle du Robot Flou Mandani : le ET logique est traduit par l'opérateur Min et le OU logique par l'opérateur Max
— Calcul de la sortie : chaq'une est une affectation de la forme Variable = ValeurLinguistique. La méthode la plus utilisée est de considérer le degré de vérité calculé pour les prémisses comme le degré de validité de l'affectation.

Deffuzification.

Il s'agit pour chaque variable de sortie, de passer des degrés de validité calculés pour chaque état à des grandeurs numériques. La méthode la plus utilisée est celle du centre de gravité. On calcul le centre de gravité formé par l'union des degrés de validité.

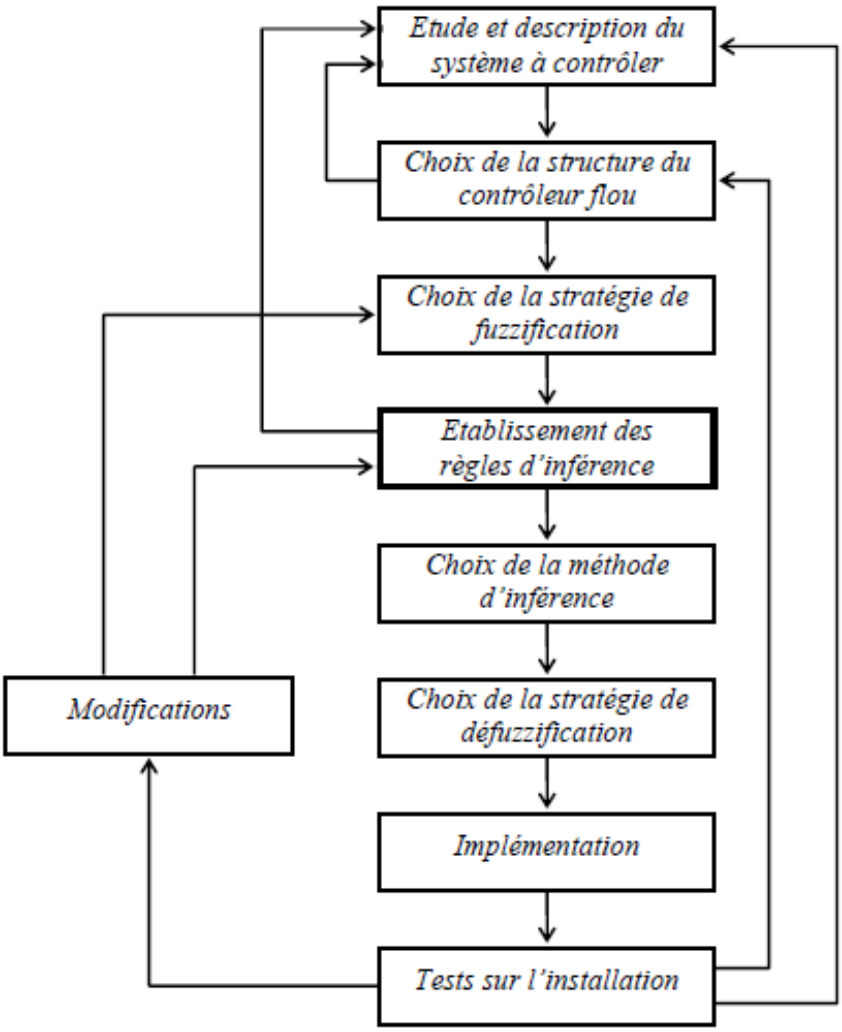


FIGURE 1.3: Procédure de conception d'un contrôleur flou
[4]

Un exemple de système basé sur la logique floue est le régulateur floue ou qui fera office de commande de vitesse.

1.5 Modélisation géométrique

1.5.1 Modèle robot mobile avec différentiel de vitesse

On s'intéresse dans notre étude au cas de robot mobile du type différentiel de vitesse, ou robot différentiel c'est-à-dire que pour tourner, il faut faire tourner les roues plus vite d'un côté que de l'autre, le schéma suivant explique le comportement d'un tel Robot.

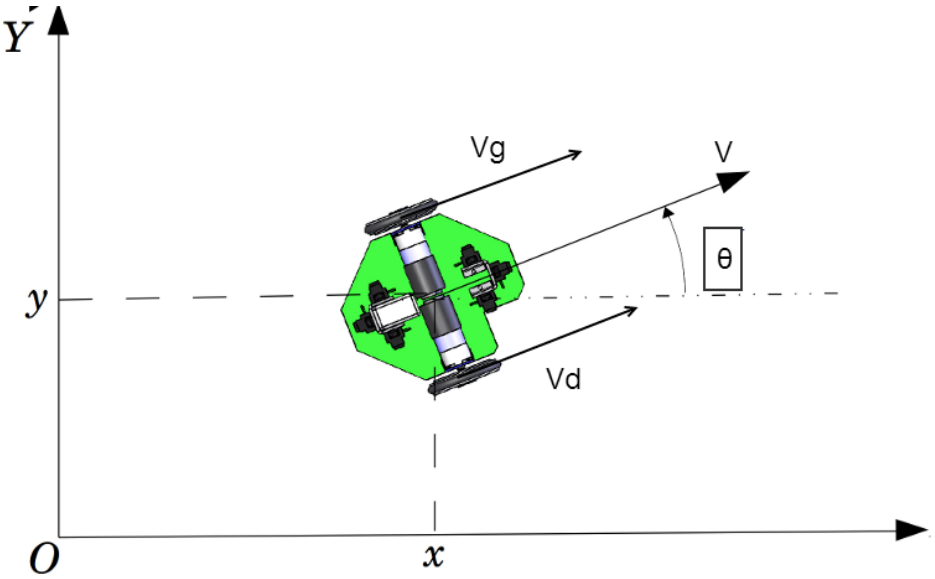


FIGURE 1.4: Modèle Robot Différentiel

$$V = (Vd + Vg)/2 \tag{1.1}$$

— V vitesse linaire du robot (m/s)
— V_g vitesse de la roue gauche (m/s)
— V_d vitesse de la roue droite (m/s)
— L moitié de la distance entre les deux roues (entraxe) (m)

1.5.2 Modèle de déplacement

Soit dans notre cas un robot dont le déplacement est contrôlé par le différentiel de vitesse entre les deux roues motrices.

Si on suppose que la trajectoire du robot est un cercle de rayon R parcouru à la vitesse angulaire $\frac{d\theta}{dt}$ (R>0 si le cercle est parcouru dans le sens trigonométrique), alors on a :

$$v = R\frac{d\theta}{dt} \tag{1.2}$$

Dans ce cas, les vitesses des roues sont données par :

$$\begin{cases} v_g &= (R - \frac{e}{2})\,\frac{d\theta}{dt} &= (R - \frac{e}{2})\frac{v}{R} \\ v_d &= (R + \frac{e}{2})\,\frac{d\theta}{dt} &= (R + \frac{e}{2})\frac{v}{R} \end{cases} \tag{1.3}$$

Nous avons donc construit un modèle direct du déplacement du robot (i.e. un modèle permettant de calculer les vitesses v_g etv_d des roues en fonction des paramètres de la trajectoire globale v et R.

L'odométrie va cependant nécessiter la connaissance du modèle inverse du déplacement du robot : connaissant les mesures odométriques, nous cherchons à retrouver les paramètres de la trajectoire. Nous pouvons supposer qu'à chaque instant et durant un intervalle de temps très court, la trajectoire du robot s'apparente à un cercle. Il nous suffit donc d'inverser le

modèle direct précédemment construit pour un cercle, et nous pourrons reconstruire le rayon de courbure local de la trajectoire et la vitesse du robot.

L'inversion du système précédent nous donne :

$$\begin{cases} v &= \dfrac{v_g + v_d}{2} \\ R &= \dfrac{e}{2}\dfrac{v_d + v_g}{v_d - v_g} \end{cases}$$

1.5.3 Détermination de la position

Nous sommes maintenant en mesure de mettre à jour la position du robot en temps réel :

à chaque instant, les mesures odométriques nous donnent les déplacements des roues d_g et d_d depuis l'instant précédent le modèle inverse (dans lequel l'intervalle de temps a été éliminé et les vitesses remplacées par des déplacements) nous permet de calculer le déplacement d du robot et le rayon de courbure instantané R de la trajectoire. on calcule le changement d'orientation $d\theta$ du robot et les coordonnées du centre O du cercle trajectoire :

$d\theta = \frac{d}{R}$ $\begin{cases} x_O &= x - R\cos(\theta) \\ y_O &= y - R\sin(\theta) \end{cases}$

On met à jour la position du robot :

$$\begin{cases} \theta &\leftarrow \theta + d\theta \\ x &\leftarrow x_O + R\cos(\theta) \\ y &\leftarrow y_O + R\sin(\theta) \end{cases} \tag{1.4}$$

Chapitre 2

Simulation

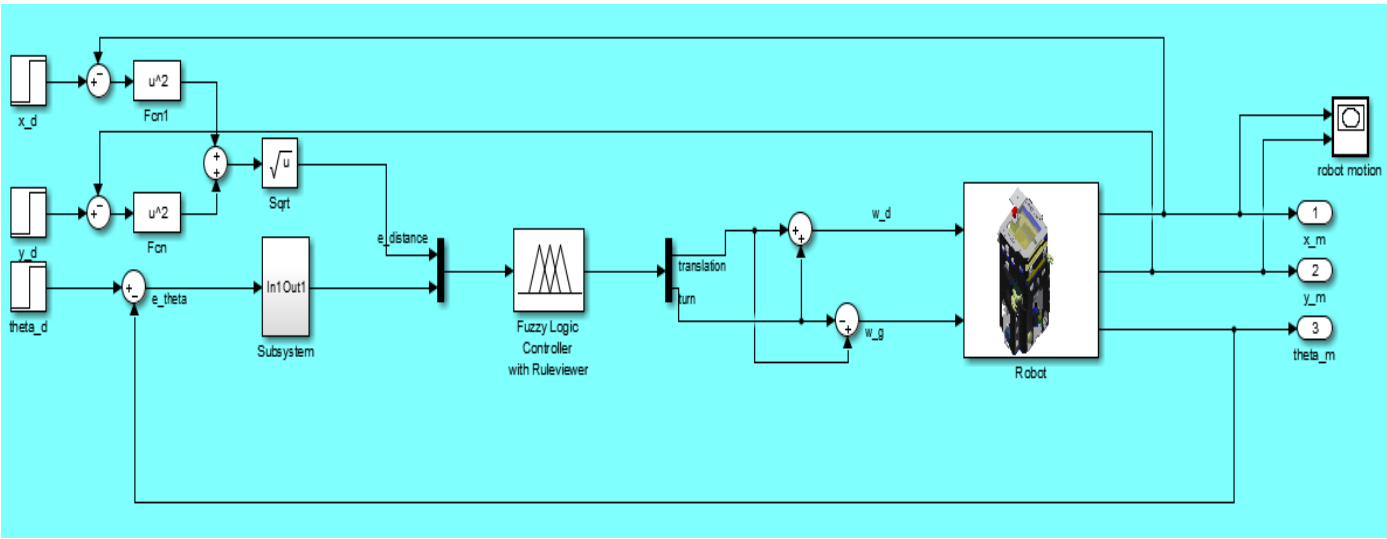
Ce chapitre se proposera d'exposer le modèle Simulation, nous permettant de vérifier le bon fondement de la démarche "Régulateur floue pour la navigation", nous aurons tout d'abord besoin de présenter le modèle du robot sous Simulink et les différents blocs notamment celui de l'odométrie pour la détermination la position du robot (x,y,θ (angle d'attaque)) mais aussi et surtout le bloc de régulation floue symbolisée par la toolbox Fuzzy Logic.

Deux approches seront explorées une première est la Navigation Floue Simple une seconde plus intéressante est la Navigation Floue avec Accostage .

2.1 Modèles de simulation

2.1.1 Modèle Navigation avec Accostage

Ce modèle à pour entrées les consignes sur la position et l'orientation d'arrivée du robot, c'est un modèle dit de navigation avec accostage [5] et [6].



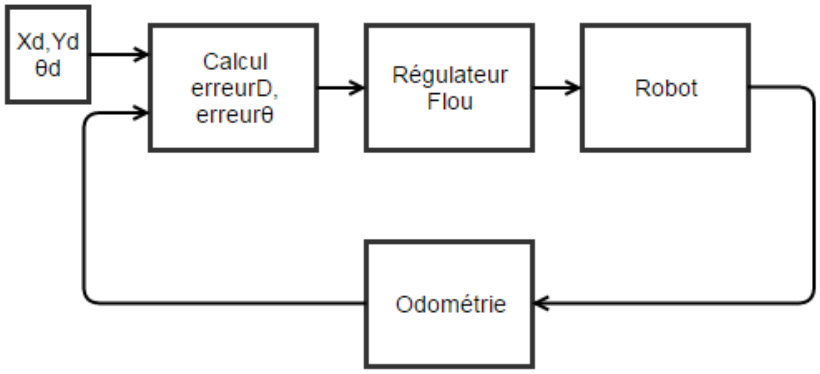


FIGURE 2.1: Modèle Accostage Flou

2.1.2 Modèle Navigation sans Accostage

Ce modèle à pour entrées les consignes sur les coordonnées souhaités du robot, c'est un modèle dit de navigation classique. [8].

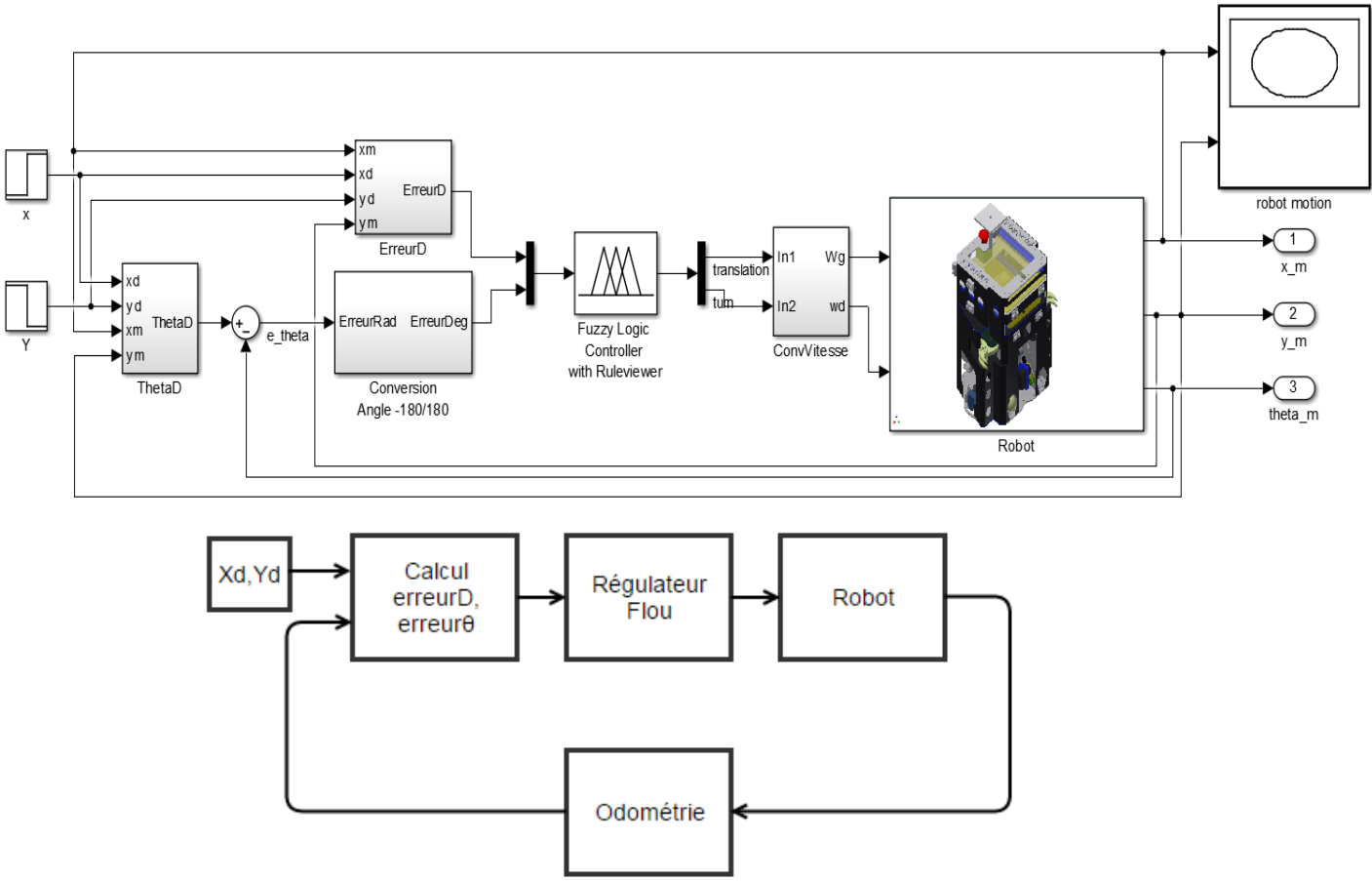


FIGURE 2.2: Modèle Navigation Floue

Le régulateur flou se charge de déterminer les consignes de vitesses permettant d'aller à la position désirée seulement.

2.1.3 Comparaison

Contrairement au modèle de Navigation seul, celui avec Accostage a pour but d'atteindre en plus de la consigne de position (X, Y), celle sur l'orientation du robot (ou angle d'attaque) à l'arrivée (Cahier de charge).

Dans les tests qui suivent nous effectuerons seulement la navigation sans prendre en compte l'angle d'arrivée, ce sera une première étape pour la validation du Cahier de charge initial.

2.2 Description du Modèle de Simulation

On distingue trois parties importantes, tout d'abord le régulateur qui se charge du calcul des vitesses permettant d'aller vers la consigne de position, l'entrée du bloc "Regulateur Floue" est l'erreur sur la distance :

$$Erreur_d = \sqrt{(X_d - X_m)^2 + (Y_d - Y_m)^2} \tag{2.1}$$

L'erreur sur l'orientation est calculée comme étant :

$$erreur_\theta = \arctan(\frac{Y_d - Y_m}{X_d - X_m}) - \theta_m \tag{2.2}$$

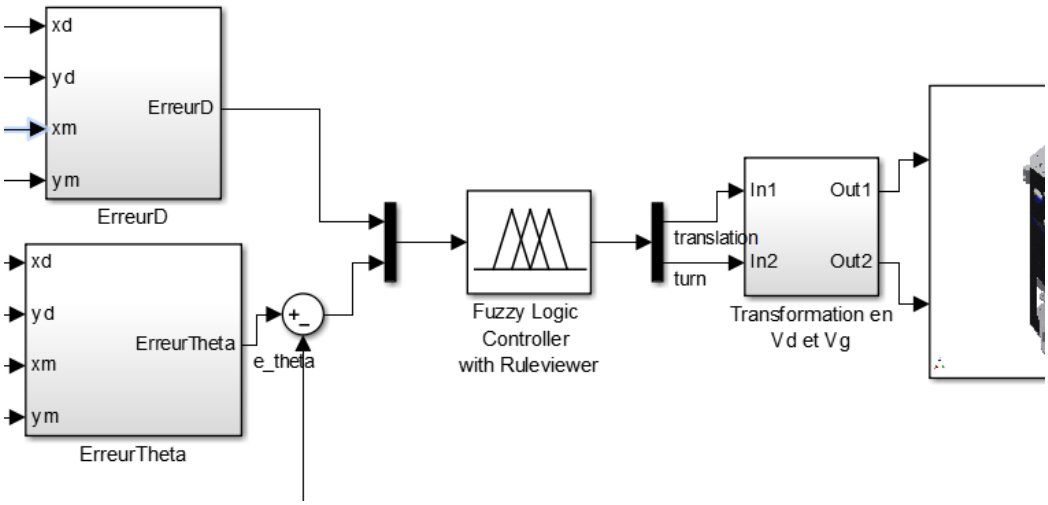


FIGURE 2.3: Bloc Logique Floue

Le bloc central est celui modélisant le robot, via les fonctions de transfert de deux moteurs à courant continu :

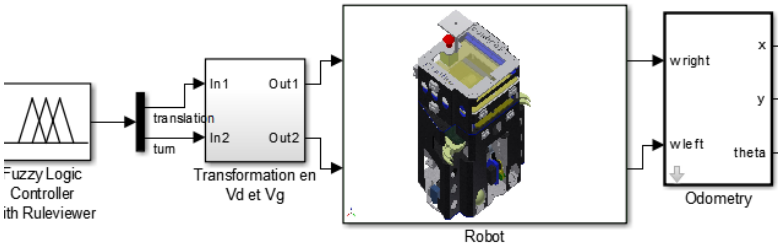


FIGURE 2.4: Bloc Logique Floue

Enfin, le dernier bloc est celui de l'odomètrie.

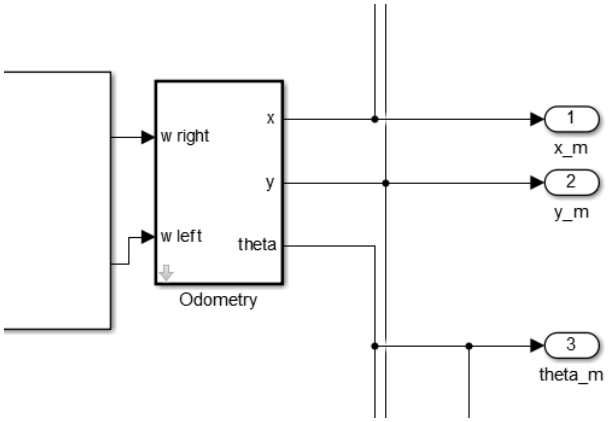


FIGURE 2.5: Odometrie

2.2.1 Modèle du Robot

Pour les besoins de simulation, le robot à été modélisé par la fonction de transfert décrivant le comportement de ses deux moteurs à courant continu.

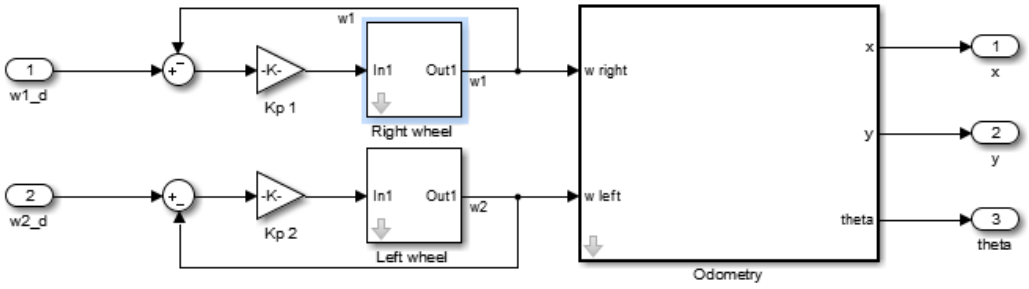


FIGURE 2.6: Modèle Robot Simulation

On cite deux procédures possibles pour simuler le comportement des moteurs et aboutir à un modèle virtuel :

— la première est de consulter les documents constructeur, afin d'obtenir les paramètres électro-mécaniques moteur à incorporer au modèle, on détermine grâce à cela la fonction de transfert du moteur.

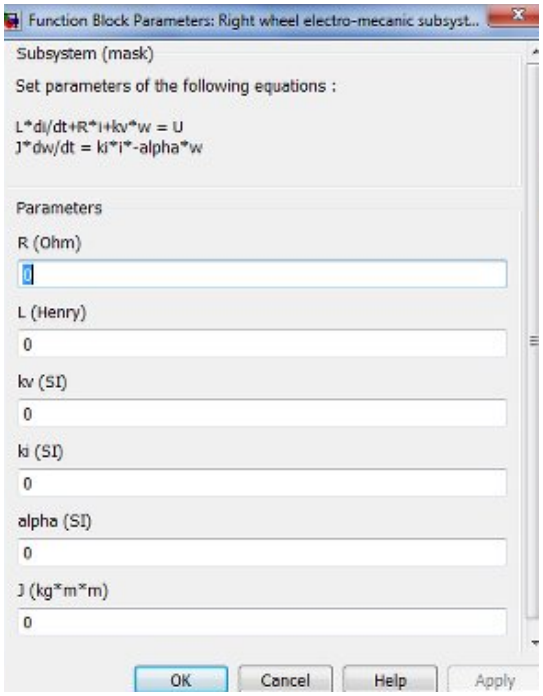


FIGURE 2.7: Insertion des Paramètres électromécaniques

— Une seconde méthode est d'utiliser des techniques d'identification on cite celle dite de Broîda [7] permettant d'approximer en un système du premier ordre en observant la réponse indicielle du système en BO.

La fonction de transfert du moteur est donnée :

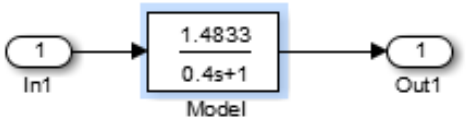


FIGURE 2.8: Fonction de transfert du moteur CC

2.2.2 Asservissement des vitesses de rotation

On à une boucle de régulation de la vitesse de chaque moteur. (Régulateur de type Proportionnel pour la simulation).

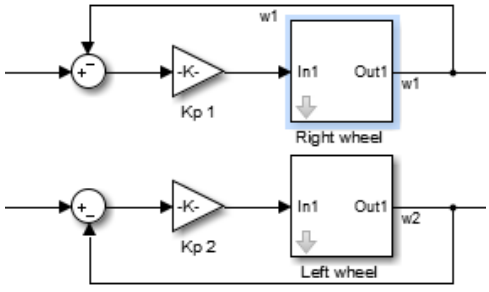


FIGURE 2.9: Fonction de transfert du moteur CC

Remarque :

— Lors de l'implémentation cette partie se fera sur la carte asservissement, en utilisant le programme C++ de la bibliothèque asservissement.
— Le modèle précédent ne prend pas en compte l'inertie du robot, les frottements et l'éventuel glissement (Environnement réel).

2.2.3 Odométrie

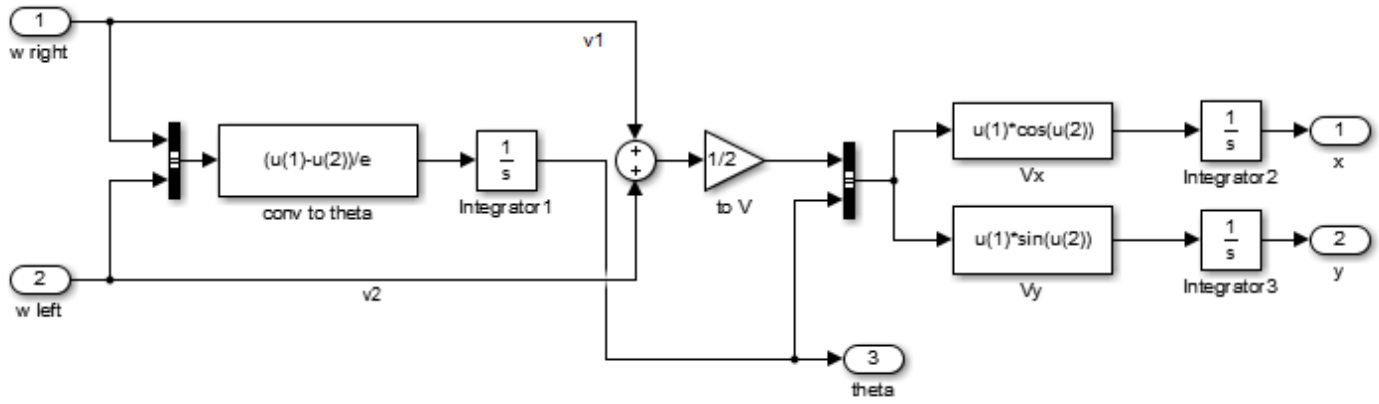


FIGURE 2.10: Bloc Odométrie

Ce bloc se charge de calculer la position et l'orientation du Robot sachant la vitesse de rotation des deux roues.

2.2.4 Logique floue et Synthèse du régulateur

2.2.4.1 Entrées / Sorties

Tout d'abord on défini les entrées et sorties du régulateur, les deux entrées sont l'erreur sur la distance et L'erreur l'orientation du robot, la position est calculée connaissant la position et l'orientation (calculés en temps réel par le module d'odométrie).

Les sorties du régulateur flou sont la vitesse de translation et celle de rotation (issue de l'erreur sur l'orientation), pour atteindre le but désiré selon l'orientation désirée.

— entrées : $erreurD$, $erreur\Theta$
— sortie : $V_{Translation}$, $V_{Rotation}$

2.2.4.2 Fonctions d'appartenance

Avant de définir la relation entre les entrées et sorties, nous allons définir les ensembles "floues" et les fonctions d'appartenance.

Pour l'erreur sur la distance on défini par exemple les états : Nulle (Z), Petite(P), Moyenne(M), Grande(G), Très Grande (TG).
On spécifie pour chaque cas une valeur en cm, des paramètres sont tirés de [5]

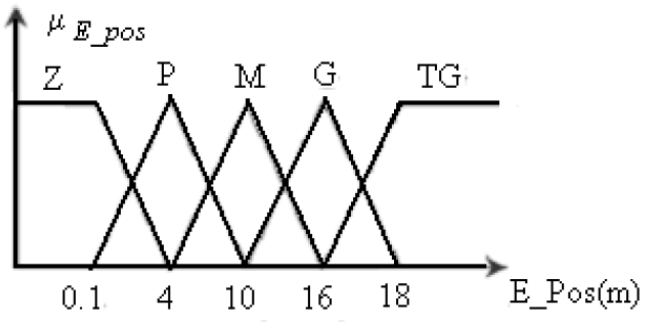


FIGURE 2.11: Erreur en Position

Pour l'erreur sur l'orientation (de -180 à 180)on défini par exemple les états : Négative Grande (NG), Négative Moyenne(NM), Négative Petite(NP), Nulle(Z), Positive Petite(PP), Positive Moyenne (PM), Positive Grande(PG).
On introduit dans chaque cas une valeur d'orientation en degrés.

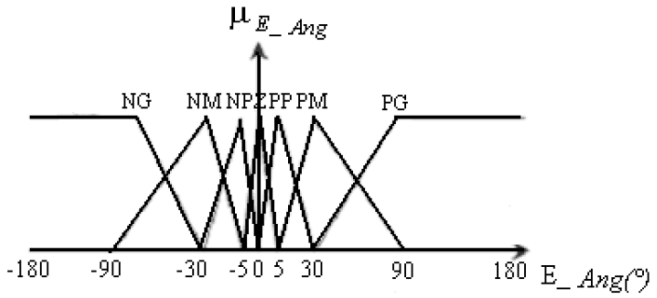


FIGURE 2.12: Erreur sur l'angle

On procède de manière similaire pour les Sorties Sauf que dans ce cas les fonctions d'appartenances seront des Dirac. (cela contribue aussi à alléger la mémoire vive lors de l'implémentation)

Pour la Vitesse de translation :

Les états sont les suivants : Nulle (Z), Faible (F), Moyenne (M), Grande (G), Très Grande (TG).

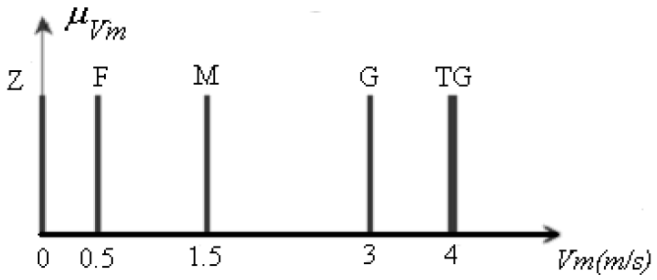


FIGURE 2.13: Valeurs vitesse de translation

Même chose pour la Vitesse de rotation :
Négative Grande (NG), Négative Moyenne (NM), Négative Petite(NP), Nulle(Z), Positive Petite(PP), Positive Moyenne (PM), Positive Grande(PG).

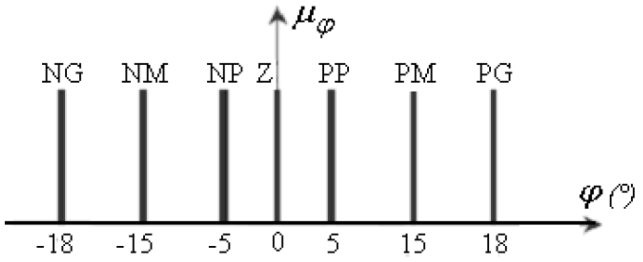


FIGURE 2.14: Valeurs Vitesse rotation

On à plus qu'a définir les relations entre entrées et sorties, appelées règles.

2.2.4.3 Définition des règles

Principe : les règles sont l'ensemble des relations entre entrées et sorties, ce sont des relations logiques du type :
"Si (ErreurD = GrandePositive et ErreurTheta = Petite) => (VitesseTranslation = Négative Grande et VitesseOrientation = Nulle").

On aboutit à un tableau de relations du type :

Erreur de Position (E_Pos)			Erreur Angulaire (E_Ang) NG	NM	NP	Z	PP	PM	PG
	Z	φ	PG	PG	PM	Z	NM	NG	NG
		Vm	Z	Z	Z	Z	Z	Z	Z
	P	φ	PG	PM	PM	Z	NM	NM	NG
		Vm	F	F	F	F	F	F	F
	M	φ	PM	PM	PP	Z	NP	NM	NM
		Vm	F	F	M	M	M	F	F
	G	φ	PM	PP	PP	Z	NP	NP	NM
		Vm	F	M	G	G	G	M	F
	TG	φ	PP	PP	Z	Z	Z	NP	NP
		Vm	F	M	G	TG	G	M	F

FIGURE 2.15: Tableau de règles

2.2.4.4 Déffuzification

Cette étape est effectuée automatiquement par le module "Logique Floue", il s'occupe de calculer les sortie selon la méthode du centre de gravité. On calcul le centre de gravité formé par l'union des degrés de validité.

2.2.5 Paramétrage du régulateur

On utilise le bloc "Fuzzy Logic" disponible sous Simulink pour concevoir le régulateur afin d'effectuer les tests.

2.2.5.1 Définition des fonctions d'appartenance

Pour la définition des fonctions d'appartenance des changements ont été apporter par rapport à ceux défini dans la section précédente, en fait le paramétrage du régulateur se fait tout au long du processus que ce soit pour la simulation ou lors de l'implémentation.

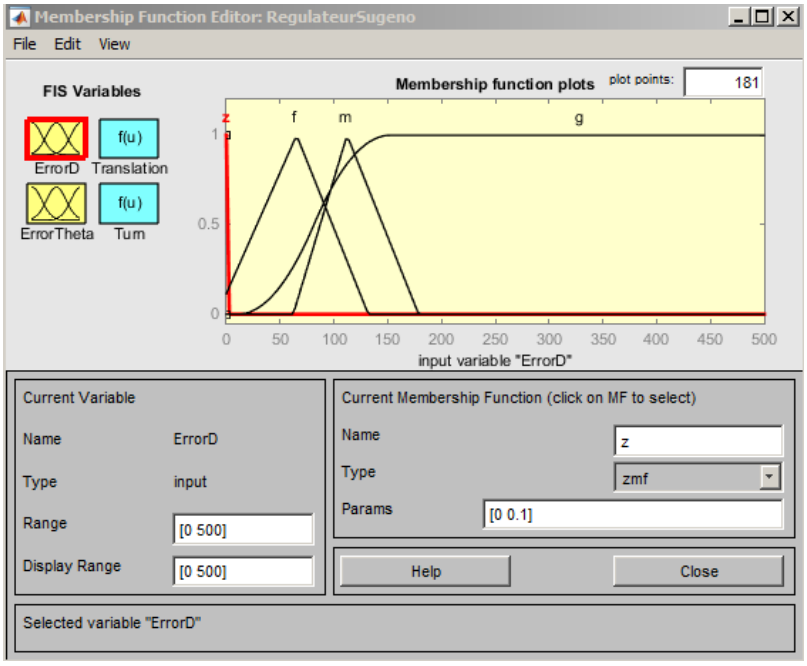


FIGURE 2.16: Erreur D

Il faut prévoir un temps assez important pour ajuster les paramètres, pour aboutir à ceux qui donnent des résultats concluants.

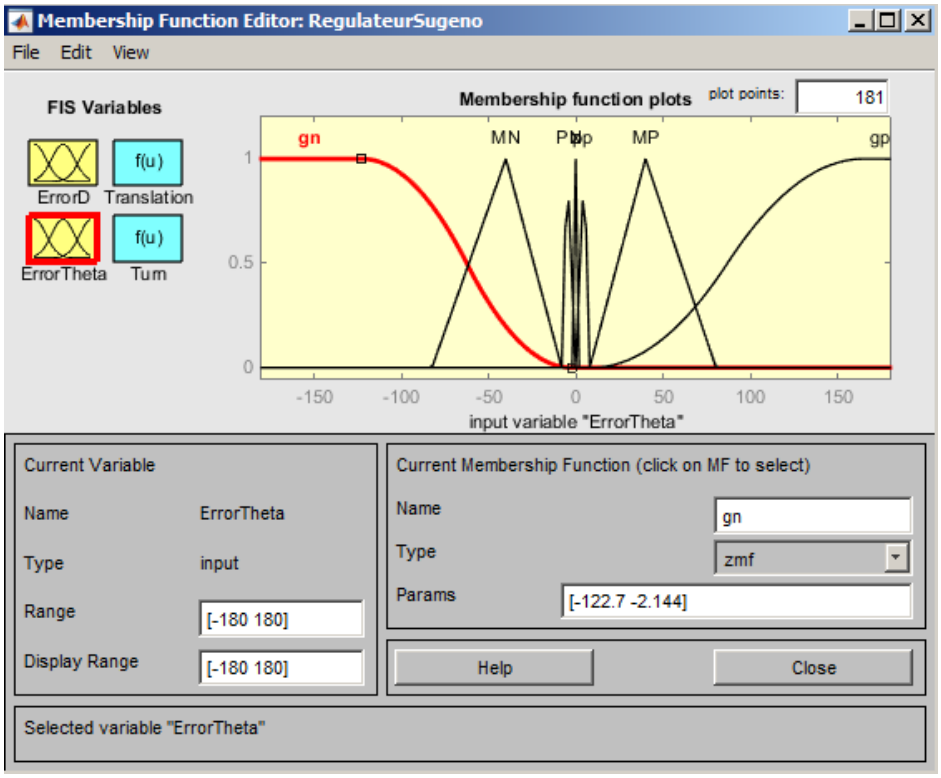


FIGURE 2.17: Erreur θ

Ici nous choisissons un régulateur floue du type "Sugeno", afin de ne pas trop charger la mémoire vive du système.

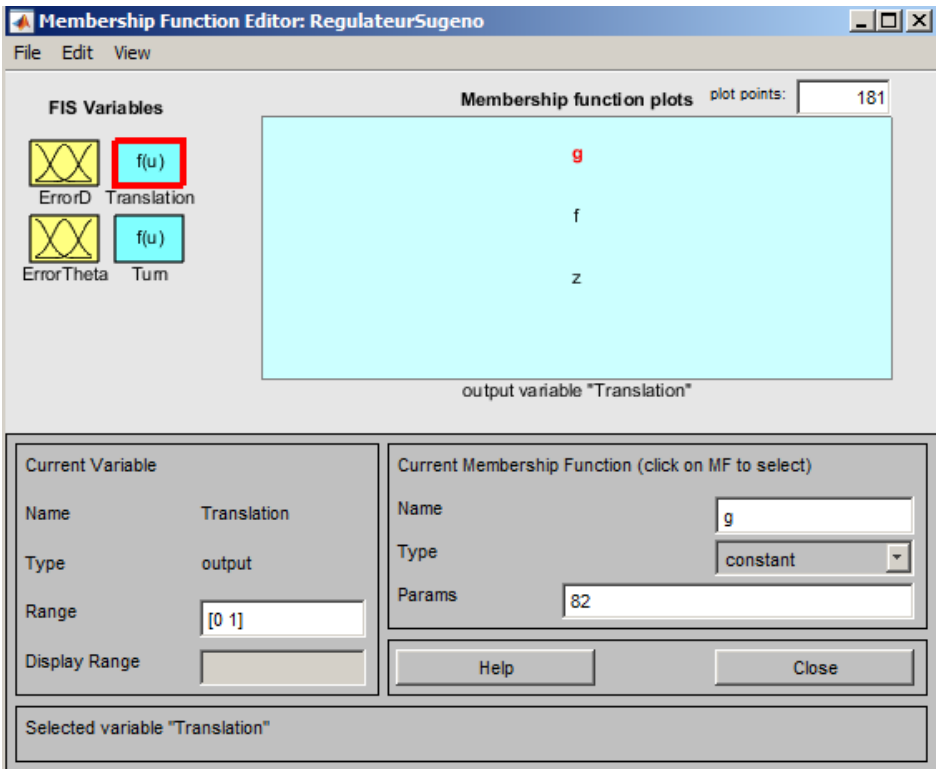


FIGURE 2.18: Vitesse Translation

Trois vitesses de translation :
Grande = 82 cm/s, Faible = 10 cm/s et Nulle = 0 cm/s.

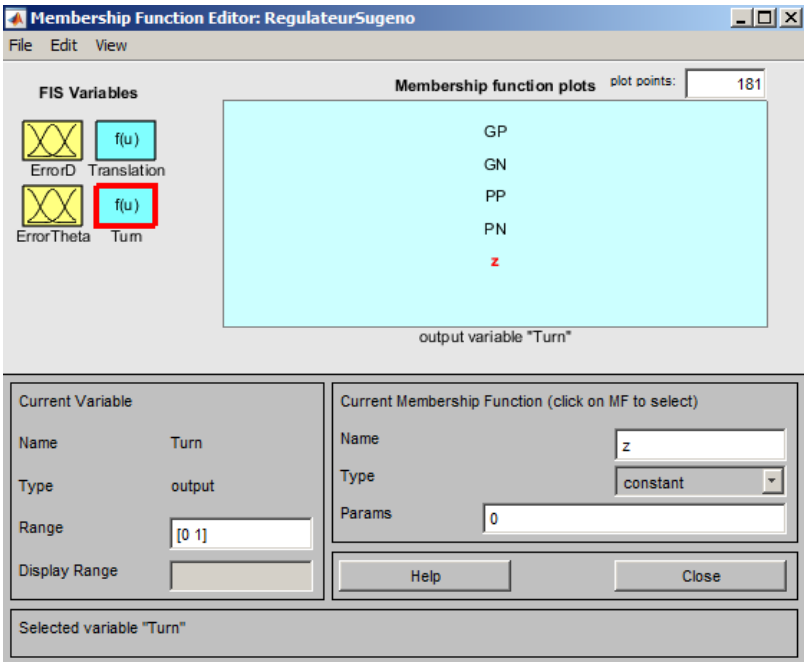


FIGURE 2.19: Vitesse de Rotation

Dans ce cas trois vitesses sont définies, Grande = 25 cm/s, Petite = 8 cm/s et nulle = 0 cm/s

2.2.5.2 Définition des règles

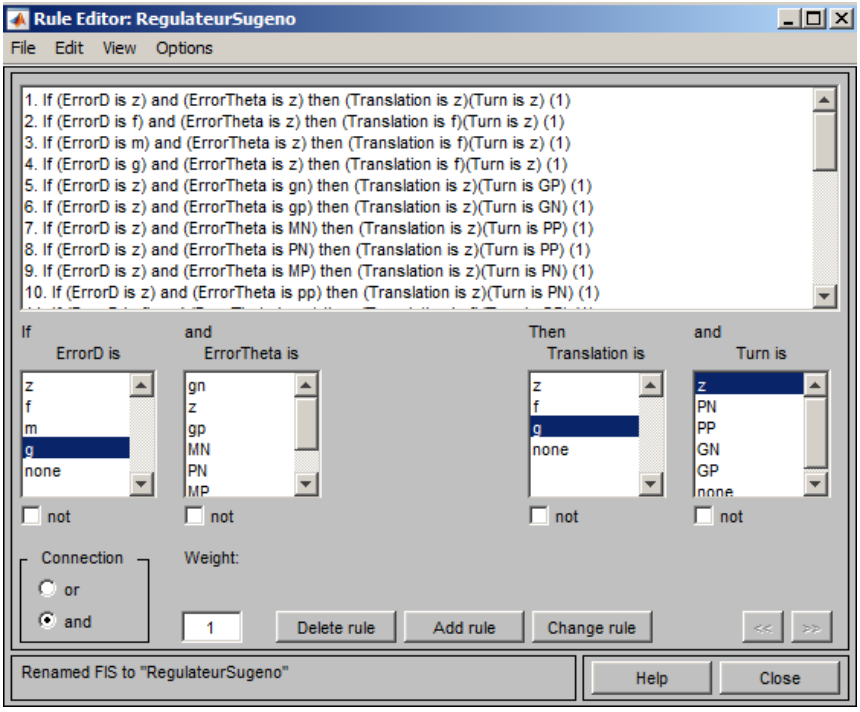


FIGURE 2.20: Règles

La dernière étape est celle de définition des règles un exemple pour illustrer la logique suivie : "Erreur Distance Grande et Erreur sur θ petite(positive/négative) Vitesse de translation grande et vitesse de rotation (négative/positive)."

2.2.5.3 Deffuzification et visualisation

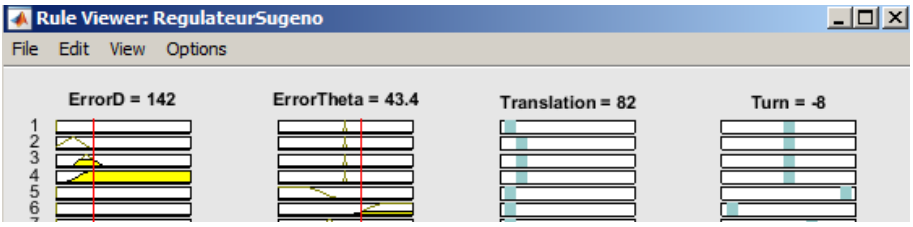


FIGURE 2.21: Visualisation

Cette option permet de parcourir les différentes entrées possibles, afin de voir le comportement du régulateur.

2.3 Résultats et commentaires

2.3.1 Test 1

Consignes : X=100 cm, Y=200 cm.

FIGURE 2.22: Test 1

Convergence en erreur statique respectée.

2.3.2 Test 2

Consignes : X=-150 cm, Y=200 cm.

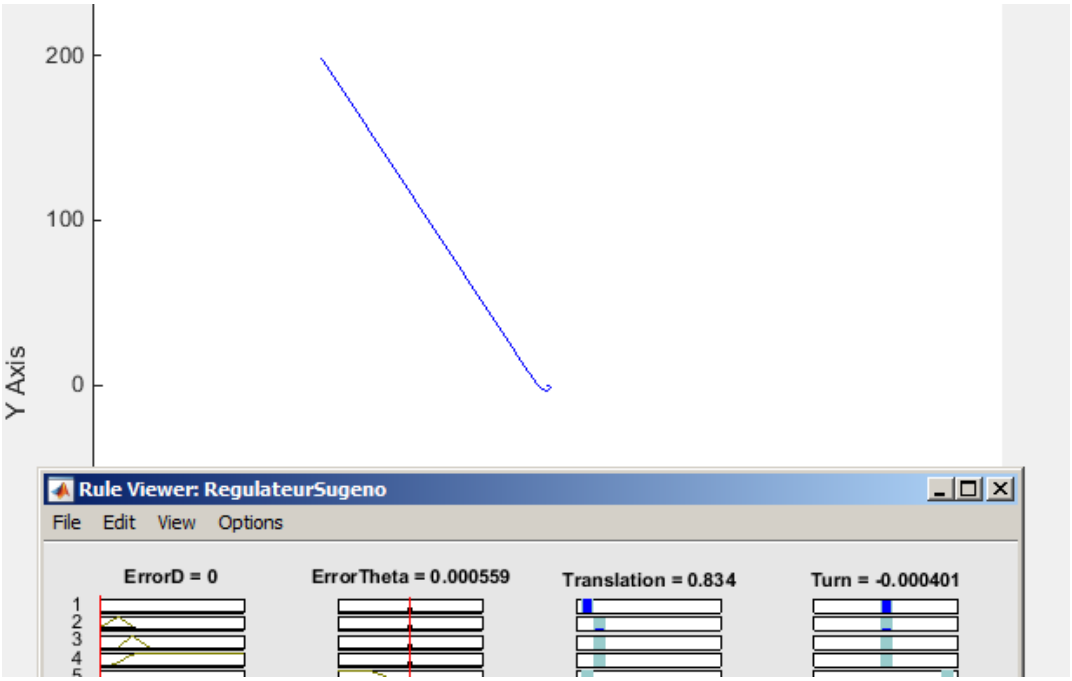


FIGURE 2.23: Test 2

2.3.3 Test 3

Consignes : x= -150 cm, Y=-200 cm.

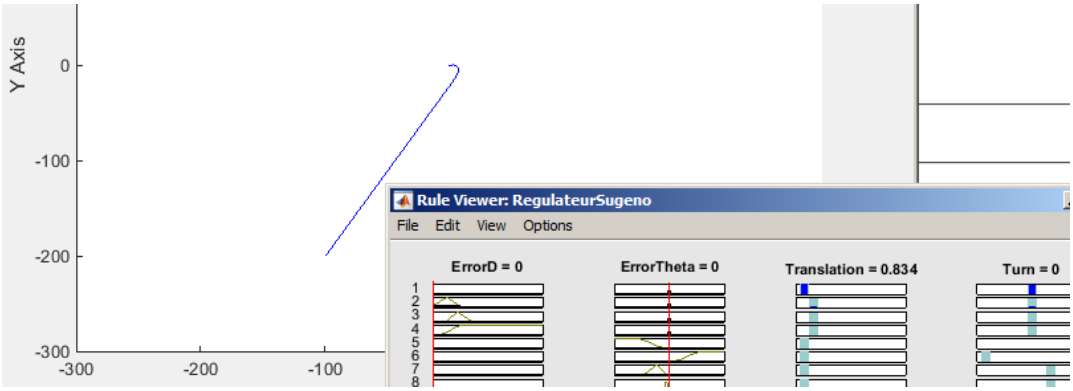


FIGURE 2.24: Test 3

Les réponses à un échelon sont satisfaisantes.

2.3.4 Test 4

On fait varier la position désirée pendant le mouvement pour voir si le régulateur suit la consigne.

Historique du mouvement sur (X, Y)

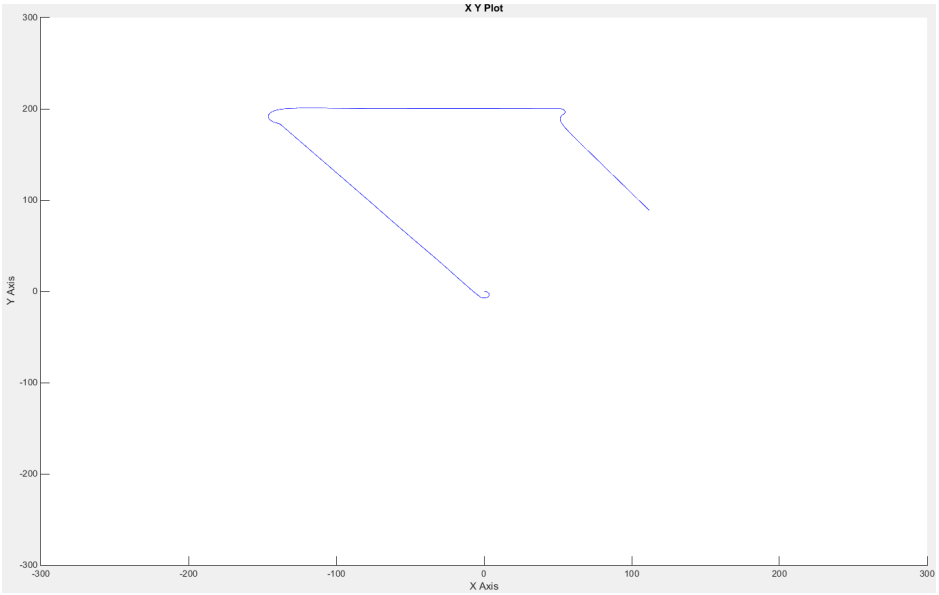


FIGURE 2.25: Test 4

Commentaire : la navigation floue converge bien, la consigne est bien respectée.

Chapitre 3

Implémentation

Ce chapitre sera tout d'abord consacré à la présentation du robot, de la structure mécanique et des cartes électroniques, puis on décrira le modèle pour l'implémentation enfin nous passerons aux tests après implémentation de l'algorithme.

3.1 Description du Robot

Dans cette section on présente brièvement le robot sur lequel on effectue les tests, celui ci à été réaliser par l'équipe "Cyborg" (Algérie) fondée en 2010, basée à Blida, participe aux compétitions "Eurobot" (3 fois champion d'algérie).

3.1.1 Structure Mécanique

Ci dessous les différentes vues du robot.

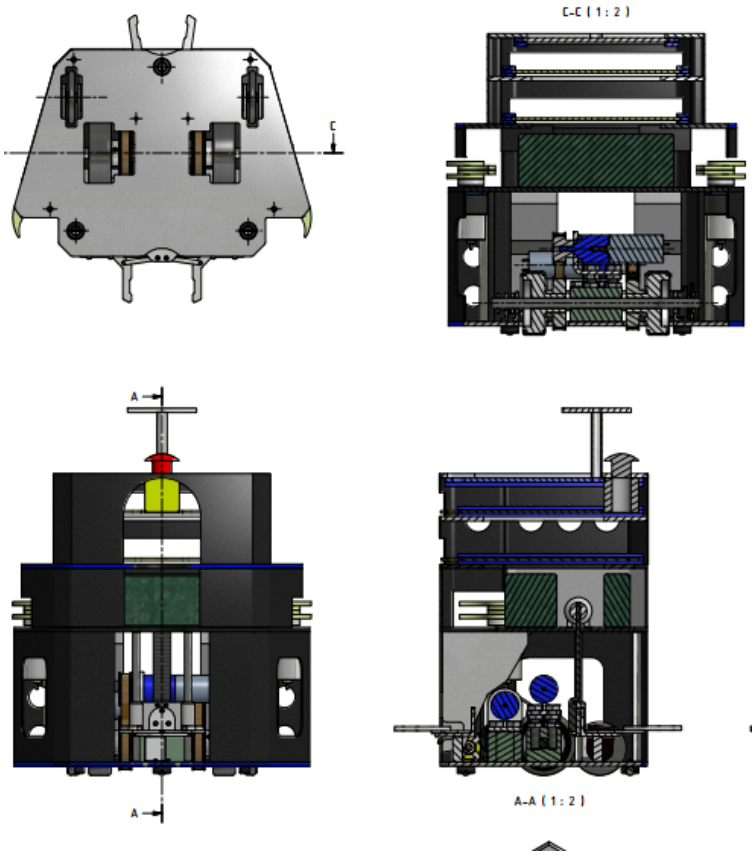


FIGURE 3.1: "Robucup" Cyborg 2015

La partie du robot qui nous intéresse est sa base, illustrée ci dessous celle ci se compose de deux roues (en Jaune) reliés a deux moteurs (en Rouge) via une courroie et un réducteur de vitesse.

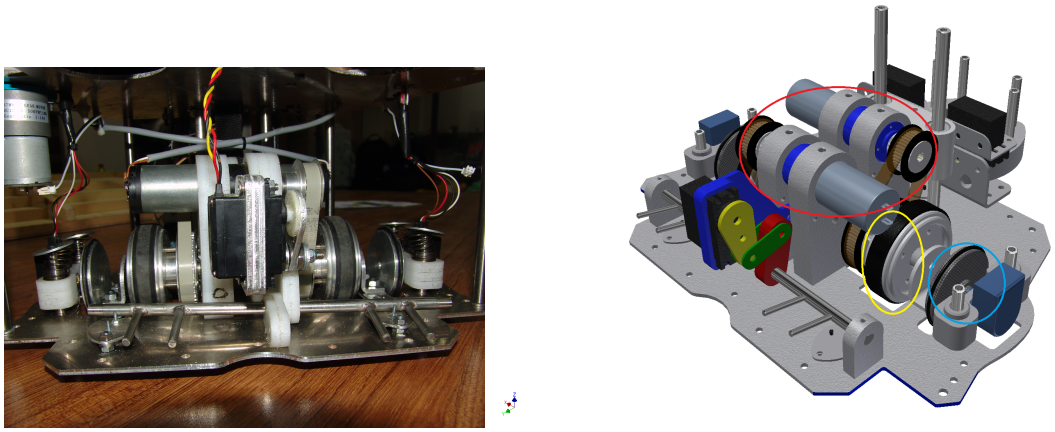


FIGURE 3.2: Base Roulante

La partie encodeur (en Bleu) est disjointe du moteur et se fait en externe pour palier au problèmes de glissements qui faussent les mesures (en effet dans le cas ou les encodeurs sont intégrés au moteurs les phénomènes de glissement du robot faussent l'odométrie).

Le robot inclus d'autres parties permettant d'effectuer les différentes taches et actions relatives au cahier de charges des concours "eurobot".

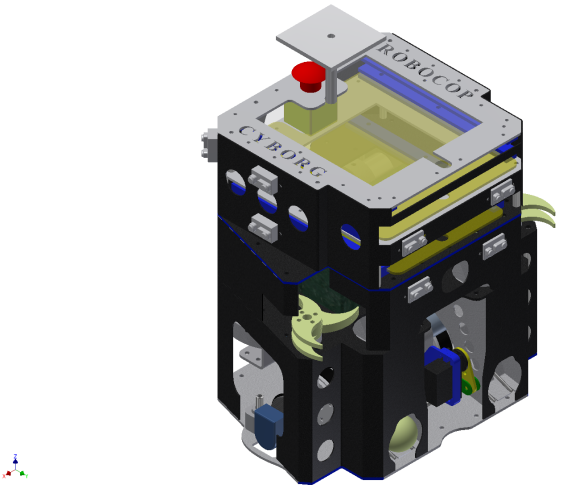


FIGURE 3.3: "Robucup" de l'équipe (Cyborg)

3.1.2 Partie électronique

L'électronique du robot est constituée de trois parties la première faisant la commande en tension des moteurs ou carte puissance, la partie asservissement s'occupe de la régulation de la consigne de tension appliquée en entrée de la carte puissance ces deux cartes sont isolée

électriquement enfin la carte navigation s'occupe du calcul de la consigne de vitesse pour atteindre la position désirée.

3.1.2.1 Moteurs

Deux moteurs à courant continu sont utilisés pour faire avancer le robot,

FIGURE 3.4: Maxon DC Motor 226771

3.1.2.2 Encodeurs

La plupart des encodeurs pour robots mobiles utilisent des capteurs optiques (mais il existe des encodeurs utilisant une information mécanique ou magnétique). L'idée est de placer un disque alternant des zones transparentes et opaques devant un capteur de lumière et de rentre le disque solidaire de l'axe de rotation de la roue. La fréquence d'apparition des zones blanches et noires (ou de tout autre principe offrant un contraste suffisant) devant le capteur de lumière va indiquer la vitesse de rotation. Le schéma suivant présente le principe de fonctionnement basique de l'encodeur.

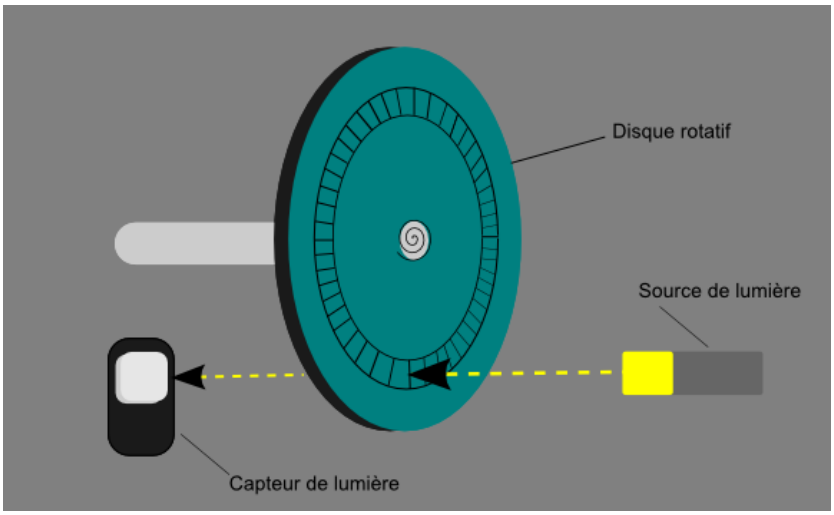


FIGURE 3.5: Encodeur principe

Lorsque le disque tourne, les segments opaques bloquent la lumière alors que les segments transparents la laissent passer. Ceci génère des impulsions d'onde carrée qui peuvent ensuite être interprétées comme position ou mouvement.

Si le fonctionnement précédent indique la vitesse de rotation, il n'indique pas le sens de rotation.

Ce problème est résolu par l'encodeur en quadrature (l'encodeur en quadrature le nom que l'on donne à l'encodeur rotatif incrémental). L'encodeur en quadrature comporte deux

pistes de code dont les secteurs sont décalés de 90 degrés d'une piste à l'autre. Ces deux pistes génèrent deux signaux de sortie. Si le premier signal devance le second alors le disque tourne dans le sens des aiguilles d'une montre et dans l'autre sens dans le cas contraire. Par conséquent, en mesurant à la fois le nombre d'impulsions et les phases relatives des deux signaux on peut mesurer la position et la direction de la rotation des roues du robot.

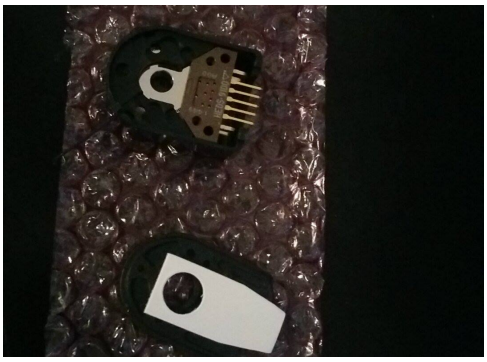


FIGURE 3.6: Encodeur Avagor

Ci dessus le type d'encodeurs utilisé.

3.1.2.3 Carte Puissance

La carte de puissance à été conçue au tour du circuit $L298$ permettant le contrôle des deux moteurs à courant continu, celle ci utilise des optocoupleurs 4n35/4n36 pour séparer totalement l'alimentation logique (Commande) de l'alimentation puissance.

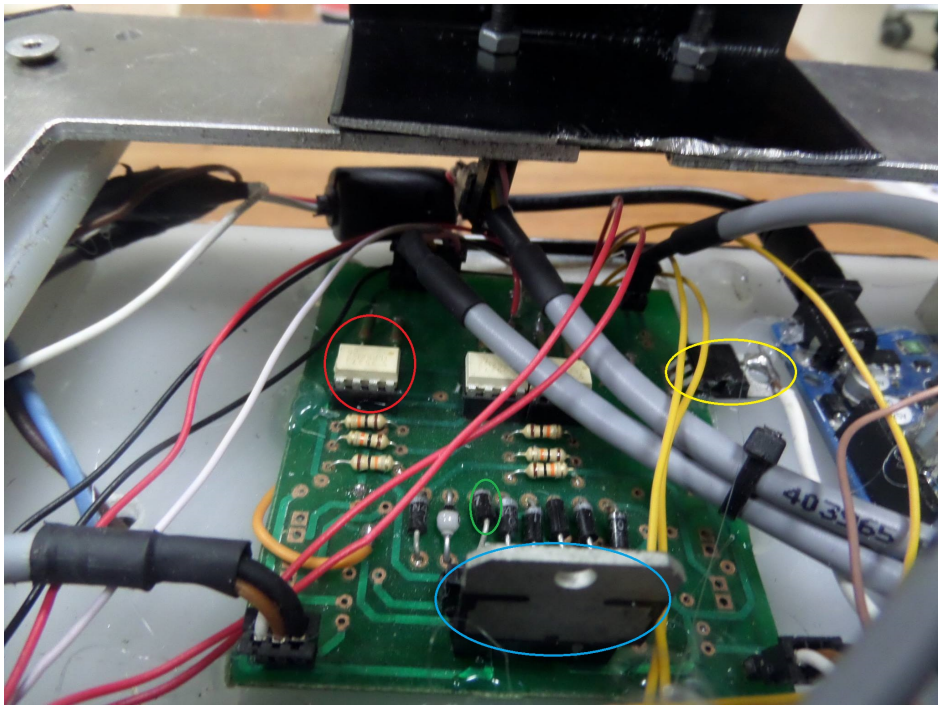


FIGURE 3.7: Carte Puissance

— Rouge : Optocoupleur

— Bleu : CI L298
— Vert : Diode de roue libre
— Jaune : Regulateur 5V

3.1.2.4 Carte asservissement

La carte asservissement s'occupe de la régulation de la vitesse de rotation des deux moteurs, afin de garder une vitesse identique ou presque à la consigne, Le logiciel embarqué se base sur la librairie C++ Arduino de Robotique Mobile 4.8 open source développée par Cyborg.

Type de carte :

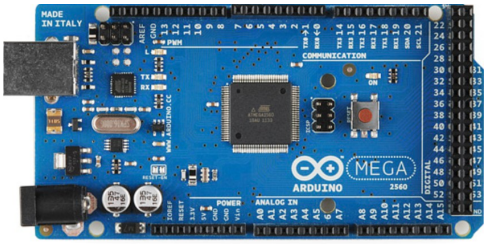


FIGURE 3.8: Carte Arduino Mega

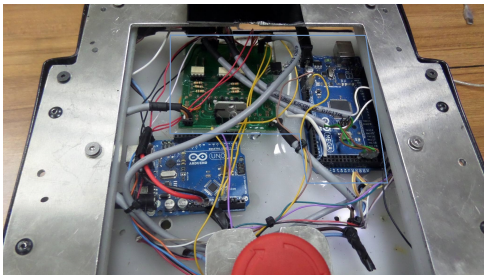


FIGURE 3.9: Cartes Puissance et Asservissement

Deux types de fonctionnement ont été envisagés :
— le premier dit en "offline" la consigne de vitesse est incluse dans le programme C++, cela pour des fins de tests de validation du fonctionnement.
— le deuxième type est le mode "online" la consigne de vitesse est récupérée depuis la carte "Navigation" via un protocole utilisant la liaison série de l'Arduino Mega.

On définit les interfaces :
— Entrées : Les données prises des encodeurs pour la mesure des vitesses de rotation, deux entrées GPIO intéruptibles pour la vitesse, et deux GPIO pour le sens de rotation.
  Les consignes de vitesse viennent de la carte Navigation, on utilise pour cela deux Ports Série l'un pour la vitesse droite l'autre pour celle de gauche pour argumenter la résolution, tandis que le sens de rotation est transmis sur les GPIO.

— Sorties : On utilise les PWMs pour la commande des ponts en H sur la carte puissance afin de contrôler les deux moteurs, deux GPIO sont utilisés pour indiquer le sens de rotation.

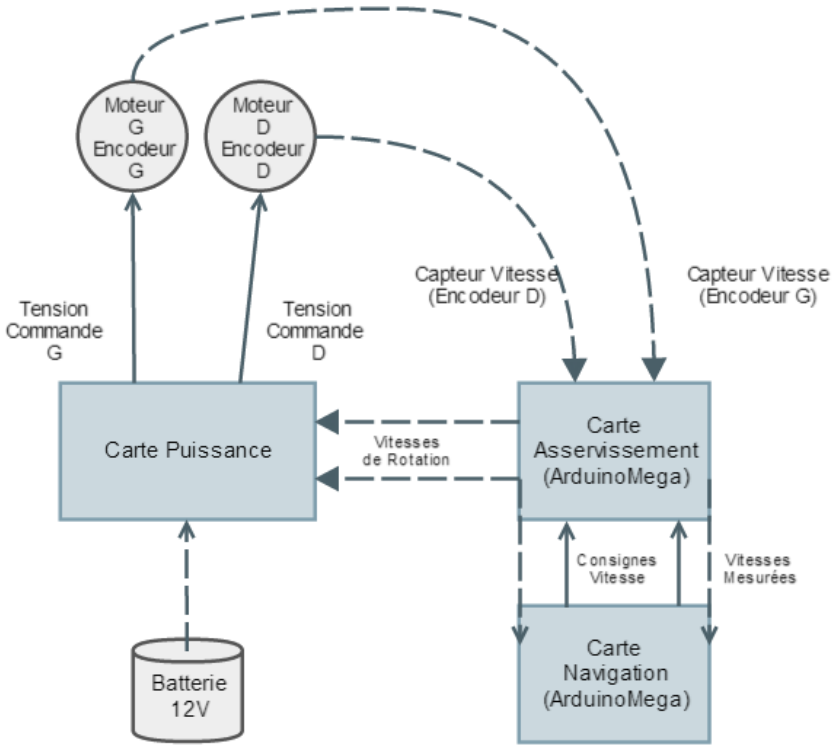


FIGURE 3.10: Schéma liaisons

3.1.3 Carte Navigation

La carte navigation s'occupe du calcul des nouvelles consignes de vitesses grâce à l'algorithme basé sur la commande floue, la vérification et le chargement du programme s'effectue à partir de Simulink.

On définit les Interfaces :

— Entrées : Les Ports Série sont utilisés pour la communication, (récupération de la vitesse mesurée par la carte Asservissement).

— Sorties : Transmission de la valeur absolue des consignes de vitesse sur les ports Série. Deux GPIO pour indiquer le sens.

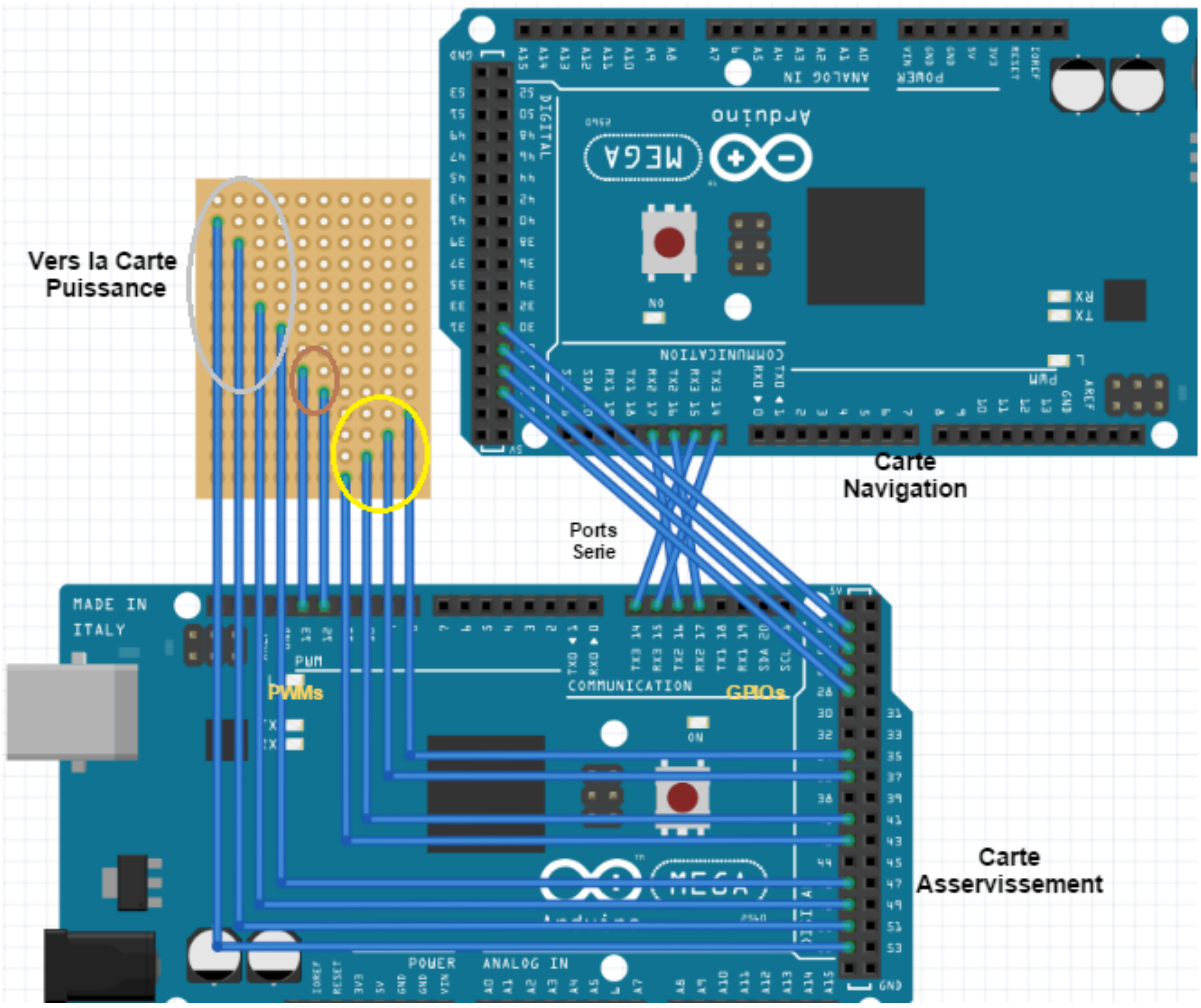


FIGURE 3.11: Câblage

Le schéma de câblage résume les différentes interfaces :

— En gris : Récupération du signal des encodeurs (2x1 pour chaque vitesse et 2x1 pour chaque sens).
— En marron : les deux signaux PWM.
— En doré : sorties numériques pour indiquer donner le sens de rotation.
— le branchement entre les deux cartes (Navigation-Asservissement) montrent les interfaces séries (transmission des valeurs absolues des vitesses consigne et mesurée, les quatre connexion GPIO pour le signe en Entrée/Sortie à droite et à gauche).

3.2 Élaboration du modèle et méthodologie de tests

La différence notable entre le modèle de simulation et celui pour l'implémentation est le bloc "Robot" en effet dans ce qui suit il est remplacé par la partie communication avec la carte Asservissement.

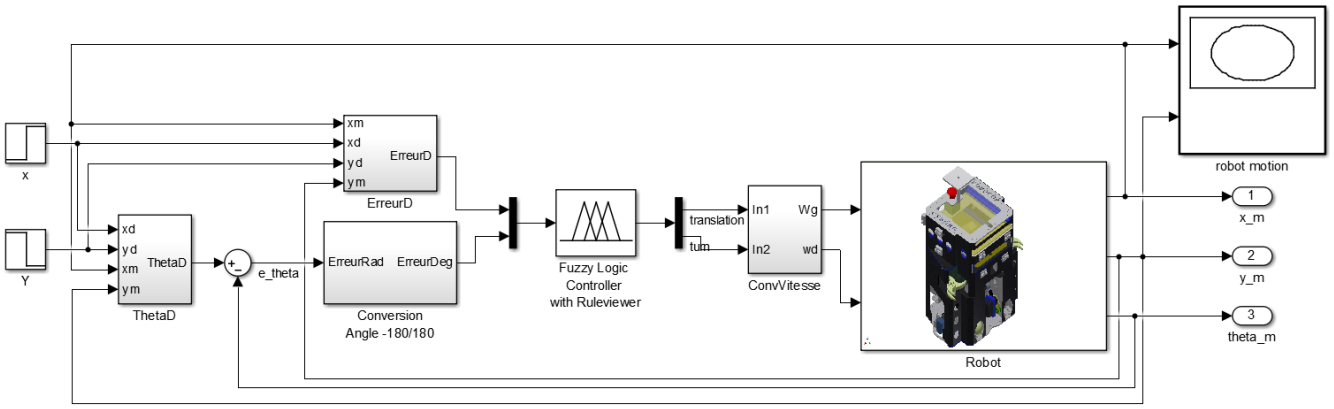


FIGURE 3.12: Modèle navigation

3.2.1 Programme et protocole de communication

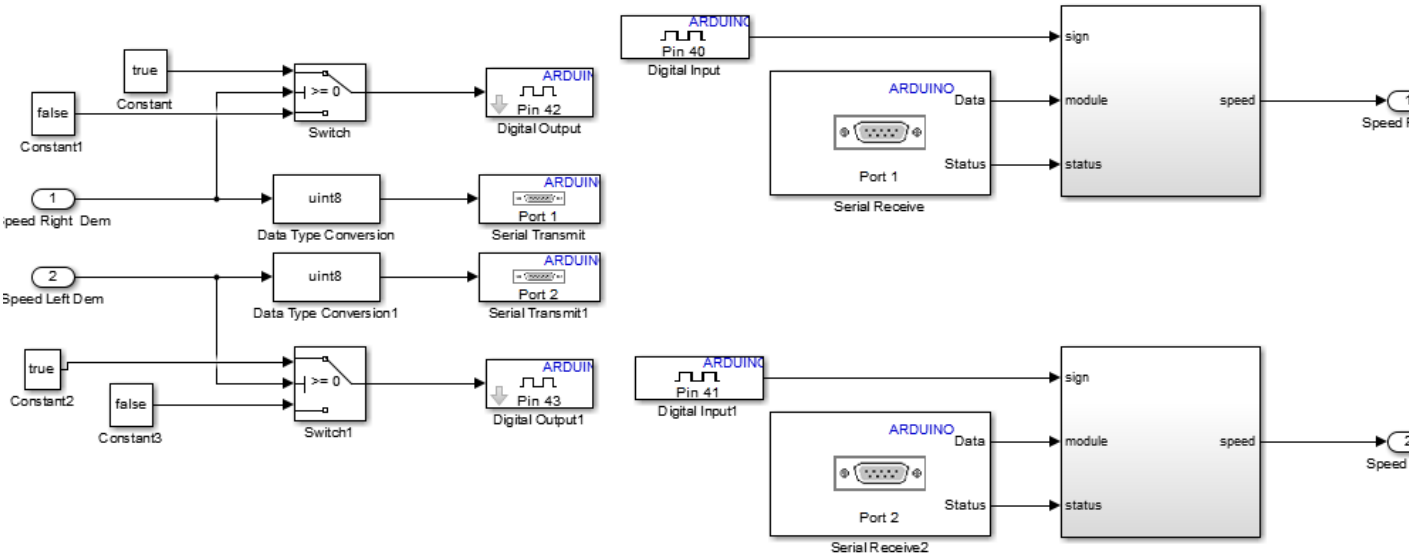


FIGURE 3.13: Partie Communication

— On envoi la consigne de vitesse et on récupère la mesure, pour calculer le déplacement afin de déterminer l'erreur à introduire au régulateur.

— Fonctionnement du protocole de communication : La valeur absolue de la vitesse passe par les ports série, et le signe de chaque vitesse (Vitesse gauche, droite en émission et réception) est transmit par les entrées numériques (GPIO), l'intérêt d'une telle technique est l'augmentation de la résolution.

— Remarque : La donnée sur la vitesse envoyée est effectuer en un rapport de 255 (valeur max à envoyer), pour avoir une résolution optimal on normalise la vitesse par un rapport et la vitesse devient : V' = $V.255/V_{max}$.

3.3 Résultats des tests

3.3.1 Déroulement des tests

La méthode est décrite pour l'organisation des tests ceci est primordial lorsqu'il sagit de la validation du fonctionnement.

La majeur partie des tests s'effectuent sous Simulink via le mode (HIL "Hardware-In-The-Loop") afin de pouvoir visualiser les signaux et les différents paramètres de test.

Le code est deployé sur la cible (Carte "Navigation" Arduino Mega), un protocole est établi entre la station de travail et la carte électronique pour l'acheminement des signaux de test en temps-réel.

3.3.2 Test N 1 : Asservissement "Offline"

La vérification du bon fonctionnement de la carte asservissement. (remarque c'est le seul test qui s'effectue sans le mode "HIL").
On introduit différentes consignes de vitesse :

— 1/ Vg=Vd=5,10,20,30, Résultats attendu : Le robot va a une vitesse plus élevée à chaque augmentation aussi la vitesse ne doit pas varier pendant le test.

— 2/ Vg=-Vd = 5,10,20,30 Résultat attendu : le robot tournes bien sur lui même (à des vitesses plus élevés pour chaque argumentation).

Résultat : Le test s'est bien déroulé, le comportement du robot à été bon (allure du mouvement, vitesse uniforme).

3.3.3 Test N 2 : Communication et asservissement "Online"

— Introduction : Test de la communication, vérification des vitesses mesurées.
— Le test s'effectue en mode "HIL" la visualisation des signaux est indispensable
— 1/ Affectation des vitesses Gauche et droite
— 2/ Observation des vitesses mesurés.
On visualise par exemple la vitesse mesurée lorsqu'on donne une consigne depuis la carte navigation vers la carte asservissement des moteur, V_g= - V_d = -5 cm/s.

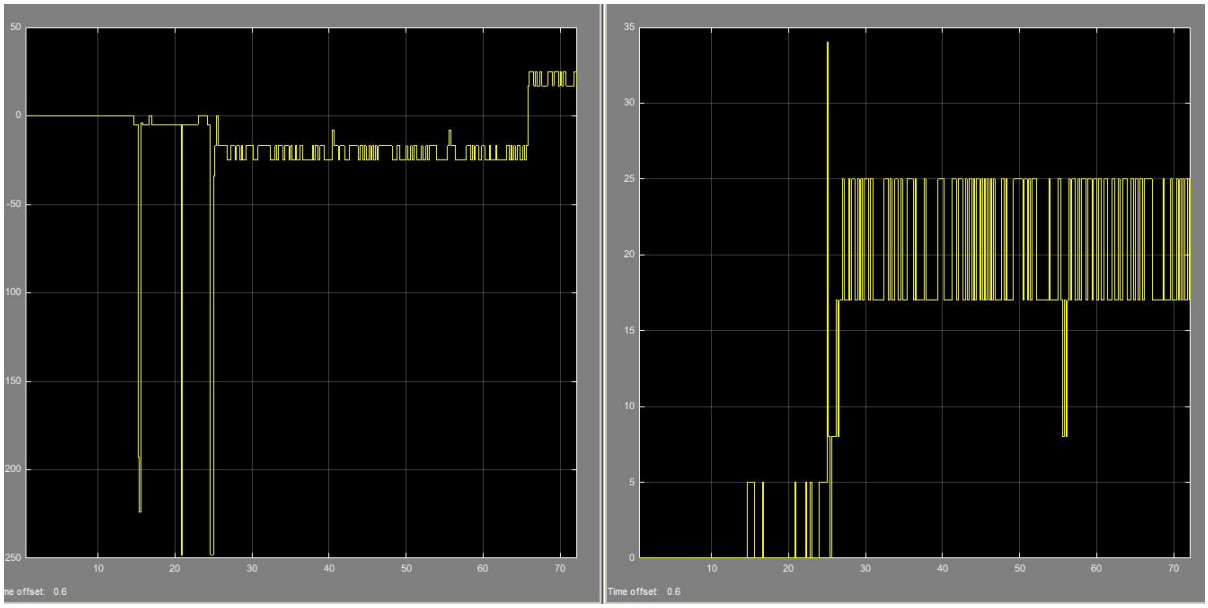


FIGURE 3.14: Pour une consigne - 5 cm/s à V Gauche et 5 cm/s à Droite

On dé-normalise sachant que Vmax a été définit coté asservissement = 60 cm/s.
$V_{mesurée} = (22,5\text{x}255)/60 = 5{,}29$ cm/s.

Le test s'est bien déroulé, on visualise la bonne vitesse en sortie, la communication (carte asservissement<->navigation) est donc validée.

3.3.4 Test N 3 : L'odométrie

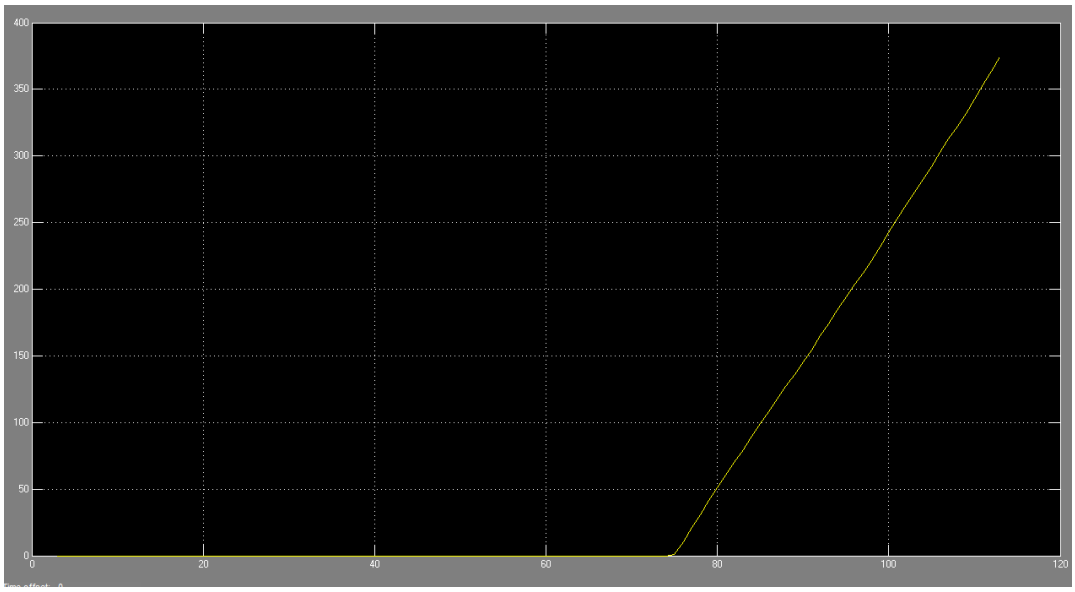


FIGURE 3.15: Évolution de x_m en fonction du temps

Pour une consigne $Vg = Vd = 10cm/s$, on suit l'évolution au cours du temps des coordonnées, un exemple illustré ci dessus la

3.3.5 Test N 4 : Tests du régulateur

Les tests du régulateur floue se font en continue le paramétrage est une procédure longue et sensible à l'environnement des tests (table du robot, frottements, glissements etc).

3.3.5.1 Rotation seule

Donner une consigne $\theta_d = 180, 90, 45, 30, 15$ au régulateur.

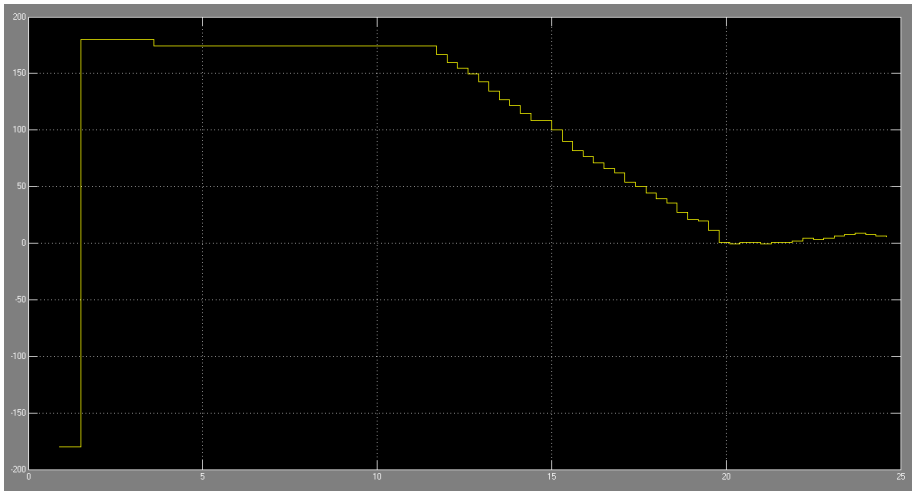


FIGURE 3.16: Erreur pour une consigne -180^0

On voit l'évolution de l'erreur, le régulateur converge bien.

3.3.5.2 Distance Seule

Le régulateur sur la distance évalue si la distance parcourue égale à la consigne.

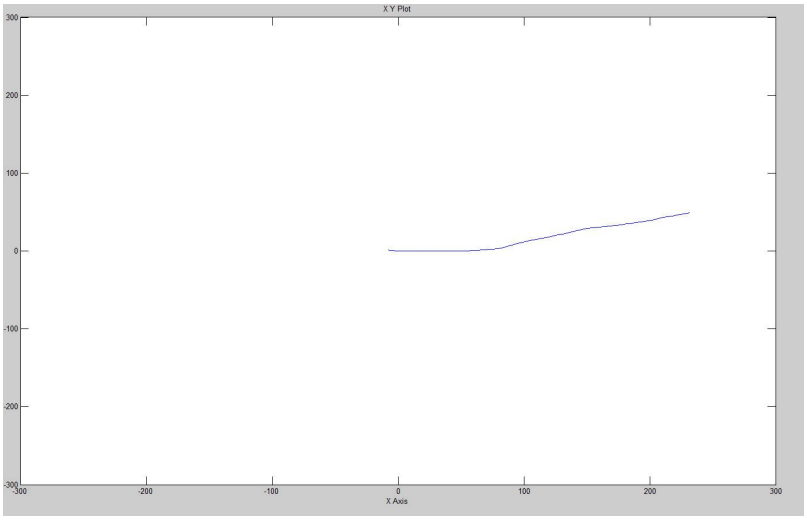


FIGURE 3.17: Translation

Les performances du régulateur doivent encore être améliorées d'autres séries de tests doivent être effectués afin d'aboutir à un régulateur au paramètres adéquats.

Conclusion générale

Dans ce mémoire, on a définit puis développé un algorithme de navigation, s'en est suivit son implémentation sur robot mobile.

La démarche suivit à permis la validation et vérification en continue du fonctionnement de la navigation, les tests sur robot on donnés des résultats concluants il reste néanmoins clair qu'il est possible d'améliorer les performances du régulateur.

Enfin, une des perspectives de ce travail serait d'ajouter au fonctionnalités une partie évitement d'obstacles.

Bibliographie

[1] Cherroun, Lakmissi. Navigation Autonome d'un Robot Mobile par des Techniques Neuro-Floues Diss. Université de Biskra , 2014.

[2] Jean-Daniel Dessimoz Commande à algorithme génétique

[3] Cheribet Mohamed, Laskri Mohamed Tayeb, Evitement d'obstacles dynamiques par un robot mobile.

[4] Benmakhlouf Abdesslam "Controleur flou pour robot d'intérieur"

[5] N. Ouadah, Implementation d'un Contrôleur Flou pour la navigation d'un Robot Mobile de type voiture.

[6] Frédéric B., février 2008, Méthodes d'accostage et Logique Flou Fuzzy control for docking operation, http ://www.pobot.org/L-accostage-c-est-Flou.html.

[7] http ://www.polytech-lille.fr/cours-regulation-automatique/pr545.htm

[8] Fuzzy Logic Based Navigation of Mobile Robots, Amur S. Al Yahmedi and Muhammed A. Fatmi Sultan Qaboos University, Oman.