REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE
MINISTÈRE DE L'ENSEIGNEMENT SUPÉRIEUR ET DE LA RECHERCHE SCIENTIFIQUE

École Nationale Polytechnique

المدرسة الوطنية المتعددة التقنيات
Ecole Nationale Polytechnique

Laboratoire de Commande des Processus

Département d'Automatique

Laboratoire de Commande des Processus

Mémoire de projet de fin d'étude

En vue de l'obtention du diplôme

d'Ingénieur d'État en Automatique

Commande Optimale d'un Robot Balanceur à Deux Roues: Etude et Implémentation

Kalb Eddine CHELLI

Sous la direction de : M. M.CHAKIR Dr

M. M.TADJINE Pr

Présenté et soutenue publiquement le 19/06/2017

Composition du Jury :

Président	M.	B. HEMICI	Professeur	ENP
Promoteur	M.	M. CHAKIR	MCB	ENP
Promoteur	M.	M. TADJINE	Professeur	ENP
Examinateur	M.	H. ACHOUR	MAA	ENP

ENP 2017

REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE
MINISTÈRE DE L'ENSEIGNEMENT SUPÉRIEUR ET DE LA RECHERCHE SCIENTIFIQUE

École Nationale Polytechnique

المدرسة الوطنية المتعددة التقنيات
Ecole Nationale Polytechnique

Laboratoire de Commande des Processus

Département d'Automatique

Laboratoire de Commande des Processus

Mémoire de projet de fin d'étude

En vue de l'obtention du diplôme

d'Ingénieur d'État en Automatique

Commande Optimale d'un Robot Balanceur à Deux Roues: Etude et Implémentation

Kalb Eddine CHELLI

Sous la direction de : M. M.CHAKIR Dr

M. M.TADJINE Pr

Présenté et soutenue publiquement le 19/06/2017

Composition du Jury :

Président	M.	B. HEMICI	Professeur	ENP
Promoteur	M.	M. CHAKIR	MCB	ENP
Promoteur	M.	M. TADJINE	Professeur	ENP
Examinateur	M.	H. ACHOUR	MAA	ENP

ENP 2017

Dédicace

A mes parents

à mes frères, ma sœur

à toute ma famille

à tous mes amis

je dédie ce mémoire.

REMERCIEMENTS

Avant tout, je remercie ALLAH, le tout puissant, pour m'avoir assisté et armé de patience afin d'accomplir ce modeste travail.

J'exprime ma profonde gratitude, mon grand respect et ma sincère reconnaissance à mes promoteurs Monsieur M.Chakir et Monsieur M.Tadjine pour avoir assumé la lourde responsabilité de m'encadrer, de m'avoir orienté et conseiller tout au long de ce travail ainsi pour la confiance qu'ils m'ont accordé.

Je remercie chaleureusement les membres du jury pour l'honneur qu'ils m'ont fait en acceptent d'évaluer mon projet.

Je souhaite aussi remercier mes professeurs d'Automatique et tous mes enseignants pour les connaissances qu'ils m'ont transmis, leur disponibilité et leurs efforts.

Que tous ceux qui ont contribué de près ou de loin à la réalisation de ce modeste travail trouvent ici l'expression de ma sincère gratitude.

ملخص:

يهدف هذا المشروع الى دراسة الروبوت المتوازن على عجلتين و التحكم فيه عن طريق المتحكم الأمثل بغية تحقيق توازن الروبوت حول النقطة الغير مستقرة (الوقوف على العجلتين) ثم التحكم في حركته.

أتبعت الدراسة النظرية بتطبيق عملي يتمثل في انجاز الروبوت المتوازن على عجلتين، والتحكم فيه بواسطة المتحكم الأمثل التربيعي الخطي ومتحكم النمط الانزلاقي.

تم الحصول على نتائج جد مرضية، حيث تمكن الروبوت من الوقوف والتوازن في وضع غير مستقر، كما تمكن من تتبع تعليمات الموقع مع وقت استجابة مقبول. النتائج المحصل عليها من المتحكم التربيعي الخطي كانت أفضل من النمط الانزلاقي بحيث من خلال هذا الأخير لم يتمكن الروبوت من الاستجابة الى تعليمات الموقع بشكل جيد.

الكلمات المفتاحية: الروبوت المتوازن، النواس المقلوب، المتحطم التربيعي الخطي، متحكم النمط الانزلاقي، أر دوينوا، مقياس التسارع، الجيرو سكوب، المرشح المكمل.

Abstract:

The purpose of this project is to study the two-wheeled balancing robot and control it by the optimal controller for balancing in the unstable point (the upright position) then controlling its movement.

The theoretical study of robot is followed by an experimental realization of a two-wheeled balancing robot. And control it by the optimal linear quadratic regulator and the sliding mode control.

The results obtained are good, such that the robot becomes able to balance on both wheels in the vertical position, and also able to follow the position setpoint with a tolerable response time. The results show that the LQR command is more efficient than the sliding mode command, the position following is still slow and sometimes diverge with the sliding mode controller.

Keywords: balancing robot, inverted pendulum, LQR control, sliding mode control, Arduino, accelerometer, gyroscope, complementary filter.

Résumé :

Le but de ce projet est l'étude du robot balanceur à deux roues, et appliquer la commande optimale pour l'équilibrer dans le point instable (la position vertical), puis commander son mouvement.

L'étude théorique du robot est suivie d'une réalisation expérimentale d'un robot balanceur à deux roues, et implémenter la commande optimale linéaire quadratique et la commande par mode glissant.

Les résultats obtenus sont bons, tel que le robot devient capable de équilibrer sur les deux roues dans la position verticale, et aussi capable de poursuivre les consignes de position avec un temps de réponse tolérable. Les résultats montre que la commande LQR plus performante que la commande par mode glissant, la poursuite de consigne de position par ce dernier est lentes et parfois diverge.

Mots clés : robot balanceur, pendule inversé, commande LQR, commande par mode glissant, Arduino, accéléromètre, gyroscope, filtre complémentaire.

Table des matières

Table des figures

Introduction générale

La robotique regroupe un domaine très vaste d'activités de la vie humaine, en effet, plus de 1 235 600 robots sont à l'œuvre aujourd'hui. Ils ont envahi notre univers, ils sont utilisés dans tous les domaines : industriel, militaire, médical et celui des services. Les robots actuels ne sont qu'au début de leur évolution, ainsi des laboratoires dans le monde entier travaillent pour les améliorer et les rendre plus efficaces et plus intelligents. Certains travaillent sur la marche, d'autres sur la connaissance faciale ou vocale, tandis que d'autres étudient encore des algorithmes d'intelligence artificielle de plus en plus élaborés.

La particularité du système à pendule inversé a suscité l'intérêt de nombreuses recherches en raison de la nature instable du système. Les études sur les pendules inversés à deux roues ou communément « balancing robot » a gagné en popularité au cours de la dernière décennie dans un certain laboratoire de robotique à travers le monde. Dans ce contexte, notre projet consiste à faire une étude sur la commande et la réalisation du robot balanceur à deux roues capable de se stabiliser sur les deux roues en position verticale et capable aussi de faire des mouvements.

Dans le premier chapitre, on présente le robot balanceur à deux roues et leur principe de fonctionnement. On procède par la suite à la modélisation du robot dans l'espace d'état en se basant sur ses équations dynamiques.

Dans le deuxième chapitre, on présente la description détaillée de la structure mécanique et la structure électrique du robot balanceur, et on présente aussi les étapes de réalisation pratique de ce robot et l'utilisation de filtre complémentaire pour améliorer la précision de mesure de l'angle d'inclinaison du robot.

Dans le troisième chapitre, on fait la synthèse de trois lois de commande, la commande optimale LQR, la commande PID et la commande par mode glissant. Puis, on fait la simulation de ces commandes sur le système du robot.

Dans le quatrième chapitre, on présent les résultats pratiques. On implémente la commande optimal LQR et la commande par mode glissant puis on test la performance de ces commandes. On termine ce chapitre par les résultats de la commande du robot en temps réel.

Chapitre 1

Modélisation

Chapitre 1

Modélisation

Dans ce chapitre, on va tout d'abord présenter le robot balanceur à deux roues, son principe de fonctionnement et ses avantages. Ensuite, on développe son modèle dynamique qui sera présent sous forme d'équations différentielles. Puis, on présente ce système dans l'espace d'état. Finalement, on fait une présentation des résultats des simulations effectués en boucle ouverte sur le modèle non linéaire et le modèle linéaire, afin de mettre en évidence le comportement dynamique et son instabilité.

1.1 Présentation du robot balanceur à deux roues

La conception du robot balanceur à deux roues ou le robot "TWIP" (Two Wheeled Inverted Pendulum) est basée sur l'idée du pendule inversé, le pendule inversé est un système non linéaire et instable en boucle ouverte. Le problème du pendule inversée est très répondu dans le domaine de l'automatique. Une large application de la technologie dérivée de ce système instable a attiré l'intérêt de nombreux chercheurs et passionnés de la robotique à travers le monde. Dans les années passées, les recherches ont appliqué l'idée d'un modèle de pendule inversé mobile aux différents problèmes comme la conception de la marche pour les robots humanoïdes, fauteuils roulants automatique et les systèmes de transport personnelle. Par conséquent , il représente une expérience idéal pour l'application de techniques de commande classiques et contemporaines.

Le robot a un corps avec deux roues pour le mouvement dans le plan. Deux roues motrices indépendantes sont utilisées pour équilibrer le robot et la commande de positon.

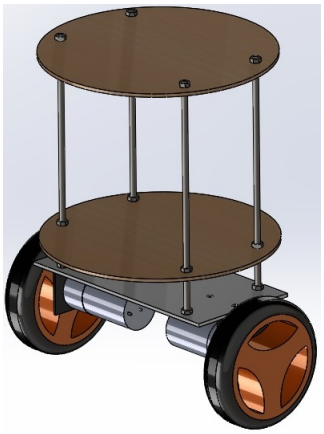

Figure 1.1 – Robot balanceur

1.2 Principe de fonctionnement du robot balanceur

Un robot balanceur à deux roues est un système d'une dynamique instable. Cela signifie que le robot est libre de tomber en avant ou en arrière sans aucunes forces appliquées. Il est équilibré

lorsque son centre de gravité et l'axe des roues sont situés sur une ligne verticale. Sinon, les roues devraient suivre les chutes du robot pour l'équilibrer.

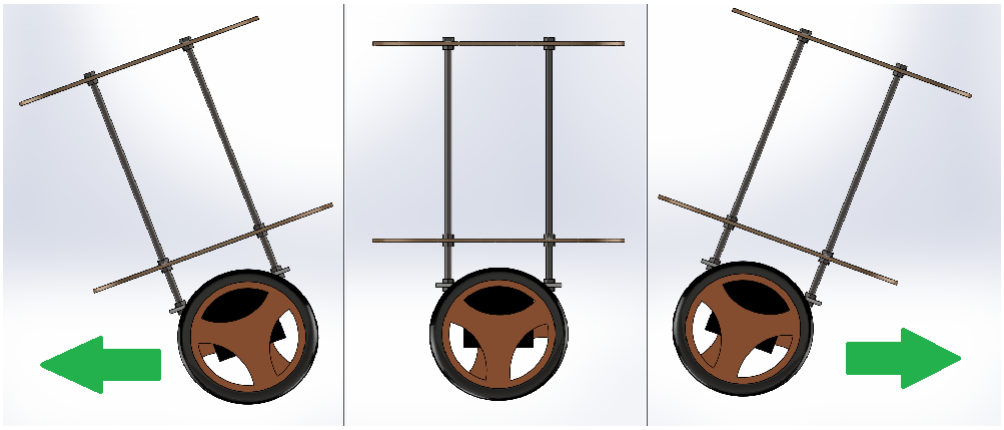

FIGURE 1.2 – Principe de fonctionnement

Le robot a un mouvement d'entraînement différentiel, quand le moteur droit tourne en avant et le moteur gauche tourne en arrière, le corps du robot va tourner à gauche, si on applique le même couples sur les deux moteurs, le robot va tourner autour de soi-même sans déplacement. Le même principe s'applique si les deux directions de moteurs sont renversées. Si le moteur droit tourne en arrière et le moteur gauche tourne en avant, le corps de robot va tourner à droite. Si les deux moteurs tourne en avant, le robot va déplacer ver l'avant et vice versa.

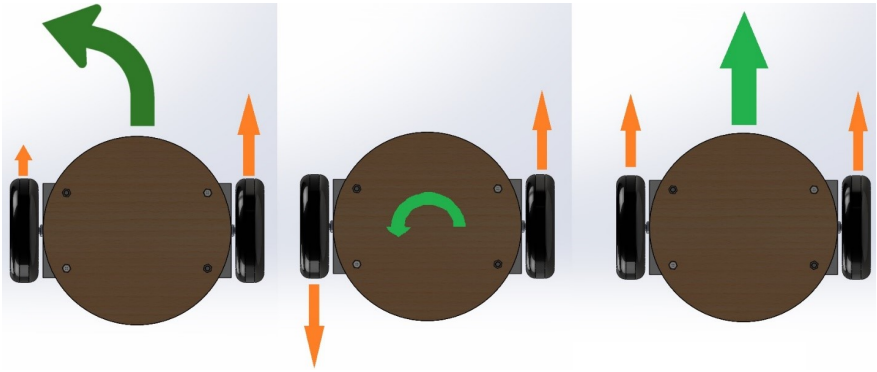

FIGURE 1.3 – Types de mouvement du robot

1.3 Existence du robot balanceur

Il y a beaucoup de modèles du robot balanceur qui ont les mêmes principes de base, on présente quelques robots connus.

Modèle de Kazuo Yamafuji : Le premier robot à deux roues a été construit par Kazuo Yamafuji un professeur à l'Université de Electra-Communications de Tokyo en 1986. Selon un article paru dans le Japan Times, le robot peut simuler le comportement d'un pendule inversé. Il s'agit notamment d'une conception avec un essieu (muni de deux roues) et d'un chariot contenant le dispositif de stabilisation qui l'empêche de basculer. [1]

FIGURE 1.4 – Modèle de Kazuo Yamafuji 1986

Le robot de Joe-le : Le 25 janvier 2000 le robot Joe-le accomplit ses premières évolutions. Il est né d'une idée germée en 1996 au laboratoire de l'Electronique Industrielle de l'école polytechnique fédérale de Lausanne. Pour parvenir à son équilibre, un régulateur électronique stabilise la plateforme en exerçant un couple de redressement opposé au sens de basculement. [9]

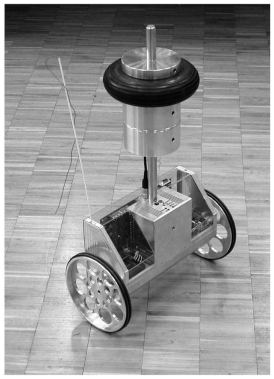

FIGURE 1.5 – Modèle de Joe-le 2000

Segway : Segway est la première entreprise qui a pu commercialiser ce robot comme un outil de transport. le SEGWAY HT a été inventé par Dean Kamen. C'est un véhicule électrique monoplace, constitué d'une plateforme munie de deux roues sur laquelle l'utilisateur se tient debout, d'un système de stabilisation gyroscopique et d'un manche de maintien et de conduite.[10]

FIGURE 1.6 – SEGWAY HT

Le robot Rezero : En Avril 2010, le robot Rezero fait son apparition pour la première fois. Il est monté sur une grosse boule et se tient en équilibre en mesurant constamment son inclinaison avec un détecteur (160 fois par seconde). Ensuite il se contrebalance et évite de

basculer en agissant sur ses moteurs. Si quelque chose ne marche pas dans le processus, Rezero tombe immédiatement parterre. A cause de son instabilité, Rezero est toujours en mouvement.[1]

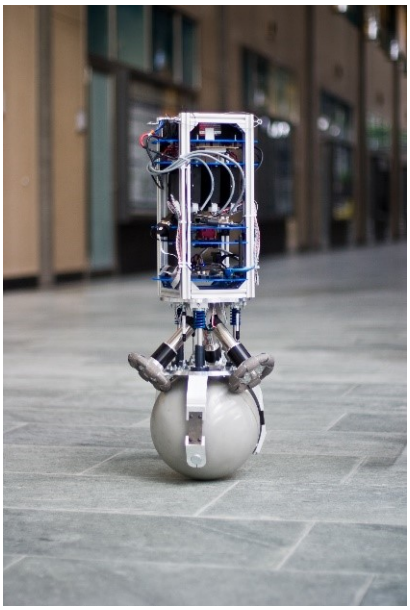

FIGURE 1.7 – Robot Rezero

iBOT : Après l'insuccès commerciale de iBOT en 2009, Toyota avec segway relance un nouveau modèle de iBOT en May 2016. iBOT est un fauteuil roulant monte-escalier. Il a donné une plus grande indépendance «verticale» aux personnes handicapées.[11]

FIGURE 1.8 – iBOT

Handle : Début février 2017, la société nord-américaine Boston Dynamics a annoncé sa dernière création : Handle, un robot bipède monté sur roues doté d'une agilité surprenante. Ce robot d'allure vaguement humanoïde est doté d'un système de stabilisation gyroscopique similaire à celui du Segway et de deux jambes articulées. Selon Boston Dynamics, c'est là une combinaison idéale. D'après le constructeur, Handle mesure 1,98 mètre de haut, il peut évoluer à environ 15 km/h avec une autonomie de 25 kilomètres. Surtout, il peut sauter jusqu'à 1,2 mètre de haut au-dessus d'obstacles et soulever des charges d'environ 45 kg.[12]

FIGURE 1.9 – Handle

1.4 Modélisation du robot

On a fait la partie de modélisation à base de [6].

Afin de pouvoir concevoir un correcteur capable de stabiliser le robot, il est nécessaire de mettre au point un modèle mathématique qui représente fidèlement son comportement. La dynamique du robot sera représentée par deux modèles découplés, le premier décrivant la dynamique d'inclinaison et de déplacement linéaire, et le deuxième décrivant la dynamique d'angle de direction. La dynamique globale du robot a été découplée afin de pouvoir concevoir deux contrôleurs séparément ; le premier pour l'angle d'inclinaison et le déplacement linéaire et le deuxième pour l'angle de direction. De cette façon, il est possible d'assigner différentes performances aux deux sous-systèmes et éviter un couplage des dynamiques par le contrôleur.

Les valeurs des paramètres du moteur et de la boîte de réduction ont été obtenues à partir de leurs fiches techniques tandis que les valeurs des moments d'inertie du robot ont été estimées à partir du modèle de conception mécanique assisté par ordinateur sous le logiciel SOLIDWORKS et les poids des différentes composantes ont été obtenus en les pesant avec une balance électronique. Toutes les valeurs des paramètres sont présentées au tableau 1.1.

Symbole	**Description**	**Valeur**	**Unité**
r_w	rayon des roues	5.5	cm
M	masse de la moitié du robot, y compris une roue	0.5335	kg
m_b	masse de la moitié du corps du robot	0.4795	kg
m_w	masse de l'une des roues	0.054	kg
I_{yy}	moment d'inertie de la moitié du corps du robot	40.15	$kg.cm^2$
I_{zz}	moment d'inertie du robot autour de l'axe Z	4.994	$kg.cm^2$
Iw	moment d'inertie de l'une des roues	0.88335	$kg.cm^2$
d	distance entre l'arbre du moteur et le centre de gravité	6.1	cm
S	distance entre les roues	25	cm
K_t	constante de couple des moteurs	0.019	$\frac{N.m}{A}$
K_e	constante de force électromotrice des moteurs	0.019	$\frac{V.S}{rad}$
r_a	résistance d'armature des moteurs	2.7	Ω
r_g	rapport de la boîte de réduction	43.7	$\frac{N}{A}$
η	efficacité de la boîte de réduction	1	%
C_f	constante de friction	0.002	$\frac{N.m.s}{rad}$

TABLE 1.1 – Paramètres du système

Pour le développement du modèle mathématique représentant la dynamique du robot, certaines hypothèses doivent être faites. De façon générale, on considère que le robot se maintient autour de la position verticale, que ses roues restent en contact avec le sol en tout temps. Les forces de réactions entre le corps du robot et les roues ainsi que la force centrifuge due au mouvement d'inclinaison du robot sont également négligées. On considère que les moteurs appliquent des couples aux roues et sur le corps du robot simultanément et que les roues appliquent à leur tour des forces sur le sol provoquant une accélération linéaire du robot ainsi qu'une accélération angulaire autour de l'axe vertical faisant varier l'angle d'orientation du robot. Ces hypothèses sont prises afin d'obtenir un modèle linéaire, bien qu'approximatif, relativement simple représentant bien la dynamique du système autour de la position d'équilibre.

1.4.1 Inclinaison et déplacement linéaire

Tout d'abord, un modèle pour la dynamique d'inclinaison et de déplacement linéaire est mis au point en faisant l'hypothèse que le robot se déplace en ligne droite. En tenant compte de la symétrie du robot par rapport à l'axe vertical, il est possible de ne considérer que la moitié du robot avec un seul moteur, en notant que des couples de valeurs égales doivent être appliqués par chacun des deux moteurs afin de provoquer un déplacement purement linéaire du robot. Par conséquent, la même tension doit être fournie aux deux moteurs et cette tension $u_x(t)$ sera considérée comme l'entrée de ce sous-système. En se référant aux variables et paramètres définis dans la liste des notations et des symboles, on peut faire les observations suivantes :

- la masse de la moitié du robot M consiste en la somme de la moitié de la masse du corps m_c et de la masse d'une roue m_r :

$$M = m_c + m_r$$

- la constante de couple K_t et la constante de la force électromotrice K_e sont équivalentes dans le système d'unités international
- la position linéaire du robot $x(t)$ peut être obtenue à partir du déplacement angulaire d'une roue $\theta(t)$ et le rayon d'une roue r de cette façon :

$$x(t) = r_w\theta(t) \tag{1.1}$$

- le déplacement angulaire de l'arbre du moteur $\theta_i(t)$ est relié au déplacement angulaire de l'arbre de la boite de réduction $\theta_o(t)$ par le facteur de réduction r_g de la boite de Réduction de la façon suivante :

$$\theta_i(t) = r_g\theta_o(t) \tag{1.2}$$

- le déplacement angulaire de l'arbre de la boite de réduction $\theta_o(t)$ est constitué du déplacement angulaire de la roue $\theta(t)$ et de l'inclinaison du robot $\psi(t)$:

$$\theta_o(t) = \theta(t) - \psi(t) \tag{1.3}$$

- la force de friction $F(t)$ entre la roue et le sol est responsable d'engendrer une accélération linéaire du robot :

$$F(t) = M\ddot{x}(t) \tag{1.4}$$

Dynamique d'un moteur

Le couple généré par un moteur est proportionnel à son courant :

$$T_i(t) = K_t i(t)$$

En faisant l'hypothèse que l'inductance du moteur est négligeable, il est possible d'exprimer le courant en fonction de la tension d'entrée et la force électromotrice en utilisant la loi d'Ohm :

$$i(t) = \frac{u_x(t)}{r_a} - \frac{K_e\dot{\theta}_i(t)}{r_a} \tag{1.5}$$

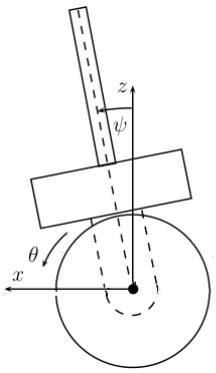

FIGURE 1.10 – Inclinaison et déplacement linéaire du robot

On peut maintenant établir la relation entre le couple généré par le moteur et la tension appliquée à son entrée en combinant les équations 1.4.1 et 1.5 de cette façon :

$$T_i(t) = K_t \left[\frac{u_x(t)}{r_a} - \frac{K_e \dot{\theta}_i(t)}{r_a} \right] \tag{1.6}$$

Couple appliqué à une roue

La relation exprimant le couple appliqué à une roue par un moteur à travers la boîte de réduction est la suivante :

$$T(t) = \eta r_g T_i(t) \tag{1.7}$$

En remplaçant $T_i(t)$ par la relation trouvée en 1.6, on obtient la relation suivante :

$$T(t) = \frac{\eta r_g K u_x(t)}{r_a} - \frac{\eta r_g K^2}{r_a} \dot{\theta}_i(t) \tag{1.8}$$

En utilisant la relation entre $\theta_i(t)$ et $\theta_o(t)$ donnée par l'équation 1.2, on peut exprimer l'équation précédente en fonction de $\theta_o(t)$ de cette façon :

$$T(t) = \frac{\eta r_g K u_x(t)}{r_a} - \frac{\eta r_g^2 K^2}{r_a} \dot{\theta}_o(t) \tag{1.9}$$

Puis, en considérant la dérivée de l'équation 1.3 par rapport au temps, on obtient :

$$T(t) = \frac{\eta r_g K}{r_a} u_x(t) - \frac{\eta r_g^2 K^2}{r_a} \dot{\theta}(t) + \frac{\eta r_g^2 K^2}{r_a} \dot{\psi}(t) \tag{1.10}$$

Finalement, à partir de l'équation 1.1, on obtient la relation donnant le couple appliqué à une roue $T(t)$ en fonction de la tension à l'entrée d'un moteur $u_x(t)$, de la vitesse de déplacement linéaire du robot $\dot{x}(t)$ et de la dérivée par rapport au temps de son angle d'inclinaison $\dot{\psi}(t)$:

$$T(t) = \frac{\eta r_g K}{r_a} u_x(t) - \frac{\eta r_g^2 K^2}{r_a r_w} \dot{x}(t) + \frac{\eta r_g^2 K^2}{r_a} \dot{\psi}(t) \tag{1.11}$$

Dynamique d'inclinaison

En se référant à la figure 1.11, et en notant que le couple appliqué à une roue est également appliqué à la moitié du corps du robot, on peut faire le bilan des couples agissant sur la moité du corps du robot comme suit :

$$I_{yy} \ddot{\psi}(t) = m_b g d \sin(\psi(t)) - T(t) - C_f \dot{\psi}(t) + C_f \dot{\theta}(t) \tag{1.12}$$

En remplaçant T (t) par la relation trouvée en (1.11), on obtient la relation suivante :

$$I_{yy} \ddot{\psi}(t) = m_b g d \sin(\psi(t)) - \frac{\eta r_g K}{r_a} u_x(t) + \frac{\eta r_g^2 K^2}{r_a r_w} \dot{x}(t) - \frac{\eta r_g^2 K^2}{r_a} \dot{\psi}(t) - C_f \dot{\psi}(t) + C_f \frac{\dot{x}(t)}{r_w} \tag{1.13}$$

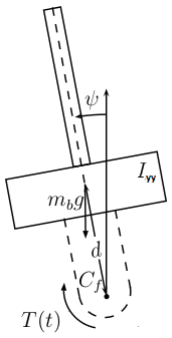

FIGURE 1.11 – Diagramme de corps libre de la dynamique d'inclinaison

ce qui donne à son tour :

$$\ddot{\psi}(t) = \frac{m_b g d \sin(\psi(t))}{I_{yy}} - \left[\frac{\eta r_g^2 K^2 + C_f r_a}{r_a I_{yy}}\right]\dot{\psi}(t) + \left[\frac{\eta r_g^2 K^2 + C_f r_a}{r_a r_w I_{yy}}\right]\dot{x}(t) - \frac{\eta r_g K}{r_a I_{yy}} u_x(t) \quad (1.14)$$

Par contre, cette équation contient le terme non linéaire $\sin(\psi(t))$ et nous devons la linéariser afin de pouvoir utiliser la théorie de la commande des systèmes linéaires. En considérant l'hypothèse que le robot reste autour de la verticale, $\psi(t)$ peut être considéré comme étant petit, et on obtient $\sin(\psi(t)) \approx \psi(t)$, ce qui implique à son tour que :

$$\ddot{\psi}(t) = \frac{m_b g d}{I_{yy}}\psi(t) - \left[\frac{\eta r_g^2 K^2 + C_f r_a}{r_a I_{yy}}\right]\dot{\psi}(t) + \left[\frac{\eta r_g^2 K^2 + C_f r_a}{r_a r_w I_{yy}}\right]\dot{x}(t) - \frac{\eta r_g K}{r_a I_{yy}} u_x(t) \quad (1.15)$$

Dynamique des roues et du déplacement linéaire

En se référant à la figure 1.12, on effectue le bilan des couples appliqués à l'une des roues comme suit :

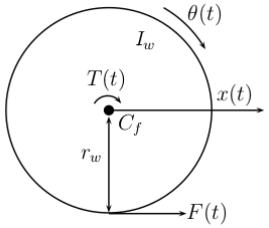

FIGURE 1.12 – Diagramme de corps libre d'une roue

$$I_w\ddot{\theta}(t) = T(t) - F(t)r_w - C_f\dot{\theta}(t) + C_f\dot{\psi}(t) \quad (1.16)$$

La force appliquée au sol par une roue $F(t)$, peut être remplacée par une relation équivalente selon l'équation 1.4 :

$$I_w\ddot{\theta}(t) = T(t) - r_w M\ddot{x}(t) - C_f\dot{\theta}(t) + C_f\dot{\psi}(t) \quad (1.17)$$

En remplaçant $T(t)$ par la relation trouvée en 1.11, on obtient la relation suivante :

$$I_w\ddot{\theta}(t) = \frac{\eta r_g K}{r_a}u_x(t) - \frac{\eta r_g^2 K^2}{r_a r_w}\dot{x}(t) + \frac{\eta r_g^2 K^2}{r_a}\dot{\psi}(t) - r_w M\ddot{x}(t) - \frac{C_f}{r_w}\dot{x}(t) + C_f\dot{\psi}(t) \quad (1.18)$$

Cette équation peut s'écrire sous la forme suivante :

$$\left[\frac{I_w}{r_w} + r_w M\right]\ddot{x}(t) = \left[\frac{\eta r_g^2 K^2 + C_f r_a}{r_a}\right]\dot{\psi}(t) - \left[\frac{\eta r_g^2 K^2 + C_f r_a}{r_a r_w}\right]\dot{x}(t) + \frac{\eta r_g K}{r_a}u_x(t) \quad (1.19)$$

Ce qui donne à son tour :

$$\ddot{x}(t) = \left[\frac{\eta r_w r_g^2 K^2 + C_f r_a r_w}{r_a(I_w + r_w^2 M)}\right]\dot{\psi}(t) - \left[\frac{\eta r_g^2 K^2 + C_f r_a}{r_a(I_w + r_w^2 M)}\right]\dot{x}(t) + \frac{\eta r_g r_w K}{r_a(I_w + r_w^2 M)}u_x(t) \tag{1.20}$$

1.4.2 Modèle d'état de la dynamique linéaire

En définissant le vecteur d'état $X) = \begin{bmatrix}\psi & \dot{\psi} & x & \dot{x}\end{bmatrix}_T$ et le vecteur de sortie $y = [\psi \;\; x]^T$, à partir des équations 1.15 et 1.20, on obtient le modèle d'état suivant :

$$\begin{cases}\dot{X}(t) = A\ X(t) + B\ u_x(t)\\ y(t) = C\ X(t)\end{cases} \tag{1.21}$$

$$u_x = \frac{u_l + u_r}{2}$$
$$u_h = u_l - u_r$$

Où,

$$A_x = \begin{bmatrix} 0 & 1 & 0 & 0 \\ \frac{m_b g d}{I_{yy}} & -\frac{\eta r_g^2 K^2 + C_f r_a}{r_a I_{yy}} & 0 & \frac{\eta r_g^2 K^2 + C_f r_a}{r_a r_w I_{yy}} \\ 0 & 0 & 0 & 1 \\ 0 & \frac{\eta r_w r_g^2 K^2 + C_f r_a r_w}{r_a(I_w + r_w^2 M)} & 0 & -\frac{\eta r_g^2 K^2 + C_f r_a}{r_a(I_w + r_w^2 M)} \end{bmatrix}$$

$$B_x = \begin{bmatrix} 0 \\ -\frac{\eta r_g K}{r_a I_{yy}} \\ 0 \\ \frac{\eta r_g r_w K}{r_a(I_w + r_w^2 M)} \end{bmatrix},\ C_x = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

En tenant compte des valeurs des paramètres du système, nous obtenons les matrices nominales suivantes :

$$A_x = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 71.16 & -16.72 & 0 & 303.93 \\ 0 & 0 & 0 & 1 \\ 0 & 2.17 & 0 & -39.46 \end{bmatrix}$$

$$B_x = \begin{bmatrix} 0 \\ -38.66 \\ 0 \\ 5.02 \end{bmatrix},\ C_x = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

1.4.3 Angle de direction

Dynamique de l'angle de direction

Le modèle représentant la dynamique de l'angle de direction du robot est établi en prenant en considération le fait que des couples égaux mais opposés doivent être appliqués par les deux moteurs de manière à engendrer un mouvement de rotation pure au robot sans affecter son inclinaison et sa position linéaire. Par conséquent, des tensions égales mais opposées doivent être appliquées aux deux moteurs et l'amplitude de ces tensions $u_h(t)$ est considérée comme étant l'entrée de ce sous-système. Ici, l'hypothèse que le robot se trouve dans le voisinage de la position verticale et que par conséquent son moment d'inertie autour de l'axe vertical peut être considéré comme une constante, est prise en considération. En se référant aux variables et aux paramètres définis dans la liste des notations et des symboles, on débute par observer le déplacement des roues produit par un changement d'angle de direction. En se référant à la figure

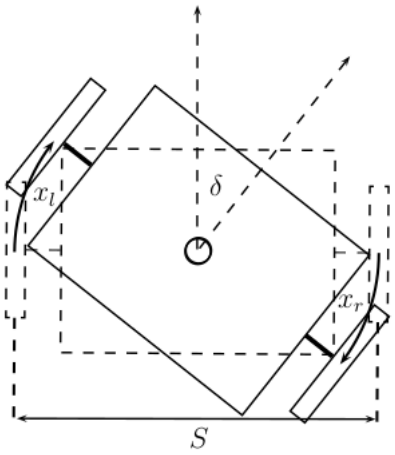

FIGURE 1.13 – Déplacements des roues lors d'un changement de direction

1.13, on peut voir que les distances parcourues par les deux roues $x_r(t)$ et $x_l(t)$ lorsque le robot tourne sur lui-même sont reliées à l'angle de direction $\delta(t)$ de la façon suivante :

$$x_l(t) = \delta(t)\frac{S}{2}$$

$$x_r(t) = -\delta(t)\frac{S}{2}$$

À partir de ces deux équations, on peut exprimer l'angle de direction en fonction des déplacements des roues de la façon suivante :

$$\delta(t) = \left[\frac{x_l(t) - x_r(t)}{S}\right] \tag{1.22}$$

Les déplacements des roues sont reliés à leurs déplacements angulaires par les équations suivantes :

$$x_r(t) = r_w\theta_r(t) \tag{1.23}$$

$$x_l(t) = r_w\theta_l(t) \tag{1.24}$$

À partir de la définition des tensions appliquées aux moteurs produisant un mouvement de changement de direction stipulant que celles-ci doivent être égales mais opposées, on peut établir la relation suivante entre la tension appliquée au moteur gauche $u_l(t)$,la tension appliquée au moteur droit $u_r(t)$ et l'entrée $u_h(t)$ du sous-système qui régit la dynamique d'angle de direction :

$$u_l(t) - u_r(t) = u_h(t) \tag{1.25}$$

À partir de l'équation 1.16, on peut exprimer la force $F(t)$ appliquée par une roue sur le sol par la relation suivante :

$$F(t) = \frac{T(t) - I_w\ddot{\theta}(t) - C_f\dot{\theta}(t) - C_f\dot{\psi}(t)}{r_w} \tag{1.26}$$

Ensuite, à l'aide de l'équation 1.11, on peut exprimer cette dernière équation comme suit :

$$F(t) = -\left[\frac{\eta r_g^2K^2 + C_f r_a}{r_a r_w}\right]\dot{\psi}(t) - \left[\frac{\eta r_g^2K^2 + C_f r_a}{r_a r_w^2}\right]\dot{x}(t) + \frac{\eta r_g K}{r_a r_w}u(t) - \frac{I_w}{r_w}\ddot{\theta}(t) \tag{1.27}$$

On peut maintenant exprimer les forces appliquées au sol par chacune des roues par les équations suivantes :

$$F_l(t) = -\left[\frac{\eta r_g^2K^2 + C_f r_a}{r_a r_w}\right]\dot{\psi}(t) - \left[\frac{\eta r_g^2K^2 + C_f r_a}{r_a r_w^2}\right]\dot{x}_l(t) + \frac{\eta r_g K}{r_a r_w}u_l(t) - \frac{I_w}{r_w}\ddot{\theta}_l(t) \tag{1.28}$$

$$F_r(t) = -\left[\frac{\eta r_g^2 K^2 + C_f r_a}{r_a r_w}\right]\dot{\psi}(t) - \left[\frac{\eta r_g^2 K^2 + C_f r_a}{r_a r_w^2}\right]\dot{x}_r(t) + \frac{\eta r_g K}{r_a r_w}u_r(t) - \frac{I_w}{r_w}\ddot{\theta}_r(t) \tag{1.29}$$

En se référant à la figure 1.14, on effectue le bilan des couples agissant sur le robot autour de l'axe vertical :

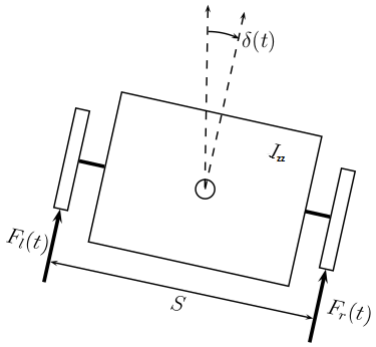

FIGURE 1.14 – Diagramme de corps libre de la dynamique d'angle de direction

$$I_{zz}\ddot{\delta}(t) = [F_l(t) - F_r(t)]\,\frac{S}{2} \tag{1.30}$$

En remplaçant $F_l(t)$ et $F_r(t)$ par les expressions trouvées en 1.28 et 1.29, on établit la relation suivant :

$$I_{zz}\ddot{\delta}(t) = \left[\frac{\eta S r_g^2 K^2 + S C_f r_a}{2 r_a r_w^2}\right][\dot{x}_r(t) - \dot{x}_l(t)] + \frac{\eta S r_g K}{2 r_a r_w}[u_l(t) - u_r(t)] + \frac{S I_w}{2 r_w}\left[\ddot{\theta}_r(t) - \ddot{\theta}_l(t)\right] \tag{1.31}$$

À partir de l'équation 1.25, on peut exprimer cette dernière équation en fonction de l'entrée $u_h(t)$ de la manière suivante :

$$I_{zz}\ddot{\delta}(t) = \left[\frac{\eta S r_g^2 K^2 + S C_f r_a}{2 r_a r_w^2}\right][\dot{x}_r(t) - \dot{x}_l(t)] + \frac{\eta S r_g K}{2 r_a r_w}u_h(t) + \frac{S I_w}{2 r_w}\left[\ddot{\theta}_r(t) - \ddot{\theta}_l(t)\right] \tag{1.32}$$

En considérant la dérivée par rapport au temps de l'équation 1.22, on peut exprimer l'équation précédente en terme de la vitesse de changement de direction $\delta(t)$ comme suit :

$$I_{zz}\ddot{\delta}(t) = -\left[\frac{\eta S^2 r_g^2 K^2 + S^2 C_f r_a}{2 r_a r_w^2}\right]\dot{\delta}(t) + \frac{\eta S r_g K}{2 r_a r_w}u_h(t) + \frac{S I_w}{2 r_w}\left[\ddot{\theta}_r(t) - \ddot{\theta}_l(t)\right] \tag{1.33}$$

Les accélérations angulaires des roues $\ddot{\theta}_r(t)$ et $\ddot{\theta}_l(t)$ peuvent premièrement être exprimées en fonction des accélérations des roues en considérant la dérivée seconde par rapport au temps des équations 1.23 et 1.24 de cette façon :

$$I_{zz}\ddot{\delta}(t) = -\left[\frac{\eta S^2 r_g^2 K^2 + S^2 C_f r_a}{2 r_a r_w^2}\right]\dot{\delta}(t) + \frac{\eta S r_g K}{2 r_a r_w}u_h(t) + \frac{S I_w}{2 r_w^2}[\ddot{x}_r(t) - \ddot{x}_l(t)] \tag{1.34}$$

Puis, ces accélérations des roues $\ddot{x}_r(t)$ et $\ddot{x}_l(t)$ peuvent être exprimées à leur tour en terme de l'accélération du changement de direction $\ddot{\delta}(t)$ en considérant la dérivée seconde par rapport au temps de l'équation 1.22 de la façon suivante :

$$I_{zz}\ddot{\delta}(t) = -\left[\frac{\eta S^2 r_g^2 K^2 + S^2 C_f r_a}{2 r_a r_w^2}\right]\dot{\delta}(t) + \frac{\eta S r_g K}{2 r_a r_w}u_h(t) - \frac{S^2 I_w}{2 r_w^2}\ddot{\delta}(t) \tag{1.35}$$

ce qui donne :

$$\left[I_{zz} + \frac{S^2 I_w}{2r_w^2}\right] \ddot{\delta}(t) = -\left[\frac{\eta S^2 r_g^2 K^2 + S^2 C_f r_a}{2 r_a r_w^2}\right] \dot{\delta}(t) + \frac{\eta S r_g K}{2 r_a r_w} u_h(t) \tag{1.36}$$

Finalement, on obtient :

$$\ddot{\delta}(t) = -\left[\frac{\eta S^2 r_g^2 K^2 + S^2 C_f r_a}{r_a (2 r_w^2 I_{zz} + S^2 I_w)}\right] \dot{\delta}(t) + \left[\frac{2\eta S r_w r_g K}{2 r_a (2 r_w^2 I_{zz} + S^2 I_w)}\right] u_h(t) \tag{1.37}$$

1.4.4 Modèle d'état de la dynamique de direction

En définissant le vecteur d'état $x_h = \begin{bmatrix}\delta & \dot{\delta}\end{bmatrix}$ et le vecteur de sortie $y_h = \delta$, à partir l'équation 1.37, on obtient le modèle d'état suivant :

$$\begin{cases} \dot{x}_h(t) = A_h x_h(t) + B_h u_h(t) \\ y_h(t) = C_h x_h(t) \end{cases} \tag{1.38}$$

Où

$$A_h = \begin{bmatrix} 0 & 1 \\ 0 & -\frac{\eta S^2 r_g^2 K^2 + S^2 C_f r_a}{r_a (2 r_w^2 I_{zz} + S^2 I_w)} \end{bmatrix}$$

$$B_h = \begin{bmatrix} 0 \\ \frac{2\eta S r_w r_g K}{2 r_a (2 r_w^2 I_{zz} + S^2 I_w)} \end{bmatrix}, \; C_h = \begin{bmatrix} 1 & 0 \end{bmatrix}$$

En tenant compte des valeurs des paramètres du système, nous obtenons les matrices suivantes :

$$A_h = \begin{bmatrix} 0 & 1 \\ 0 & -491.55 \end{bmatrix}$$

$$B_h = \begin{bmatrix} 0 \\ 250.10 \end{bmatrix}, \; C_h = \begin{bmatrix} 1 & 0 \end{bmatrix}$$

1.4.5 Modèle d'état du robot

Après la modélisation des deux sous-système , On les assemble dans un système global du robot.
En définissant le vecteur d'état $X = \begin{bmatrix}\psi & \dot{\psi} & x & \dot{x} & \delta & \dot{\delta}\end{bmatrix}^T$ et le vecteur de sortie $y = [\psi \;\; x \;\; \delta]^T$

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 71.16 & -16.71 & 0 & 303.93 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 2.17 & 0 & -39.46 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & -491.55 \end{bmatrix}$$

$$B = \begin{bmatrix} 0 & 0 \\ -38.66 & 0 \\ 0 & 0 \\ 5.02 & 0 \\ 0 & 0 \\ 0 & 250.10 \end{bmatrix}, \; C = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

$u = \begin{bmatrix} u_x & u_h \end{bmatrix}$

En réalité, on commande le robot par u_l et u_r qui sont les tensions du moteur gauche et du moteur droit, respectivement. On peut calculer ces tension par les relations :

$$u_r = \frac{2u_x - u_h}{2} \tag{1.39}$$

$$u_l = \frac{2u_x + u_h}{2} \tag{1.40}$$

1.5 Simulation en boucle ouverte

Pour vérifier si la sortie du modèle linéaire et non linéaire semble acceptable, un simple test en boucle ouverte a été effectué. Le pendule a commencé à 0.1 rad, ce qui signifie légèrement déplacé du point d'équilibre instable, puis simulé pendant une durée de 20 secondes.

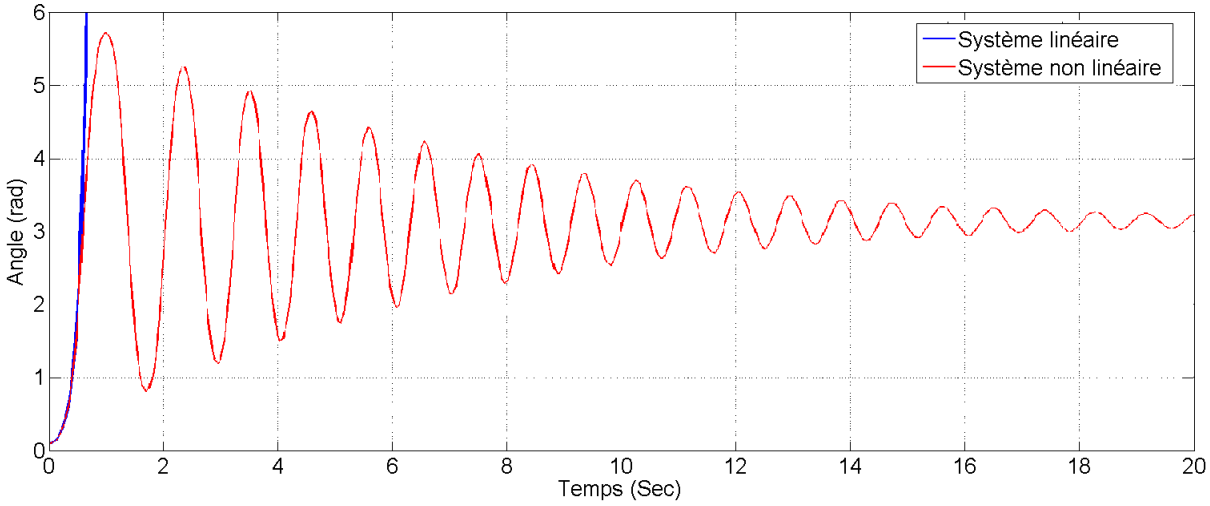

FIGURE 1.15 – Simulation du système en boucle ouverte de l'angle ψ

Le modèle non linéaire montre que le pendule tombe du point de départ et commence à osciller autour du point d'équilibre stable. Il est clair que les oscillations diminuent avec le temps. Il est également clair que le modèle linéaire présente une dynamique similaire pour la région proche du point d'équilibre instable, mais comme prévu, il ne peut pas donner une représentation précise du système lorsqu'il se déplace trop loin de ce point (Plus de 1.4 radian par rapport au point d'équilibre).

A l'aide de Matlab (Pole-Zero Map), on peut tracer les pôles du système pour vérifie que le système est instable car il y a un pôle dans le demi-plan droite. Idéalement, tous les pôles devraient être sur le demi-plan gauche pour que le système soit stable.

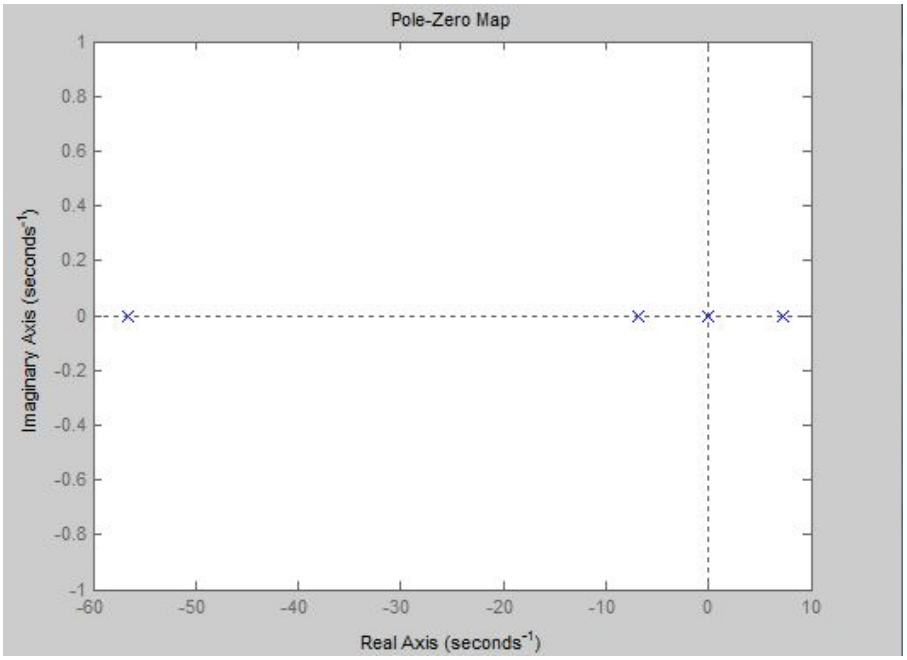

FIGURE 1.16 – Pôles du système

1.6 Conclusion

La modélisation du robot balanceur a montré que le système est composé de deux sous-système indépendant,le premier est la dynamique d'inclinaison et de déplacement linéaire, et le deuxième est la dynamique d'angle de direction. Ce modèle nous a bien montré que le système est multi variable et non-linéaire, Les simulations effectuées en boucle ouverte sur le robot nous ont permis de constater l'instabilité de celui-ci. On essayera dans le chapitre suivant de commander ce système en utilisant trois lois de commande.

Chapitre 2

Conception du robot balanceur

Chapitre 2

Conception du robot balanceur

2.1 Introduction

On a réservé ce chapitre pour présenter la procédure de réalisation du robot. On va commencer par la présentation du système mécanique. Puis, on va présenter le système électrique. On aborde la conception détaillée de chaque partie des deux systèmes.

2.2 Système mécanique

La forme générale du robot est un corps cylindrique pose sur deux roues. Les roues sont placées parallèlement l'une à l'autre, les moteurs sont fixés sur une base rectangulaire. La figure 2.1 montre le design CAD (solide Works) du robot vue de face et de côté.

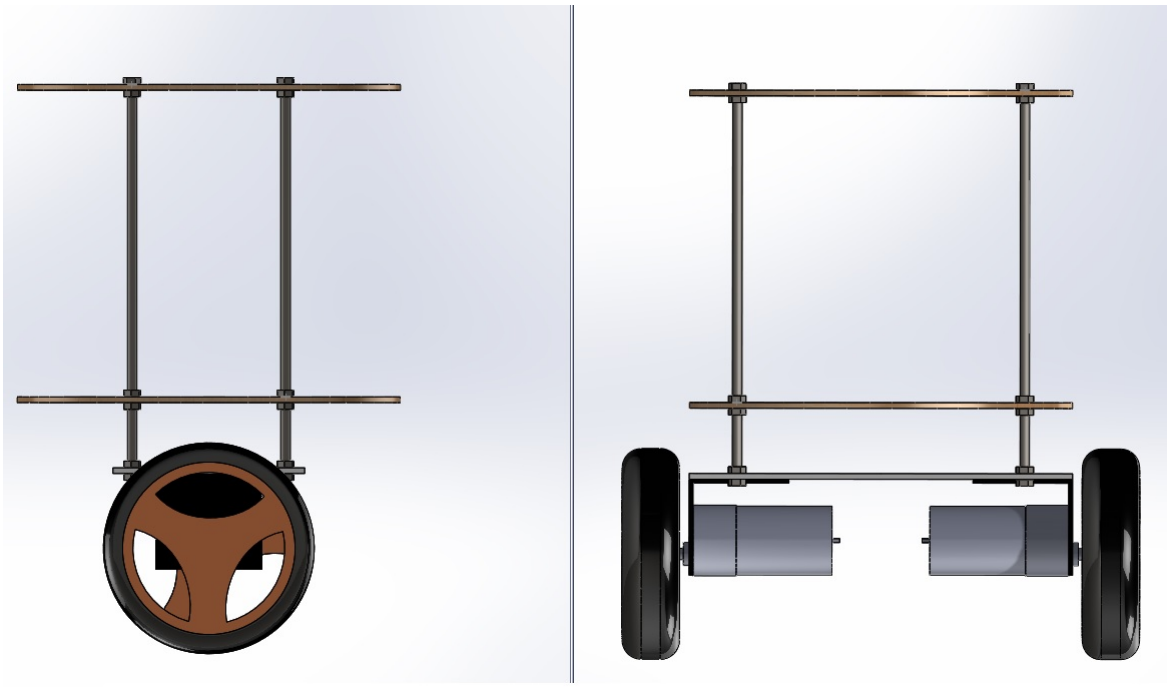

FIGURE 2.1 – Vue de face et de côté du robot

2.2.1 Forme du robot

Le robot contient quatre tiges de fer qu'on peut représenter comme 4 pieds d'étagères, les étagères sont des disques de bois. Sur l'étagère bas il y a la batterie, les moteurs y sont fixés par des supports en aluminium. Sur le deuxième étagère il y a le driver de moteur, Arduino, MPU 6050, Bluetooth et le capteur sonar. J'ai fait ce design pour faciliter les modifications tel que je peux ajouter un étagère au robot pour ajouter une autre fonction comme le capteur de distance ou la camera. Les dimensions de robot sont $20 * 27 * 29.7cm$ et son poids sans batterie est $1067g$.

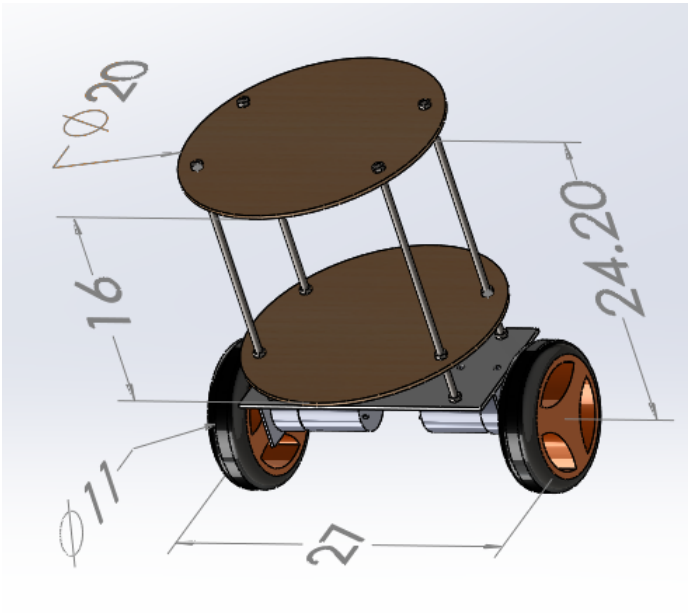

FIGURE 2.2 – Démentions du robot

2.2.2 Les roues

Les deux roues $11 * 2.5\ cm$ sont choisies pour le poids et le diamètre qui assure la vitesse désirée du robot, chaque roue pèse $56g$. elles sont faites de plastic et de caoutchouc et sont directement fixées aux moteurs .

FIGURE 2.3 – Roue du moteur

2.2.3 Base du robot

On a fait la base du robot en forme rectangulaire fabriqué de résine. Elle rassemble le corps du robot avec les moteurs et porte la batterie. Les dimensions de la base sont $10 * 20 * 3cm$ et son poids est $91g$.

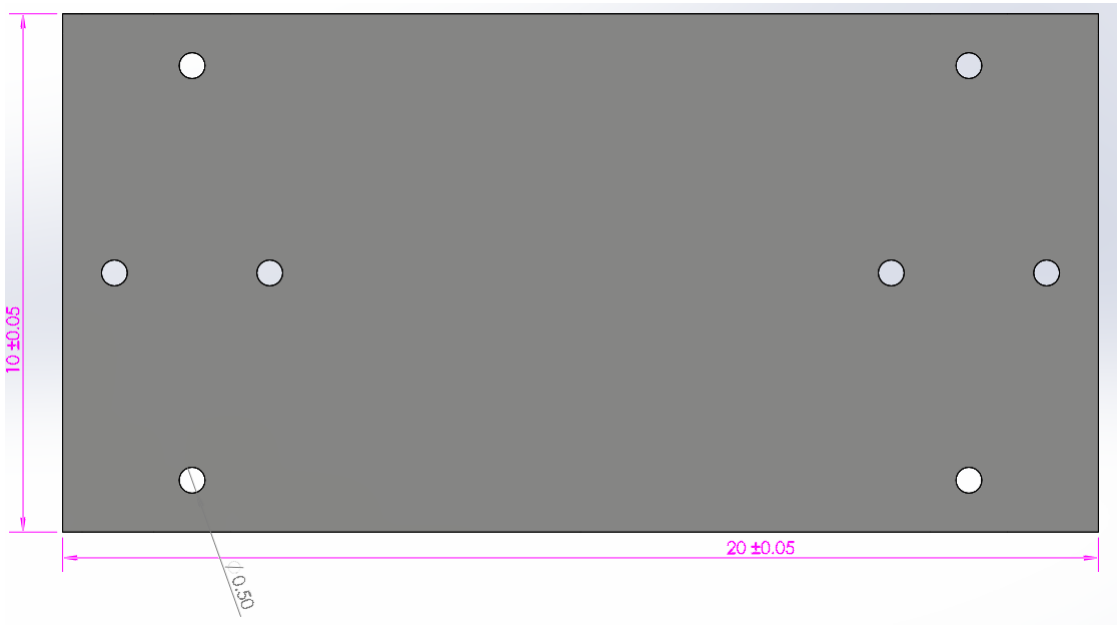

FIGURE 2.4 – Base du robot

2.2.4 Supports des moteurs

Pour fixer les moteurs sur la base du robot, On a fabriqué des pièces en aluminium en forme de "L". Le poids de ce support est $18g$.

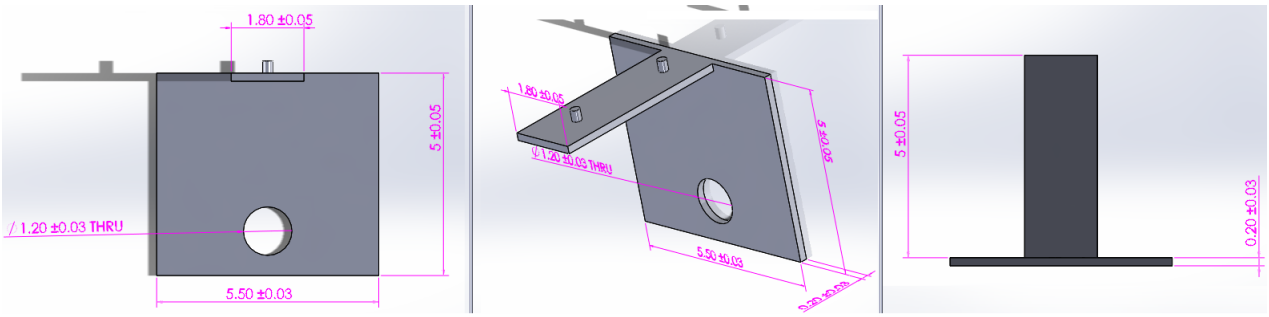

FIGURE 2.5 – Support du moteur

2.2.5 Les disques en bois

Le rôle des disques en bois est comme des étagères, ils portent les composants électrique, le nombre de disques est variable avec le nombre de fonctions du robot. pour notre robot on a besoin deux disques de $75g$, $20cm$ de diamètre et de $3mm$ d'épaisseur. Chaque disque contient 4 trous de $0.5cm$ pour les fixer sur le robot et un trou dans le centre pour passer les fils électrique de l'étagère a l'autre.

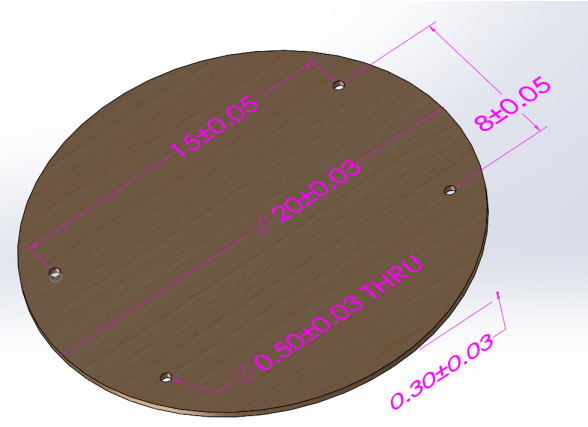

FIGURE 2.6 – Disque de bois

2.2.6 Les Tiges filetées

Les tiges filetées sont les bases du corps du pendule. elles sont fixées sur la base rectangulaire du robot et portent les disques. La longueur de la tige est $21cm$, son diamètre est $0.5cm$, son poids est $23g$. On a utilisé des écrous pour fixer les disques sur les tiges.

FIGURE 2.7 – Tige filetée

FIGURE 2.8 – Fixation des disques

2.2.7 Assemblage du robot

On a conçu le robot dans une forme qui garantit la facilité de l'assemblage, le changement et l'adjonction des pièces. Les images suivantes montrent comment on a assembler les pièces du robot. On a utilisé les images de modèle 3D parce que elles sont plus claires que les images réel.

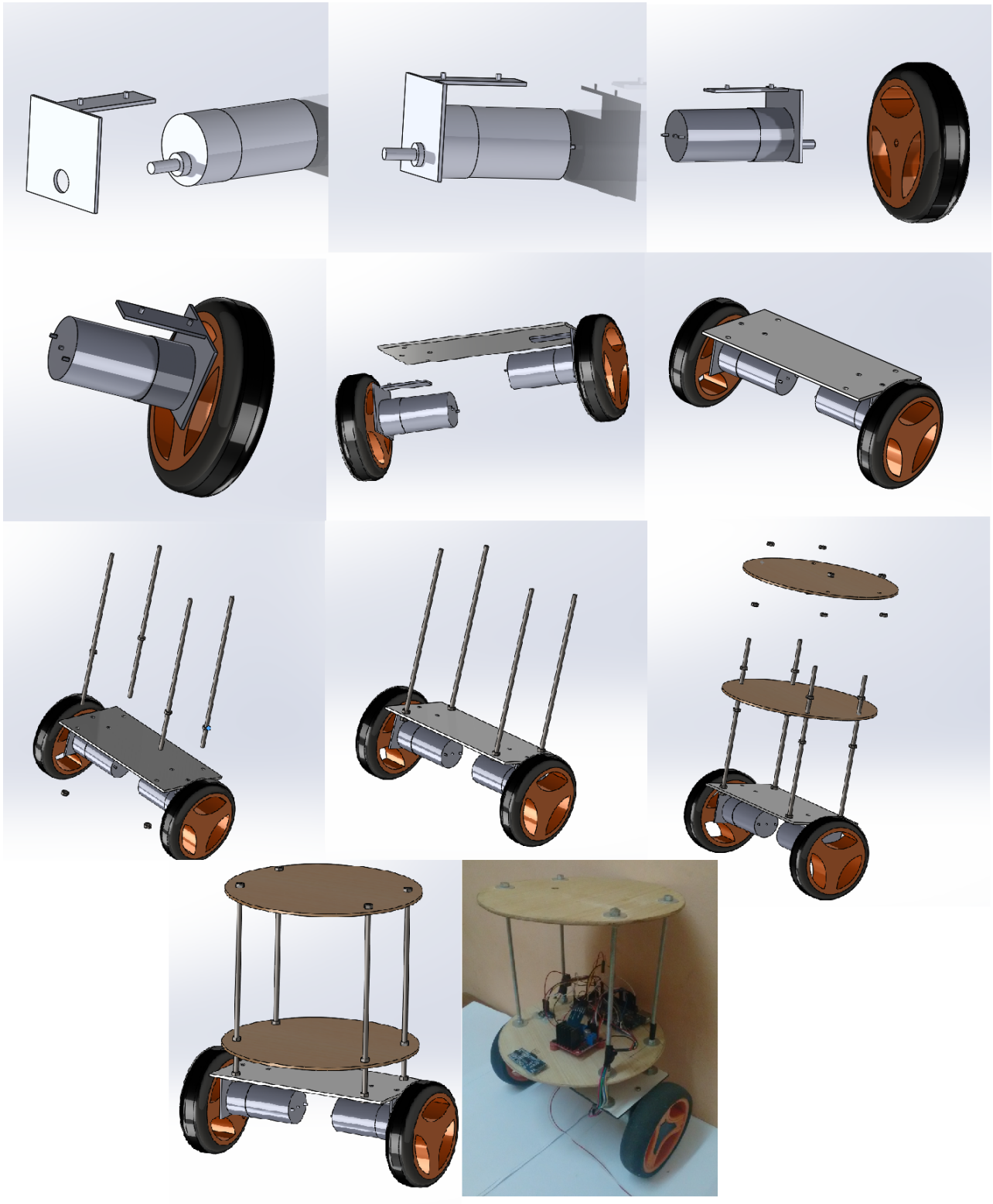

FIGURE 2.9 – L'assemblage du robot

2.3 Système électrique

Dans cette partie on va décrire les composants électriques qui sont utilisés dans le robot. La figure 2.10 montre le schéma électrique et la communication entre les composantes.

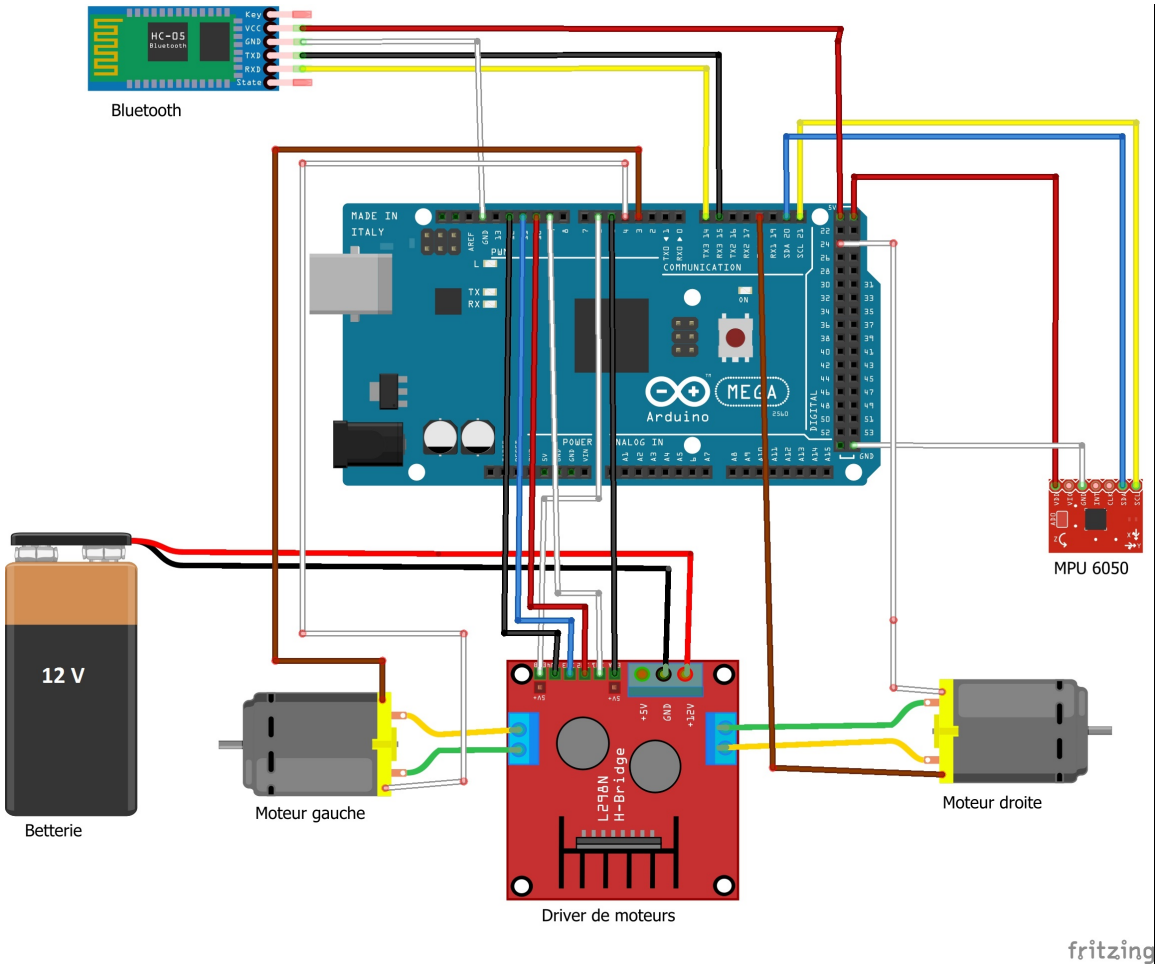

FIGURE 2.10 – Schéma électrique du robot

On a utilisé la carte Arduino comme un contrôleur, cette dernière est connectée à tous les capteurs, le driver des moteurs et les module de communication (USB, Bluetooth). Elle reçoit les données du robot à partir des capteurs par le protocole de communication I2C puis calcule la commande et l'envoie au driver des moteurs comme un signal PWM.

Les capteurs sont des module qui sont directement connectés à Arduino parce que chaque capteur contient le conditionneur et convertisseur analogique numérique et le protocole de communication.

On a utilisé le module Bluetooth (HC-05) et le port USB (porte série) pour faire communiquer l'Arduino avec l'ordinateur.

2.3.1 Le microcontrôleur Arduino

L'Arduino Mega est une carte microcontrôleur basé sur l'ATmega1280. Il dispose de 54 broches numériques d'entrée / sortie (dont 14 peuvent être utilisées comme sorties PWM), 16 entrées analogiques, 4 UART (ports série matériels), un oscillateur en cristal de 16 MHz, d'une connexion USB, une prise d'alimentation, d'une embase ICSP et un bouton de réinitialisation. Il contient tout le nécessaire pour soutenir le microcontrôleur, il suffit de le connecter à un ordinateur avec un câble USB ou avec un adaptateur AC-DC ou batterie pour commencer.[13]

La programmation sur Arduino se fait par l'interface Arduino IDE avec langage de programmation C ou C++.

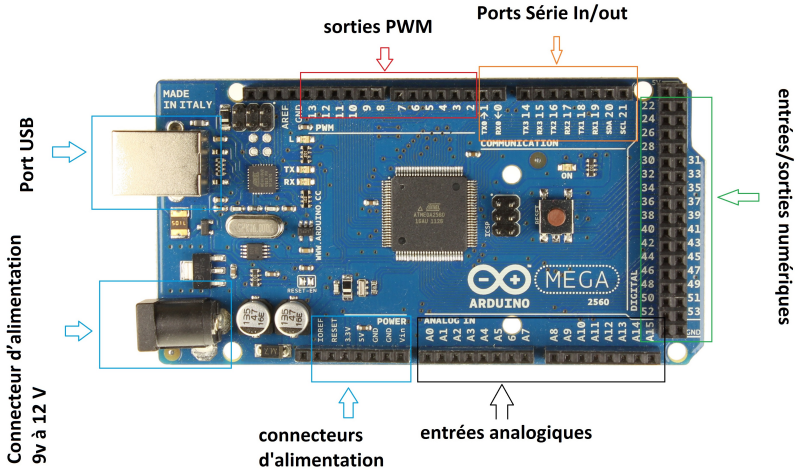

FIGURE 2.11 – Arduino Mega

2.3.2 Le driver des moteurs Module L298N

Le moteur électrique ne fonctionne pas si on le branche directement à la sortie du micro-contrôleur, donc on a besoin d'utiliser un pont H pour chaque moteur On a choisi d'utiliser le pont L298N. Ce breakout board est un Double Pont-H destiné au contrôle de moteur continu (H-Bridge Motor Driver). Il est basé sur le composant L298N qui est un double Pont-H conçu spécifiquement pour ce cas d'utilisation. C'est un module extrêmement utile pour le contrôler des robots et des ensembles mécanisés. Il est conçu pour supporter des tensions plus élevées, des courants importants tout en proposant une commande logique TTL (basse tension, courant faibles, idéal donc pour un microcontrôleur). Il peut piloter des charges inductives comme des relais, solénoides, moteurs continus et moteurs pas-à-pas. Les deux types de moteurs peuvent être contrôlés aussi bien en vitesse (PWM) qu'en direction. Toutes les sorties en puissance sont déjà protégées par des diodes anti-retour. Il s'agit d'un module prêt à l'emploi.[14]

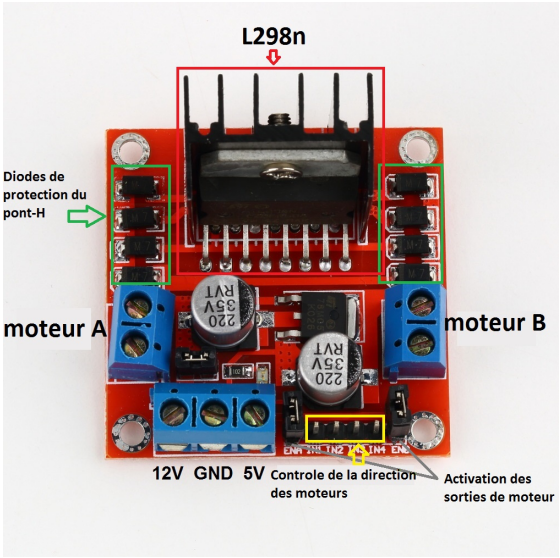

FIGURE 2.12 – Module L298N

2.3.3 Moteur

Chaque roue est motorisée indépendamment de l'autre, offrant le contrôle du robot par différentiation. Le moteur utilisé est un motoréducteur avec encodeur modèle GB37Y3530-12V-251R, il est un moteur à courant continu avec boite de réduction métallique 43.7 :1 et un encodeur intégré qui fournit une résolution de 16 compteurs par tour de l'arbre du moteur. Ce

moteur a un arbre de sortie en forme de D, il est destiné à être utilise a $12V$, bien que le moteur commence à tourner a des tensions faible que $1V$

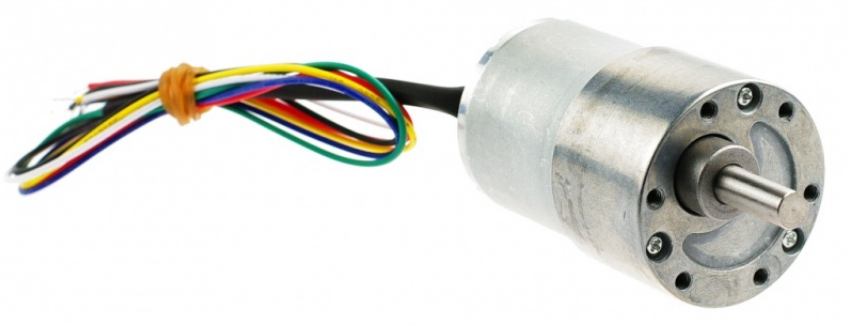

FIGURE 2.13 – Moteur DC

2.3.4 Batterie

Pour le source d'énergie, on a utilisé une batterie Lithium-polymer de 3 cellule et 11.1V avec un capacité de $2200mAh$. Le raison de utilisation 3 cellule est parce que 1 cellule de batterie LiPo a un voltage nominale $3.7V$ et a maximum voltage quand il est complètement chargée $4.2V$, Arduino a besoin de $9 - 12V$ pour un travail optimal et aussi parce que le voltage nominale de moteurs est $12V$.

FIGURE 2.14 – Batterie

2.3.5 Capteurs

Gyroscope

Il est nécessaire pour un système de balancement d'avoir la possibilité de déterminer l'angle d'inclinaison. Ceci peut être obtenu par l'utilisation d'un gyroscope.

Le gyroscope est un appareil qui exploite le principe de la conservation du moment angulaire en physique (ou encore stabilité gyroscopique ou effet gyroscopique), Il fut inventé par le physicien français Léon Foucault en 1852.[15] Il donne la position angulaire (selon un, deux ou trois axes). Il ne donne pas directement un angle d'orientation. Cet angle s'obtient par intégration dans le temps de la vitesse angulaire, en faisant attention au cumul des erreurs de dérivée. Le gyroscope permet de déterminer la rotation ressentie par l'objet auquel il est attaché, il calcule la vitesse angulaire en degrés par seconde. Dans notre cas il va servir à atténuer les bruits et à déterminer l'angle lorsqu'il y a des mouvements brusques ce qui est difficile à mesurer par l'accéléromètre

Accéléromètre

Lors de l'expérimentation, on a découvert que le gyroscope seul ne peut pas fournir la valeur exacte de l'angle d'inclinaison. On a donc ajouté un autre capteur MEMS pour estimer cette dernière.

L'accéléromètre est un élément important dans la conception d'un système d'acquisition de données, car il permet de mesurer l'accélération linéaire d'un objet mobile selon 3 axes orthogonaux(X, Y et Z). À partir de cette mesure, l'utilisateur peut déduire le déplacement et la vitesse de l'objet mobile. Le principe de tous les accéléromètres est basé sur la loi fondamentale de la dynamique $F = M * a$, avec (F : force, M : masse, a : accélération). Plus précisément, il consiste à l'égalité entre la force d'inertie de la masse sismique du capteur et une force de rappel appliquée à cette masse. [16]

Le principe de l'accéléromètre a permis de mettre œuvre un certains nombres d'applications, on le trouve dans les appareils mobiles comme les téléphones, les tablettes, les manettes de jeu et même dans notre voiture pour déclencher les airbags suite à un choc violant et il y a même des fabricants des ordinateurs portables prenant l'exemple "Apple" qui ont incorporé des accéléromètres dans leurs produits afin de détecter une chute soudaine de la machine et d'éteindre le disque dur d'une façon automatique pour protéger les données.

Mais en raison de forces brusques et sa sensibilité aux vibrations qui pourraient induire des erreurs, on utilise un gyroscope combiné avec l'accéléromètre pour pallier à ce problème.

MPU 6050

Puisqu'il est nécessaire de mesurer avec précision l'inclinaison du robot, on a décidé de travailler avec le composant MPU 6050 qui travaille par la technologie de MEMS (Micro-Electro-Mechanical Systems).
Le MPU 6050 est capable de mesurer l'accélération et la vitesse angulaire. Il fonctionne donc comme étant un accéléromètre et un gyroscope. Il existe différentes technologies de centrale inertielle mais l'une des plus efficaces est la centrale inertielle capacitive tout simplement parce que son rapport qualité prix est excellent.

Encodeur de moteur

[17] Un encodeur est un dispositif électromécanique qui génère un signal électrique en fonction de la position ou du déplacement de l'élément mesuré. En robotique mobile, les encodeurs rotatifs sont utilisés pour mesurer le déplacement (sens et vitesse de rotation) de chacune des roues du robot.

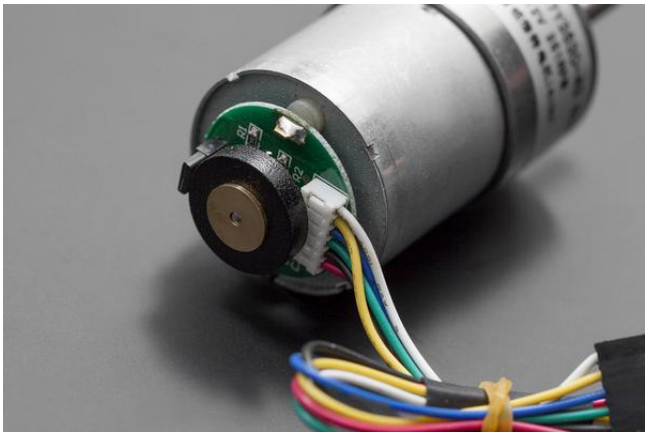

FIGURE 2.15 – Encodeur de moteur

La plupart des encodeurs pour robots mobiles utilisent des capteurs optiques mais il existe des encodeurs utilisant une information mécanique ou magnétique. Dans notre projet on a utilise un encodeur magnétique. L'idée de l'encodeur magnétique est de placer un disque alternant des pôles magnétique(Nord et Sud) devant un capteur magnétique (capteur effet Hall) et de rendre le disque solidaire de l'axe de rotation de la roue. La fréquence d'apparition des pôles Nord et Sud (ou de tout autre principe offrant un contraste suffisant) devant le capteur de Hall va indiquer la vitesse de rotation. Le schéma suivant présente le principe de fonctionnement basique de l'encodeur :

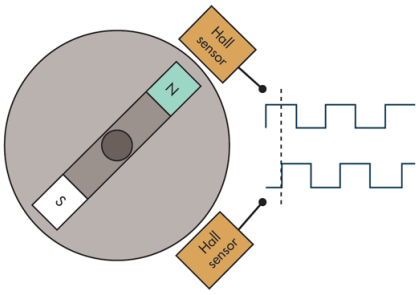

FIGURE 2.16 – Schéma de l'encodeur optique

Lorsque le disque tourne, les pôles magnétique se déplacent devant le capteur de Hall qui détecte les changement dans le champs magnétique. 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 de notre robot.

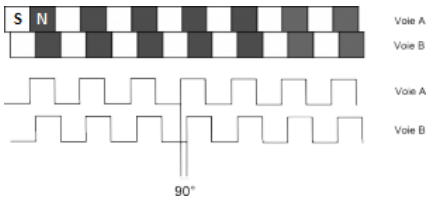

FIGURE 2.17 – Signal de l'encodeur

Capteur de distance ultrason

[18] Les capteurs ultrasons fonctionnent en mesurant le temps de retour d'une onde sonore inaudible par l'homme émise par le capteur. La vitesse du son étant à peu près stable, on en déduit la distance de l'obstacle.

FIGURE 2.18 – Capteur ultrason

Les capteurs ultrasons ont souvent la forme d'une paire d'yeux car il y a deux parties essentielles :
- L'émetteur
- Le récepteur

L'émetteur émet un son à une fréquence définie (généralement autour de 40 kHz) et le récepteur collecte le son répercuté.

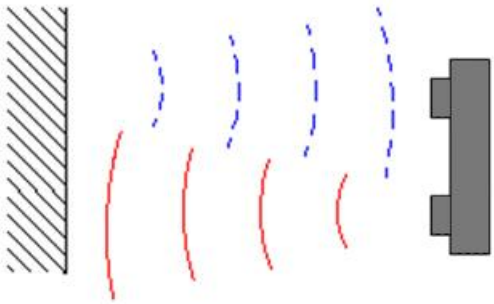

FIGURE 2.19 – La forme typique de faisceau d'ultrasons

L'utilisation du capteur ultrason pour mesurer la distance L qui nous permet de calculer l'angle de l'inclinaison du plan α (figure 2.20).

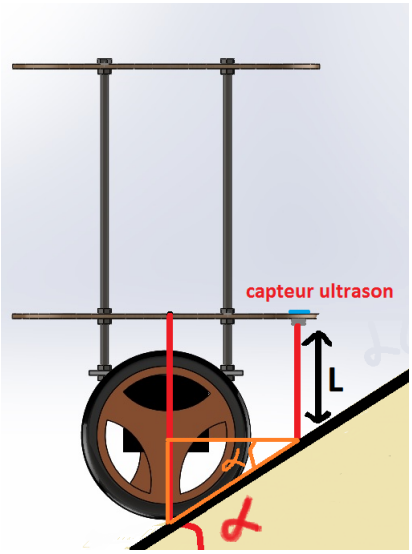

FIGURE 2.20 – Calcul de l'angle d'inclinaison du plan

2.3.6 Communication

Signal PWM

Un signal à modulation de largeur d'impulsion (PWM) est une méthode qui permet de générer un signal analogique en utilisant une source numérique. Un signal PWM est constitué de deux composantes principales qui définissent son comportement : le rapport cyclique et la fréquence. Le rapport cyclique décrit la durée pendant laquelle le signal est à l'état haut (actif) en pourcentage de la durée d'un cycle complet. La fréquence détermine la vitesse à laquelle le PWM effectue un cycle (par exemple, 1000 Hz serait 1000 cycles par seconde) et par conséquent à quelle vitesse il passe de l'état haut à l'état bas et vice versa. En changeant l'état d'un signal numérique suffisamment rapidement, et avec un certain rapport cyclique, la sortie donnera l'apparence de se comporter comme un signal analogique à tension constante lorsqu'elle alimente des périphériques. Les signaux PWM sont utilisés dans une grande gamme d'applications de contrôle. Ils sont principalement utilisés pour contrôler des moteurs DC mais peuvent également servir à contrôler des vannes, des pompes, des systèmes hydrauliques et d'autres pièces mécaniques. La fréquence nécessaire pour le signal PWM dépend de l'application et du temps de réponse du système alimenté.[19]

La fréquence du signal PWM de Arduino est 980 Hz et le rapport cyclique est 0-255 tel que 255 est 100%.[13]

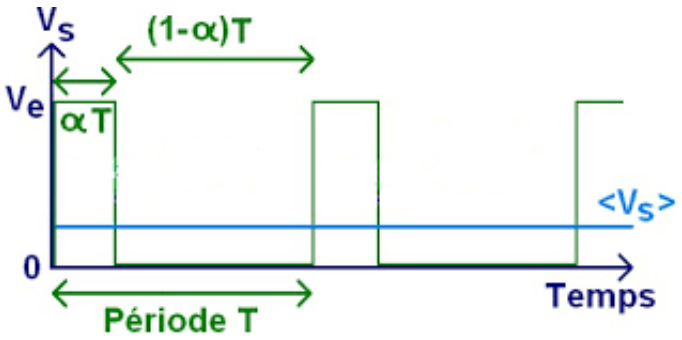

FIGURE 2.21 – Signal PWM

Bus I2C

Le bus I2C (Inter Integrated Circuit) fait partie des bus série. Il permet d'effectuer et de garantir la communication entre deux composants électroniques très divers grâce à trois fils :
- Un signal de données (SDA), généré par un maître ou un esclave.
- Un signal d'horloge (SCL), généré par le maître.
- un signal de référence électrique (masse).

Le bus I2C permet cependant des échanges de données à la vitesse de 100 kbits par seconde et peut même atteindre 400 kbits par second pour les nouvelles technologies.[7]

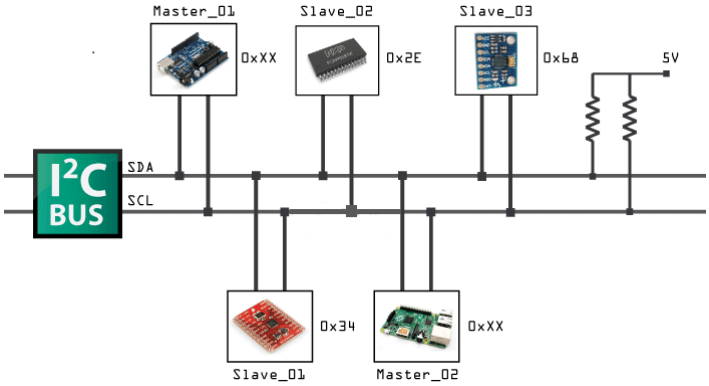

FIGURE 2.22 – Communication entre les composants électroniques par I2C

La communication sur le bus est faite de la manière suivante :
- Le bus d'adresse sera envoyé par le maître à l'esclave.
- L'esclave qui reconnaît son adresse répond par un signal de confirmation ensuite le maître continu la procédure de communication.
- Les transactions seront confirmées par un bit d'acquittement (ACK).

Pour transmettre des données sur le bus I2C, il faut surveiller deux conditions particulières :la condition de départ et la condition d'arrêt.[7]

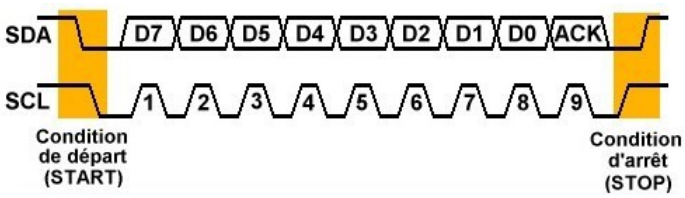

FIGURE 2.23 – Signal I2C

SDA et SCL sont au repos, c'est à dire à l'état haut. Si c'est le cas, le circuit indique qu'il prend le contrôle du bus en mettant la ligne SDA à 0 et en conservant SCL à 1.
A partir de ce moment, les autres circuits ont l'information que le bus est occupé et ils ne devraient pas tenter d'en prendre contrôle. Le circuit qui vient de prendre le contrôle du bus en devient le maître. C'est lui qui génère le signal d'horloge, quel que soit le sens de transfert des données. La condition d'arrêt est représentée par le passage de SDA à 1 et en conservant SCL à 1.[7]

Bluetooth

Concernant notre maquette expérimentale, la communication est réalisée entre le PC (qui sert à fournir les consignes au robot et à rapatrier les grandeurs mesurées) et le microcontrôleur Arduino (qui exécute le programme de commande et applique les consignes de référence) et aussi entre système Andriode (smartphone) et Arduino. La liaison Bluetooth est un choix idéal puisque c'est une liaison sans fil qui permet la liberté de mouvement de robot et transmettre les données avec PC.
Ceci est réalisé à l'aide du module hc-05 du coté robot et d'une carte Bluetooth et Matlab du coté PC Windows.

2.3.7 Ardumotive BT controller

Ardumotive BT controller est une application Andriod (smart phone), qui permet de connecter le smart phone à Arduino par Bluetooth et d'envoyer des consignes. L'application contient des boutons de directions et de fonctions, chaque bouton envoie une consigne à Arduino qui applique l'instruction équivalente à cette consigne. La figure 2.24 illustre l'interface de l'application.

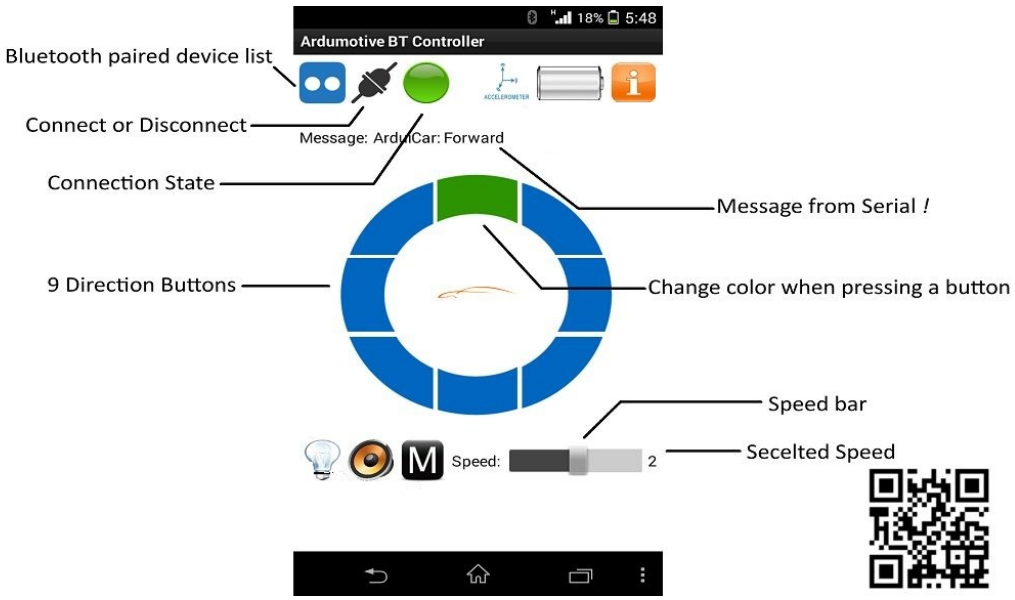

FIGURE 2.24 – L'interface de l'application

2.4 Filtre complémentaire

Le filtre complémentaire sert à fusionner les capteurs pour obtenir des informations d'attitude fiables. Pour supprimer l'erreur des données initiale, les données de l'accéléromètre sont passées à travers le filtre passe-bas et les données du gyroscope sont passées à travers le filtre passe-haut. La fonction de transfert pour l'entrée passant par le filtre passe-bas est $\frac{1}{1+TS}$ et le filtre passe-haut est $\frac{TS}{1+TS}$. Ensuite, les données sont intégrées. La moyenne pondérée est mesurée par l'équation (2.1). Le gain total des deux filtres est de 1.[3]

$$G1(s) + G2(s) = 1 \tag{2.1}$$

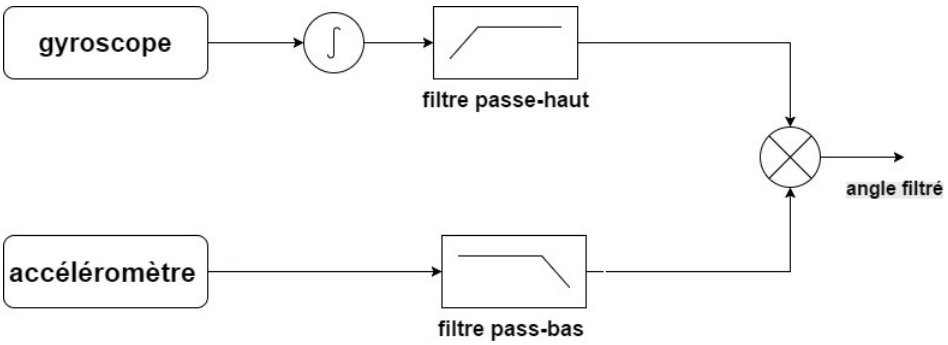

FIGURE 2.25 – La structure du filtre complémentaire

En termes de fonction de transfert, l'équation du filtre complémentaire peut être écrite comme 2.2.

$$\psi(final) = \frac{\psi_{acc}}{1+TS} + \left[\frac{T}{S(1+TS)} * \frac{1}{S}\right] \dot{\psi}_{gyro} = \frac{\psi_{acc} + T\dot{\psi}_{gyro}}{1+TS} \tag{2.2}$$

Où, T = Fréquence de coupure du filtre et ψ_{acc} = Entrée des données de l'accéléromètre.
L'angle calculé du gyroscope après avoir traversé un intégrateur est $\frac{\dot{\psi}_{gyro}}{S}$.
L'équation finale est donnée par :
- premier ordre :

$$\psi_n = c[\psi_{n-1} + \dot{\psi}gyro_n] + (1-c)\psi acc_n \tag{2.3}$$

Où,$c = \frac{T/t}{1+T/t}$ et t= le temps de l'intervalle.

- deuxième ordre :

$$\psi_{final} = \frac{1}{S}\left[\dot{\psi}_{gyro} * (\psi_{final} - \psi_{Acc}) * (K_p + \frac{K_i}{S})\right] \tag{2.4}$$

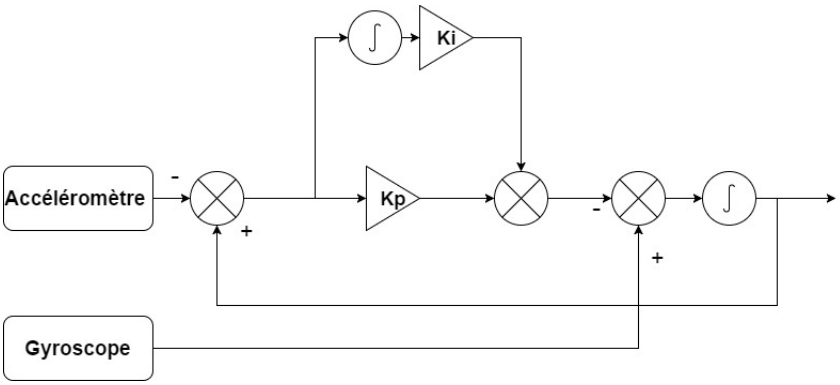

FIGURE 2.26 – La structure du filtre complémentaire 2eme ordre

On a appliqué les deux algorithmes sur les données du gyroscope/Accéléromètre MPU6050. les résultats sont représentés dans les figures 2.27, 2.28.

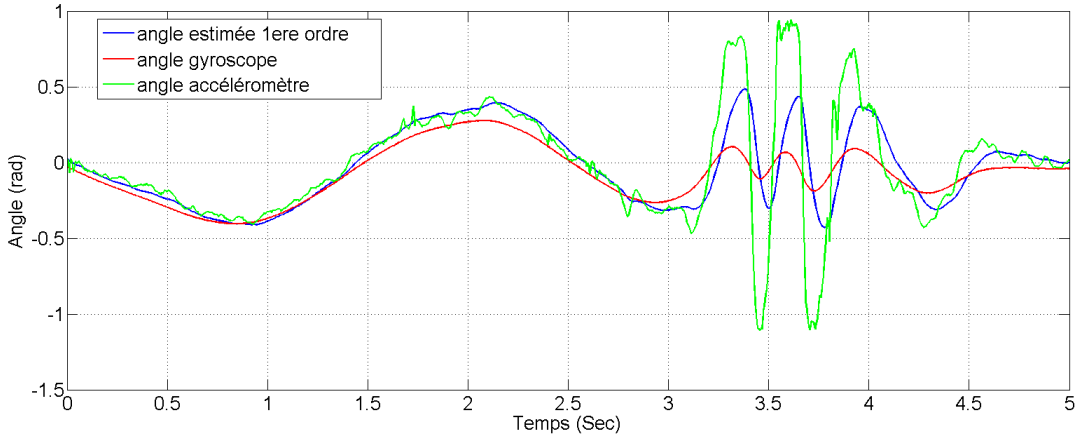

FIGURE 2.27 – Le résultat du filtre complémentaire 1 èr ordre

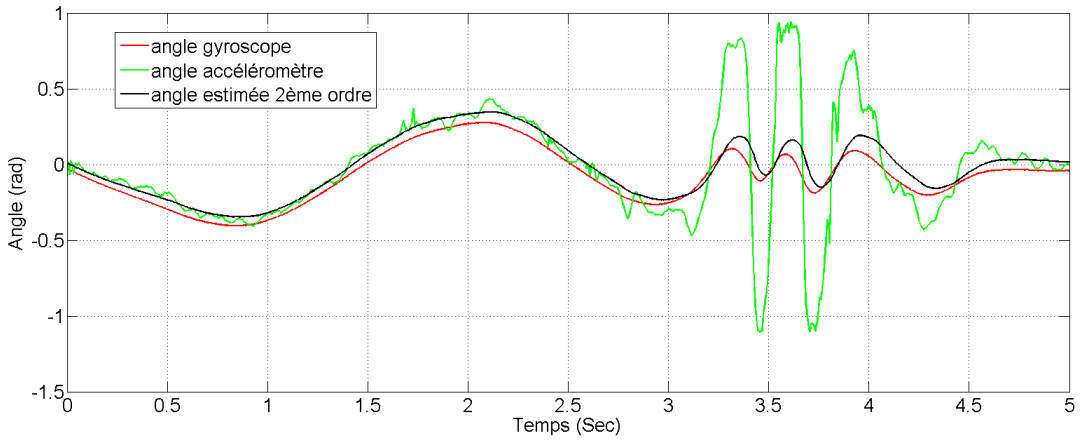

FIGURE 2.28 – Le résultat du filtre complémentaire 2 ème ordre

A partir des résultats, on peut vérifier que le filtre complémentaire a rejeté (pour les deux ordres) les signaux de hautes fréquences de l'accéléromètre et que l'angle estimée prend la valeur de l'accéléromètre pou les bases fréquences où il y a un erreur statique dans l'angle de gyroscope. On a aussi trouvé que l'angle estimé par le filtre de premier ordre comporte plus d'oscillations que celui estimé par le filtre de deuxième ordre et que plus on augmente le gain de dernier pour diminuer les oscillations, l'erreur statique augmente. Tandis que pour le filtre de deuxième ordre, l'angle estimé suit les indications dynamiques du gyroscope et les valeurs statiques de l'accéléromètre. alors on peut conclure que l'angle estimé par le filtre du deuxième ordre est le plus proche de l'angle réel.

2.5 Conclusion

On a développé ce chapitre de façon à montrer le système mécanique et électrique du projet et expliquer les différents composants utilisés lors de la construction du robot en passant par les protocoles de communication utilisé et le filtre complémentaire.

Chapitre 3

Lois de commande et simulation

Chapitre 3

Lois de commande et simulation

3.1 Introdution

Dans ce chapitre on va développer trois lois de commande PID, LQR et commande par mode glissant pour deux objectif principaux : le premier c'est la stabilisation du robot balanceur à l'origine. Le deuxième objectif est la poursuite d'une trajectoire référence.

3.2 commandabilité et observabilité

Avant de présenter les lois de commande du système à régler, nous devons montrer que le système à étudier est commandable et observable.

3.2.1 Commandabilité

On dit qu'un système est commandable à l'instant $t0$ s'il est possible de transférer le système à partir de tout état initial $x(t0)$ à tout autre état dans un intervalle de temps fini au moyen d'un vecteur de contrôle sans contrainte.[2]

Le critère de Kalman établit qu'un système est commandable si et seulement si le rang de la matrice de commandabilité ζ est égal à l'ordre du système. Par conséquent, il faut que le déterminant de la matrice de commandabilité soit différent de zéro.[2]

$$\zeta = [B\ AB\ ...\ A^{n-1}B]$$

Où n est l'ordre du système à régler.

A l'aide de Matlab, on calcule la matrice de commandabilité ainsi que son rang, ce calcul a montré qu'elle est de rang complet, cela signifie que le système est commandable.

3.2.2 Observabilité

On dit qu'un système est observable à l'instant $t0$ si, avec le système dans l'état $x(t0)$, il est possible de déterminer cet état à partir de l'observation de la sortie sur un intervalle de temps fini.[2]

$$\vartheta = [C\ \ CA\ \ ...\ \ CA^{n-1}]^T$$

L'observabilité est garantie si le rang de la matrice d'observabilité ϑ est égal à n.

Un calcul sur Matlab a montré que la matrice d'observabilité pour le système est de rang $n = 6$, donc le système est complètement observable.

3.3 Le calcul des états à partir des capteurs

Le but du robot balanceur est la navigation dans son terrain quand il balance, Pour satisfaire ces spécifications, plusieurs variables doivent être disponibles à partir des capteurs et du contrôleur . l'angle d'inclinaison $\psi(t)$, la position $x(t)$, et l'orientation $\delta(t)$ du robot doivent être mesurés et commandés.

Les capteurs utilisés dans le robot ne donnent pas ces variables directement mais on peut les calculer.

Par le filtre complémentaire on peut fusionner les données du gyroscope et de l'accéléromètre pour calculer l'angle d'inclinaison $\psi(t)$.

Les encodeurs du moteurs mesurent la position angulaire de l'arbre de boite de réduction pour chaque moteur θ_{ol} *et* θ_{or}.

A partir l'équation (1.3) on peut calculer la position angulaire de chaque roue θ_l *et* θ_r.

$$\theta_r = \theta or + \psi$$

$$\theta_l = \theta ol + \psi$$

et à partir l'équation (1.1) on trouve :

$$x_r = r_w \theta_r$$

$$x_l = r_w \theta_l$$

Maintenant on peur calculer la position du robot $x(t)$ et son orientation $\delta(t)$ par les équations suivantes :

$$x(t) = \frac{x_l(t) + x_r(t)}{2} \tag{3.1}$$

$$\delta(t) = \frac{x_l(t) - x_r(t)}{S} \tag{3.2}$$

Où S est la distance entre les deux roues.
la figure 3.1 montre le block diagramme de la mesure des variables du système

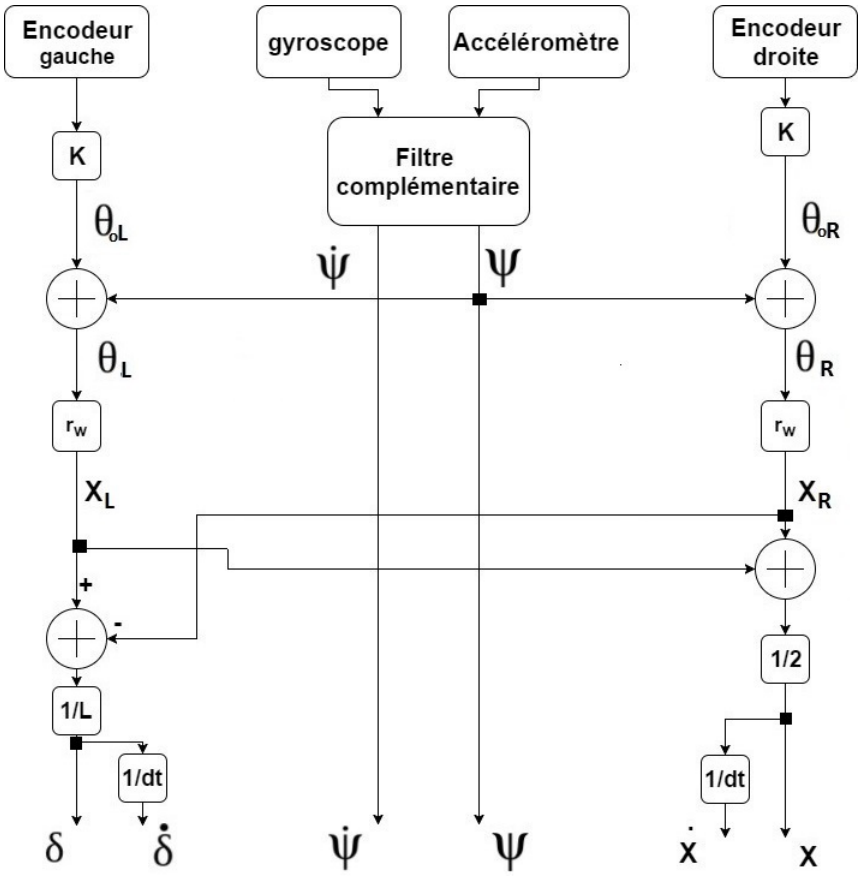

FIGURE 3.1 – Calcul des variables du système

3.4 régulateur PID

3.4.1 Synthèse du régulateur PID

Le régulateur PID (Proportional Integral Derivative Controler) est un système de contrôle en boucle fermée très largement utilisé en pratique, C'est une méthode très typique de contrôle dans l'industrie.[4] PID est composé de trois parties :
- La partie proportionnelle qui est utilisé dans le but de l'élimination d'erreur et la diminution du temps de réponse. mais une augmentation trop importante de celui-ci peut mener à une instabilité du système.
- La partie intégrale qui est utilisé pour faire la moyenne de l'erreur passée et la suppression de l'erreur en régime stationnaire
- La partie dérivable est pour prédire l'erreur supplémentaire par la variation de l'erreur passée.

La valeur finale de la commande u(t) est calculée par la somme de trois termes.[4]

$$u(t) = K_p e(t) + K_i \int e(t)dt + K_d \frac{de(t)}{dt} \tag{3.3}$$

On peut obtenir la performance de sortie souhaitée par trois gains , il existe de nombreuses méthodes de choix des trois gains pour atteindre la performance désirée.

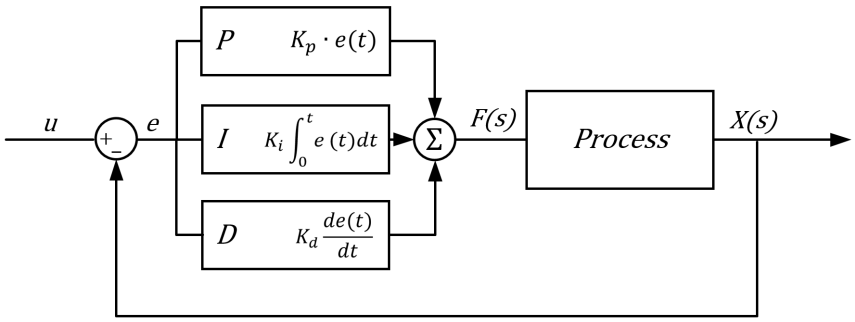

FIGURE 3.2 – Régulateur PID

3.4.2 Application du régulateur PID au robot

Pour stabiliser le robot en position vertical et commander sa position et son orientation par la commande PID, on a conçu trois régulateurs PID - régulateur de l'angle d'inclinaison $\psi(t)$, régulateur de position $x(t)$ et le régulateur de rotation $\delta(t)$, Les équations des régulateurs PID sont données comme suit :

$$U_\psi(t) = K_p e_\psi(t) + K_i \int e_\psi(t)dt + K_d \frac{de_\psi(t)}{dt}. \tag{3.4}$$

$$U_x(t) = K_p e_x(t) + K_i \int e_x(t)dt + K_d \frac{de_x(t)}{dt} \tag{3.5}$$

$$U_X(t) = U_\psi(t) + U_x(t) \tag{3.6}$$

$$Uh = K_p e_\delta(t) + K_i \int e_\delta(t)dt + K_d \frac{de_\delta(t)}{dt} \tag{3.7}$$

Où $e_\psi(t) = 0 - \psi(t)$, $e_x(t) = x_{ref}(t) - x(t)$ et $e_\delta(t) = \delta_{ref}(t) - \delta(t)$.
- x_{ref} la référence de déplacement linéaire.
- δ_{ref} la référence de déplacement angulaire.
- U_X la commande du premier système.
- U_h la commande du deuxième système.

Le dynamique de l'angle d'inclinaison et le dynamique de la position sont couplées l'une à l'autre donc une variation dans un paramètre d'un régulateur affecte les deux autres ce qui rend

le réglage difficile, Au contraire le dynamique de rotation est indépendante des autres. Le réglage des paramètres des régulateurs est fait par essais et simulation sous Simulink.

La structure de commande est montrée dans la figure 3.3 :

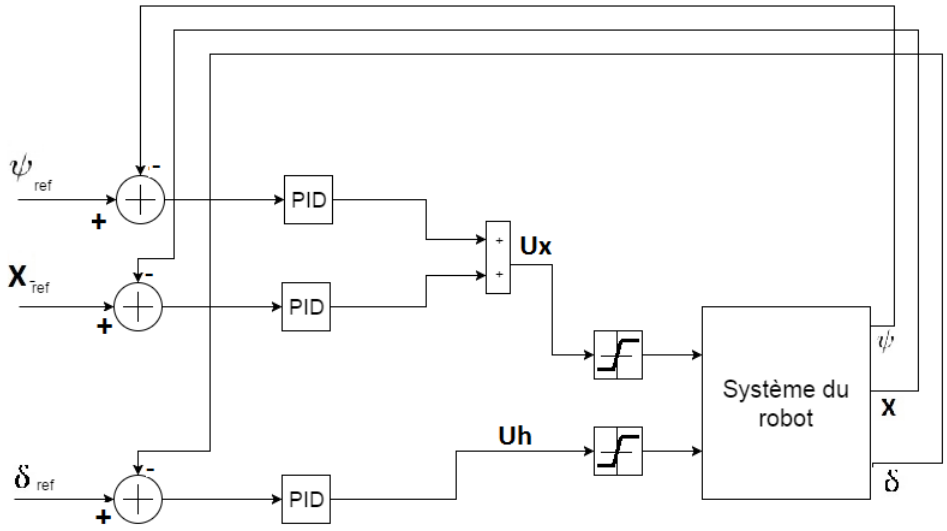

FIGURE 3.3 – Structure du régulateur PID

3.4.3 Simulation

A l'aide du logiciel MATLAB / SIMULINK, on a simulé les commandes du robot en deux parties, dans la première parties on a étudié la stabilité du robot avec un état initial de l'angle d'inclinaison ψ non nul. Dans la deuxième partie on a étudié la poursuite de trajectoire.

La figure 3.4 montre l'angle d'inclinaison, la position linéaire et leurs vitesses pendant la stabilisation à la position verticale. Avec une valeur initiale constante de l'angle d'inclinaison $\psi_0 = 0.35rad$). Le robot avance $0.08m$ pour se stabiliser dans la position vertical avec une convergence relativement rapide $0.7s$. La convergence de la position x au point initial est lente avec des oscillations que disparaissent après $4.3s$.

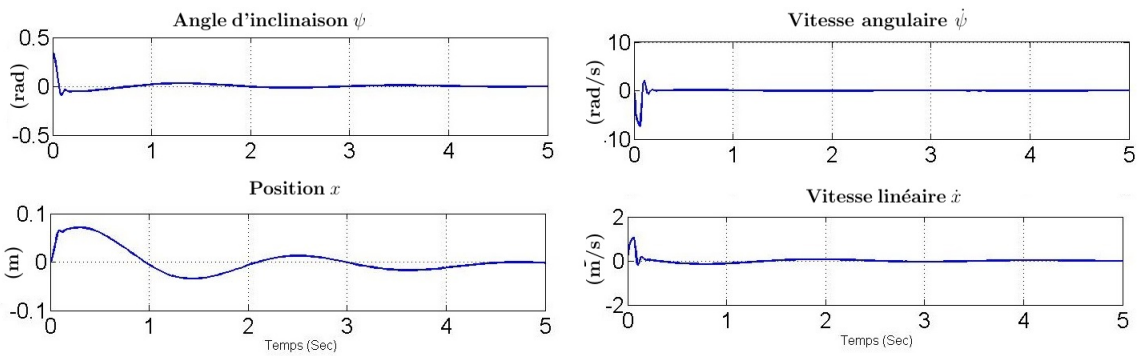

FIGURE 3.4 – Équilibration par le régulateur PID

La figure 3.5 montre la réponse du système avec les consignes $X_{ref} = 1m$ *et* $\delta_{ref} = \frac{\pi}{2}$. On observe qu'avant d'exécuter une marche en avant, le robot est contraint de préalablement reculer afin d'incliner le corps dans la direction de translation, autorisant ensuite le déploiement d'un couple accélérateur, ce parce que le dynamique de x a un zéro positif (phase non minimale). On remarque que la position x a un dépassement de consigne par $0.2m$ dans le régime transitoire et elle prend la valeur de consigne a l'instant $4.1s$.

La poursuite de référence de l'angle de rotation δ est atteinte sans dépassement et avec un temps de réponse acceptable égal à $1.2s$.

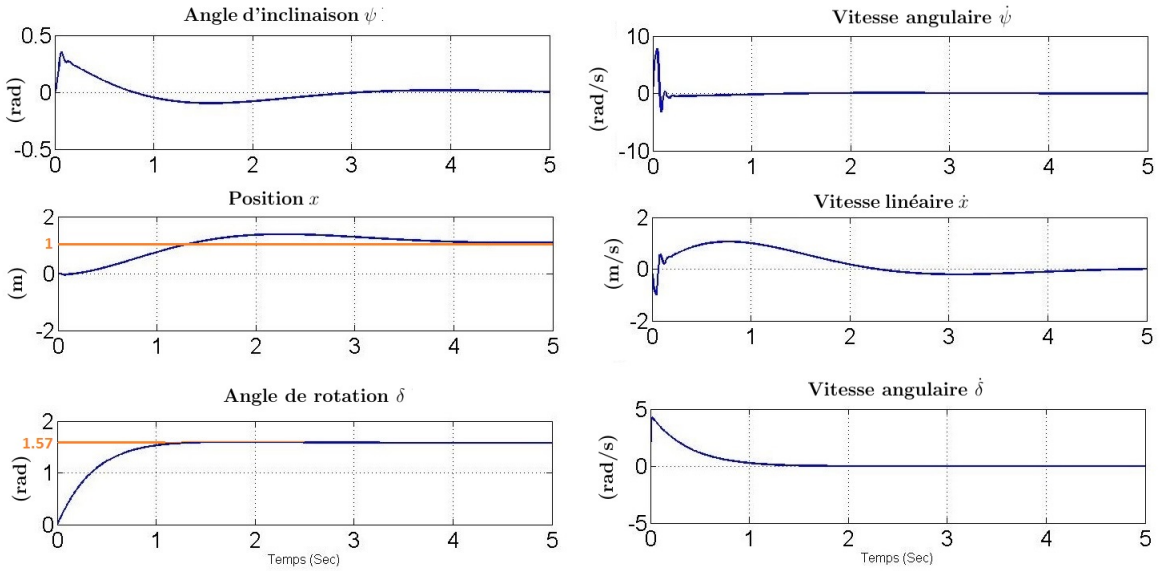

FIGURE 3.5 – Simulation du système avec régulateur PID

3.5 Commande LQR

3.5.1 Synthèse de la commande LQR

Ce type de régulateur utilise un modèle d'état linéaire du système. Un régulateur quadratique linéaire optimal (LQR) estime le gain des contrôleurs en utilisant le modèle du système. Le but du contrôleur est de minimiser la fonction de coût[5] :

$$J = \frac{1}{2}\int_0^\infty (X^T(t)QX(t) + u^T(t)Ru(t))dt \tag{3.8}$$

Les matrices R et Q, équilibrent l'importance relative des entrées et des états dans la fonction de coût (J) étant optimisé avec une condition que les éléments dans Les matrices Q et R sont des valeurs positives. La taille de la matrice Q dépend de la taille de la matrice d'état du système et la matrice R dépend du nombre d'entrées de contrôle du système.[5]

Le gain $K = R^{-1}B^TP$ de la commande $u = -KX$ est obtenu par application de l'indice de performance de l'équation 3.8, et en utilisant la solution de l'équation algébrique de Riccati donnée dans l'équation 3.9.[5]

$$A^TP + PA - PBR^{-1}B^TP + Q = 0 \tag{3.9}$$

3.5.2 Choix des pondérations

Il est intéressant de remarquer d'abord que la multiplication des pondérations Q et R par un même scalaire laisse inchangée le gain K. En effet, soit P solution de l'équation de Ricatti et soit le nouveau problème basé sur les pondérations $\hat{Q} = \lambda Q$ et $\hat{R} = \lambda R$ On vérifie que $\hat{P} = \lambda P$ est solution de l'équation de Riccati correspondante. En effet :

$$\hat{K} = -\hat{R}^{-1}B^T\hat{P} = R^{-1}B^TP = K \tag{3.10}$$

Sans restriction, les pondérations peuvent être choisies symétriques. Elles sont généralement choisies diagonales. Un valeur élevé de pondération de Q indique l'importance de cet état par rapport aux autres. La valeur élevé de R indique que moins d'énergie de commande des moteurs est utilisée pour équilibrer le robot.

3.5.3 Application de la commande LQR sur le robot

A partir de Matlab, on peut résoudre l'équation de Ricatti et calculer le gain K par différent valeurs de Q et R les matrices de pondération.

Les matrices Q et R sont de la forme :

$$Q = \begin{bmatrix} a & 0 & 0 & 0 & 0 & 0 \\ 0 & b & 0 & 0 & 0 & 0 \\ 0 & 0 & c & 0 & 0 & 0 \\ 0 & 0 & 0 & d & 0 & 0 \\ 0 & 0 & 0 & 0 & e & 0 \\ 0 & 0 & 0 & 0 & 0 & f \end{bmatrix} \qquad R = \begin{bmatrix} R_1 & 0 \\ 0 & R_2 \end{bmatrix}$$

Où les valeurs a, b, c, d, e, f sont les pondérations des états $\psi, \dot{\psi}, x, \dot{x}, \delta, \dot{\delta}$ et R_1, R_2 sont des pondérations de commandes U_x, U_h ,respectivement.

Comme on a dit précédemment les deux sous-systèmes sont indépendants donc le changement des valeurs e, f, R_2 n'affecte pas la réponse du premier sous-système et vice versa.

Les valeurs de la matrice Q sont ajustées pour obtenir la réponse désirée du système.

L'objectif principal de la commande est que tous les états poursuivent les consignes (la consigne est null pour ψ et $\dot{\psi}$) en le plus court temps possible.

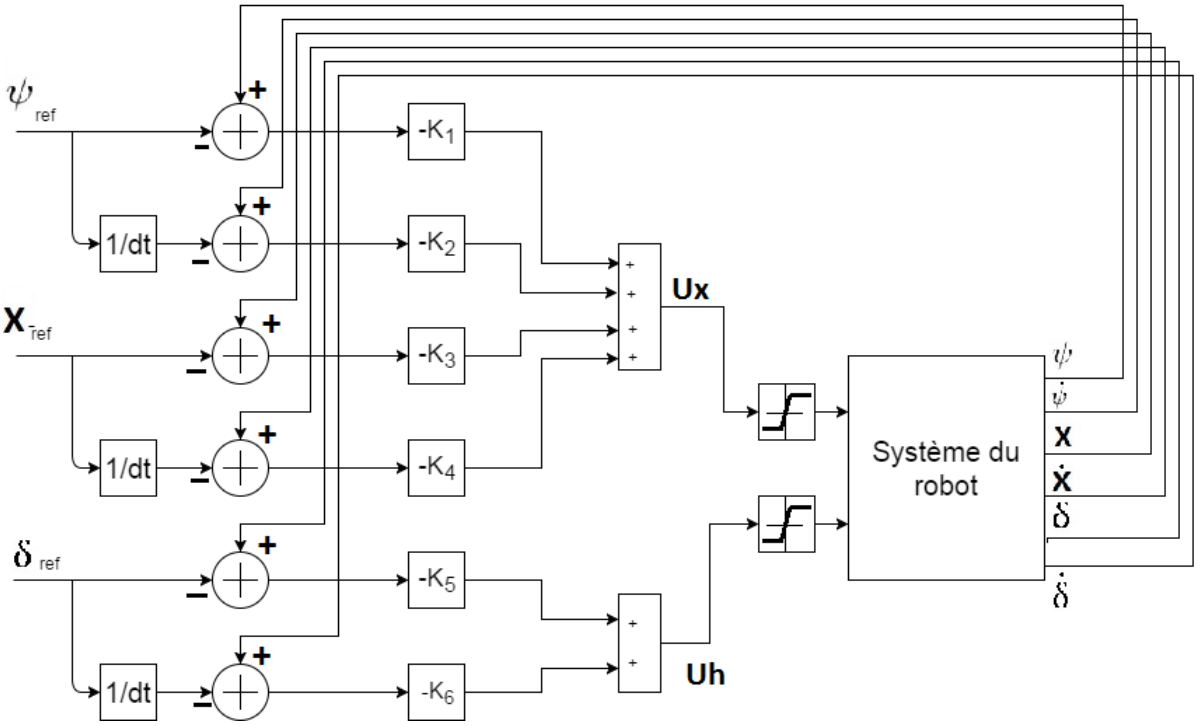

FIGURE 3.6 – Structure de la commande LQR

3.5.4 Simulation

On reprend les mêmes conditions dans la simulation du régulateur PID pour le régulateur LQR :

Premièrement, on simule le système avec un angle d'inclinaison initial $\psi_0 = 0.2rad$. On peut voir la réponse du système en figure 3.7, le robot se stabilise après $0.8s$ avec un déplacement $0.05m$ du point initiale. On remarque aussi que le robot retourne à sa position initial après $2s$ qu'est moins que le temps de réponse du régulateur PID.

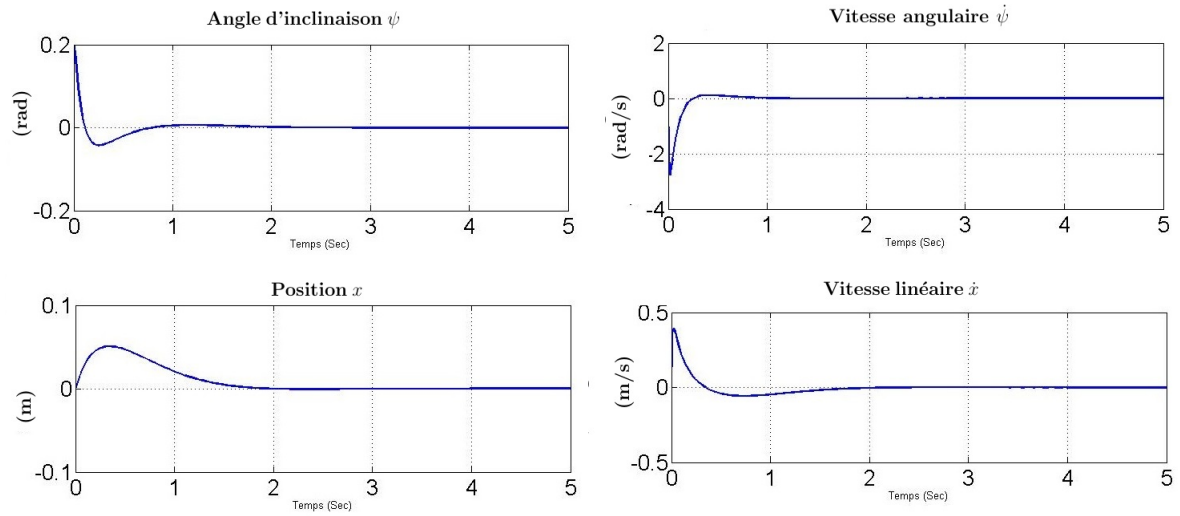

FIGURE 3.7 – Équilibration de robot par la commande LQR

la figure 3.7 montre le résultat de la simulation de poursuite par le robot de la référence. On prend une référence $x_{ref} = 1m$ *et* $\delta_{ref} = \frac{\pi}{2}$, le robot poursuit les deux référence sans oscillation , on peut voir aussi que le temps de réponse est moins que le temps de réponse du régulateur PID , soit $2s$ pour la position et $1s$ pour l'angle de rotation. Le robot se déplace un peu au sens inverse de la consigne afin d'incliner le corps dans la direction de celle-ci, ce parce que le dynamique de x a un zéro positif.

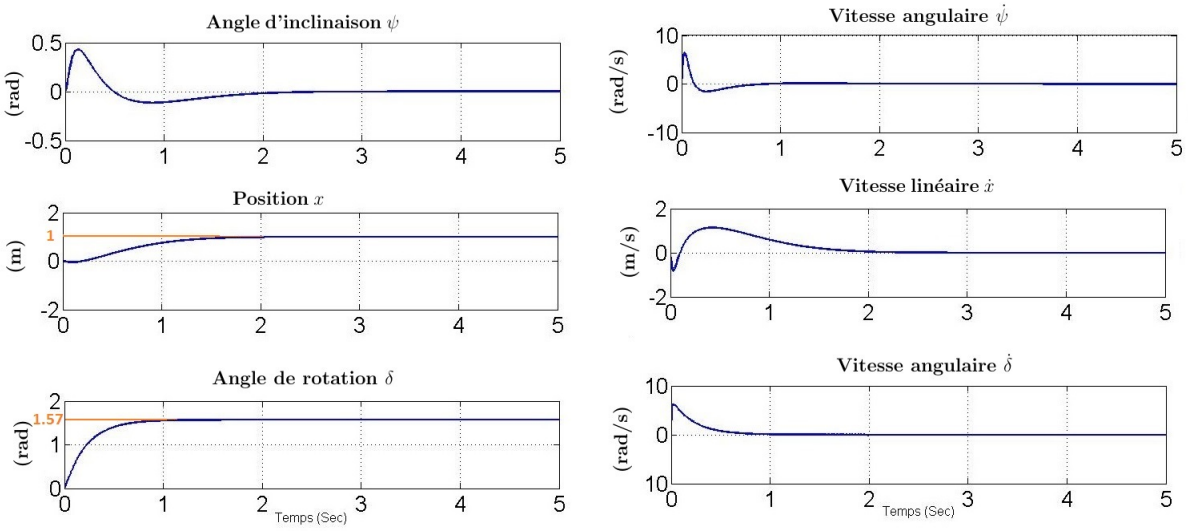

FIGURE 3.8 – Poursuite de la consigne par la commande LQR

3.6 Commande par modes glissants

La commande par mode glissant a connu un essor considérable durant les dernières décennies.Ceci est dû principalement à la propriété de convergence rapide et en temps fini des erreurs, ainsi, que la grande robustesse par rapport aux erreurs de modélisation et certains types de perturbations extérieures.[8]

La commande par mode glissant est une commande à structure variable pouvant changer de structure et commutant entre deux valeurs suivant une logique de commutation bien spécifique $s(x)$. Le principe de la commande par modes glissants est de contraindre le système à atteindre une surface donnée appelée surface de glissement et d'y demeurer jusqu'à l'équilibre. Cette commande se fait en deux étapes : la convergence vers la surface et ensuite le glissement le long de celle-ci figure 3.9.[8]

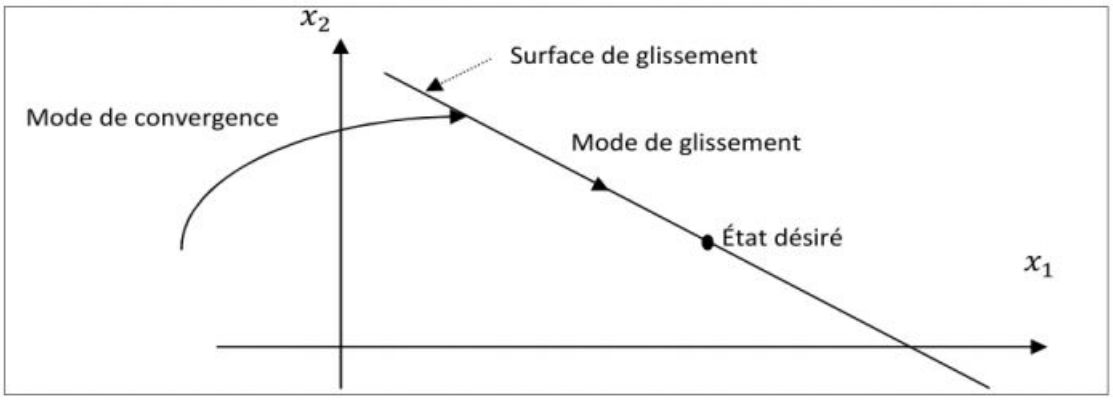

FIGURE 3.9 – Surface de glissement

3.6.1 Synthèse de la loi de commande

La synthèse de la commande par modes glissants se fait en trois étapes :
- choix de la surface de glissement
- Établir la condition de convergence
- déterminer la loi de commande qui permet d'atteindre la surface et d'y demeurer.

3.6.2 Choix de la surface de glissement

Soit le système décrit par l'équation différentielle suivante 3.11 :

$$x^n = f(x,t)x + g(x,t)u \tag{3.11}$$

où f et g sont des fonctions non linéaires, g est supposée inversible.u : L'entrée du système. x :état du système. Soit x_{ref} le référence et e l'erreur de poursuite définie par :

$$e = x - x_d \tag{3.12}$$

La formule générale de la surface de glissement est définie en fonction de l'ordre du système comme suit :

$$s(x) = (\frac{\partial}{\partial t} + \lambda)^{n-1} e(x) \tag{3.13}$$

Où n : le degré relatif du système par rapport à la sortie y(t).Il représente le nombre minimum de fois qu'il faut dériver la sortie y(t) par rapport au temps, pour y voir apparaître l'entrée.

3.6.3 Condition d'existence du mode de glissement

Le choix de la fonction de glissement étant fait, la deuxième étape consiste à concevoir une loi de commande qui puisse amener le vecteur d'état à converger vers la surface et y demeurer (S=O). Pour cela, il faut que la loi de commande soit conçue de telle manière à ce que S soit attractive.

Une condition nécessaire et suffisante, appelée condition d'attractivité, pour qu'une variable de glissement $s(x,t)$ tende vers 0 est que la dérivée temporelle de S soit définie négative :

$$S\dot{S} < 0 \tag{3.14}$$

3.6.4 Calcul de la commande

Dans notre cas, la méthode choisie est celle de la commande équivalente, schématisée sur la figure 3.10. La commande équivalente est une fonction continue qui sert à maintenir la variable

à contrôler sur la surface de glissement $S = 0$. Elle est obtenue grâce aux conditions d'invariance de la surface :

$$S = 0$$

$$\dot{S} = 0$$

Où u_{eq} est déduite de la relation $\dot{S} = 0$

Physiquement la commande équivalente présente la valeur moyenne de la commande u. Cependant, cette commande ne force pas les trajectoires du système à converger vers la surface de glissement. Ainsi, la commande u est la somme de la commande équivalente et d'une composante discontinue (figure 3.10) assurant une convergence et un régime glissant.

$$u = u_{eq} + u_d \quad avec \quad u_d = -\alpha sign(s)$$

$$u_{eq}(x,t) = -\left[\left(\frac{\Delta S}{\Delta x}\right)^T g(x,t)\right]^{-1}\left\{\left(\frac{\Delta S}{\Delta x}\right)^T f(x,t) + \frac{\Delta S}{\Delta t}\right\}$$

α est une constante positive, sign est la fonction signe et u_d est la commande discontinue

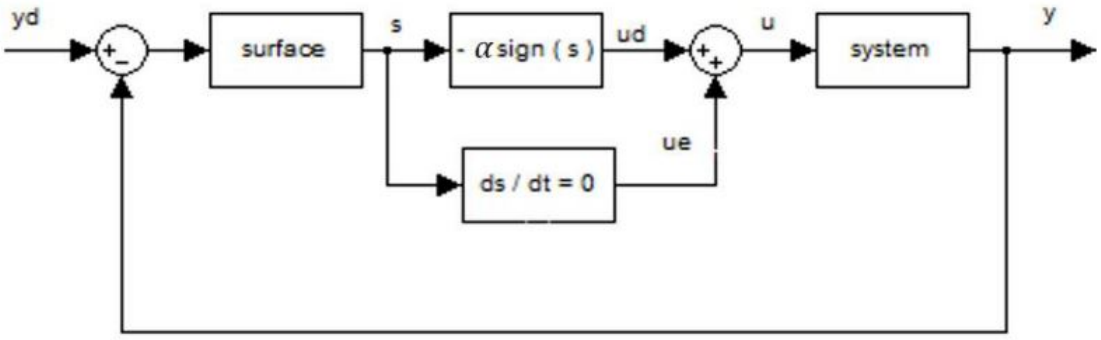

FIGURE 3.10 – Structure de la commande par mode glissant

3.6.5 Le broutement (chattering)

Un régime glissant idéal requiert une commande pouvant commuter à une fréquence infinie. Ainsi, durant le régime glissant, les discontinuités appliquées à la commande peuvent entraîner un phénomène de broutement, appelé réticence ou "chattering" en anglais. Celui-ci se caractérise par de fortes oscillations des trajectoires du système autour de la surface de glissement (figure (4.9)). Les principales raisons à l'origine de ce phénomène sont les limitations des actionneurs ou les retards de commutation au niveau de la commande. Ces commutations détériorent la précision de la commande et peuvent s'avérer néfastes pour l'organe de commande en provoquant une détérioration prématurée des systèmes mécaniques et une élévation de température dans les systèmes électriques (perte d'énergie non négligeable).

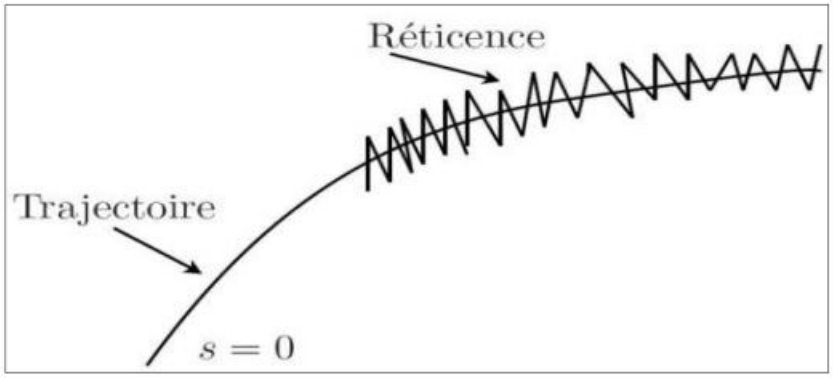

FIGURE 3.11 – Phénomène de broutement

3.6.6 Solutions pour atténuer le phénomène de réticence

Dans le but de réduire ou d'éliminer ce phénomène, de nombreuses solutions ont été proposées, comme la solution de couche limite, fuzzy sliding mode, mode glissant d'ordre supérieur, approch law, etc...

Dans ce projet on a utilisé la seule méthode de Solution de couche limite.

Cette solution, connue aussi sous le nom de " boundary layer solution", consiste à remplacer la fonction signe par une approximation continue, de type grand gain, uniquement dans un voisinage de la surface, Parmi les fonctions utilisées nous citerons la fonction de saturation

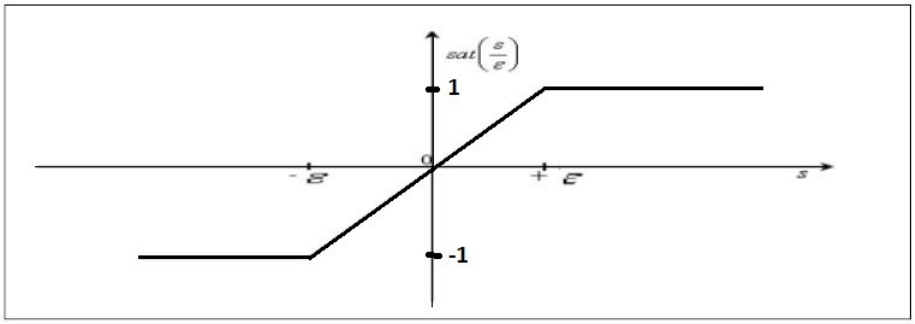

FIGURE 3.12 – Fonction de saturation

$$\begin{cases} \frac{S}{\xi} & \text{Si } |\frac{S}{\xi}| < 1 \\ sing(S) & \text{Si } |\frac{S}{\xi}| > 1 \end{cases}$$

ξ : Largeur du seuil de la fonction de saturation.

D'autres fonctions existent telles que les fonctions, $tanh(\frac{S}{\xi}), 2\Pi \arctan(\frac{S}{\xi})$....

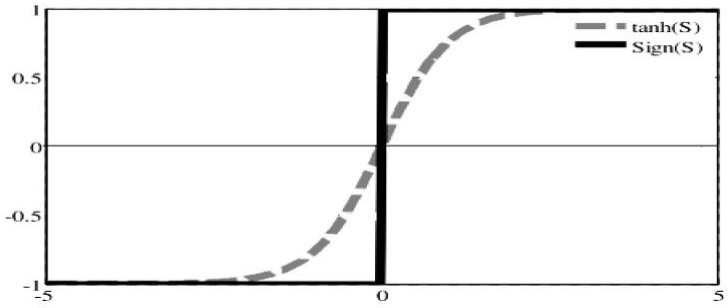

FIGURE 3.13 – Fonction tangente hyperbolique

Le système ne converge plus vers la valeur désirée, mais vers un voisinage de cette dernière dans ce cas, le système est dit en régime pseudo-glissant. Bien que cela permette d'atténuer le phénomène de réticence, la précision par rapport à l'objectif fixé, la robustesse de la commande et le temps de réponse s'en trouvent dépréciés.

Cette méthode est paramétrée par une constante positive ξ réglée pour avoir un bon compromis entre réduction du chattering et conservation de la robustesse. Dans les méthodes présentées ici, plus ξ est petit, plus l'approximation tend vers la fonction signe, et donc meilleure est la robustesse, au détriment de la réduction du chattering.

3.6.7 Application de la commande par mode glissant au robot

Le problème de poursuite de trajectoire consiste à déterminer une loi de commande $U_x(t)$ et $U_h(t)$ qui permet d'assurer la convergence de l'état $X(t)$ du système vers l'état désiré $X_{ref}(t)$.

La première étape est le choix de la surface de glissement. On a deux sous-système avec deux commandes donc on a besoin de deux surfaces S_1 et S_2 . la surface S_1 garantit la poursuit de

$\psi(t)$ et $x(t)$, la surface S_2 garantit la poursuite de l'angle de rotation $\delta(t)$.
On a définit l'erreur comme suit :

$$e_1 = \begin{pmatrix} e_\psi \\ e_x \end{pmatrix} = \begin{pmatrix} \psi(t) - 0 \\ x(t) - x_{ref}(t) \end{pmatrix} = \begin{pmatrix} x_1(t) \\ x_3(t) - x_{3ref}(t) \end{pmatrix} \tag{3.15}$$

$$e_2 = \delta(t) - \delta_{ref}(t) = x_5(t) - x_{5ref}(t) \tag{3.16}$$

La surface est définit comme suit :

$$S(x) = (\frac{\partial}{\partial t} + \lambda)^{n-1} e(x)$$

avec $\lambda > 0$
Les surfaces des systèmes sont :

$$\begin{cases} S_1(x) = \lambda_1 x_1 + \dot{x}_1 + \lambda_2(x_3 - x_{3ref}) + \dot{x}_3 - \dot{x}_{3ref} \\ S_2(x) = \lambda_3(x_5 - x_{5ref}) + \dot{x}_5 - \dot{x}_{5ref} \end{cases} \tag{3.17}$$

On a $\dot{x}_1 = x_2,\ \dot{x}_3 = x_4$ *et* $\dot{x}_5 = x_6$:

$$\begin{cases} S_1(x) = \lambda_1 x_1 + x_2 + \lambda_2(x_3 - x_{3ref}) + x_4 - \dot{x}_{3ref} \\ S_2(x) = \lambda_3(x_5 - x_{5ref}) + x_6 - \dot{x}_{5ref} \end{cases} \tag{3.18}$$

On calcule la dérivée de la surface S :

$$\begin{cases} \dot{S}_1(x) = \lambda_1 \dot{x}_1 + \dot{x}_2 + \lambda_2(\dot{x}_3 - \dot{x}_{3ref}) + \dot{x}_4 - \ddot{x}_{3ref} \\ \dot{S}_2(x) = \lambda_3(\dot{x}_5 - \dot{x}_{5ref}) + \dot{x}_5 - \ddot{x}_{5ref} \end{cases} \tag{3.19}$$

A partir du modèle d'état on trouve :

$$\begin{cases} \dot{S}_1(x) = \lambda_1 x_2 + 71.16 \sin(x_1) - 16.71 x_2 + 303.93 x_4 - 38.66 U_x + \lambda_2(x_4 - \dot{x}_{3ref}) \\ \qquad\qquad +2.17 x_2 - 39.46 x_4 + 5.02 U_x - \ddot{x}_{3ref} \\ \dot{S}_2(x) = \lambda_3(x_6 - \dot{x}_{5ref}) - 491.55 x_6 + 250.10 Uh - \ddot{x}_{5ref} \end{cases} \tag{3.20}$$

Finalement on trouve :

$$\begin{cases} \dot{S}_1(x) = f_1 + g_1 U_x \\ \dot{S}_2(x) = f_2 + g_2 U_h \end{cases} \tag{3.21}$$

avec :
$f_1 = \lambda_1 x_2 + 71.16 \sin(x_1) - 16.71 x_2 + 303.93 x_4 + \lambda_2(x_4 - \dot{x}_{3ref}) + 2.17 x_2 - 39.46 x_4 + 5.02 U_x - \ddot{x}_{3ref}$
$g_1 = -38.66 + 5.02$
$f_2 = \lambda_3(x_6 - \dot{x}_{5ref}) - 491.55 x_6 - \ddot{x}_{5ref}$
$g_2 = 250.10$

La commande par modes glissants qui assure la convergence asymptotique de l'erreur vers zéro en un temps fini est donnée par la relation suivante :

$$Ux = g_1^{-1}(-f_1 - \alpha_1 sign(S_1)) \tag{3.22}$$

$$Uh = g_2^{-1}(-f_2 - \alpha_2 sign(S_2)) \tag{3.23}$$

Les constantes λ *et* α sont choisies par le concepteur de manière à garantir la convergence de la trajectoire vers la surface de glissement.

3.6.8 Simulation

On fait les mêmes simulations sur la commande par mode glissant.

La figure 3.14 montre la réponse du système avec l'état initial $\psi_0 = 0.2rad$, Le temps de réponse est lent $1.2s$ pour stabiliser le robot et le dépassement de l'angle est petit. Pour la position linéaire on peut remarquer la position du robot pour la stabilisation est petit $0.18m$ et que le temps de convergence de la position x à la position initiale est grand $5s$.

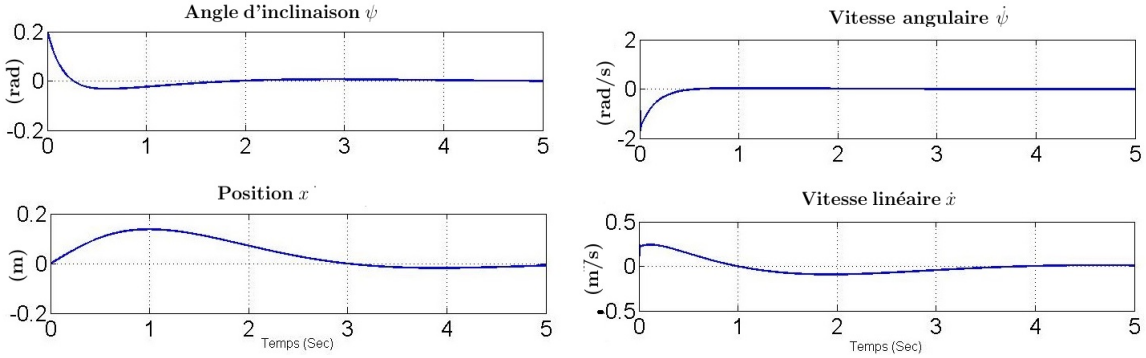

FIGURE 3.14 – Équilibration du robot avec la commande par mode glissant

La figure 3.15 représente la réponse du système avec les consigne $x_{ref} = 1m$ *et* $\delta_{ref} = \frac{\pi}{2}$. On remarque que toujours (comme les autres régulateurs) le robot se déplace un peu au sens inverse de la consigne afin d'incliner le corps dans la direction de celle-ci, ce parce que le dynamique de x a un zéro positif (phase non minimale). On remarque aussi que la convergence à la référence est très lente $5s$.

On observe que la vitesse angulaire $\dot{\delta}$ est saturée après $0.5s$ parce que elle dépend directement de la fonction de saturation dans commande par mode glissant. Le temps de réponse du système de rotation égale à $1.2s$ il est acceptable.

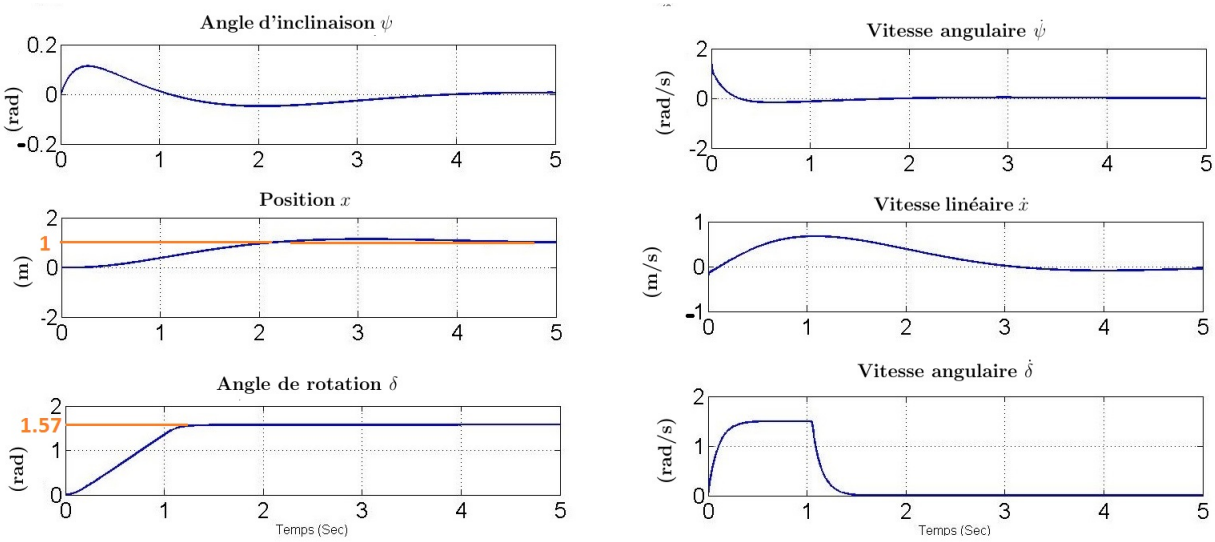

FIGURE 3.15 – Poursuit de la consigne avec la commande par mode glissant

3.7 Commande du robot en temps réel

Ce robot doit être contrôlé à distance, pour ça on a ajouté un module Bluetooth (HC-05) qui peut le connecter à un smart-phone. On a utilisé le programme "Arduimotive BT Controller" dans le smart-phone pour envoyer la consigne au robot en temps réel. L'homme peut commander les mouvements du robot (déplacement linéaire et rotation) et changer la vitesse de mouvement.

On a utilisé la commande LQR pour faire la régulation de la vitesse parce que elle est la plus performante. La figure 3.16 représente la structure de la commande de vitesse par LQR.

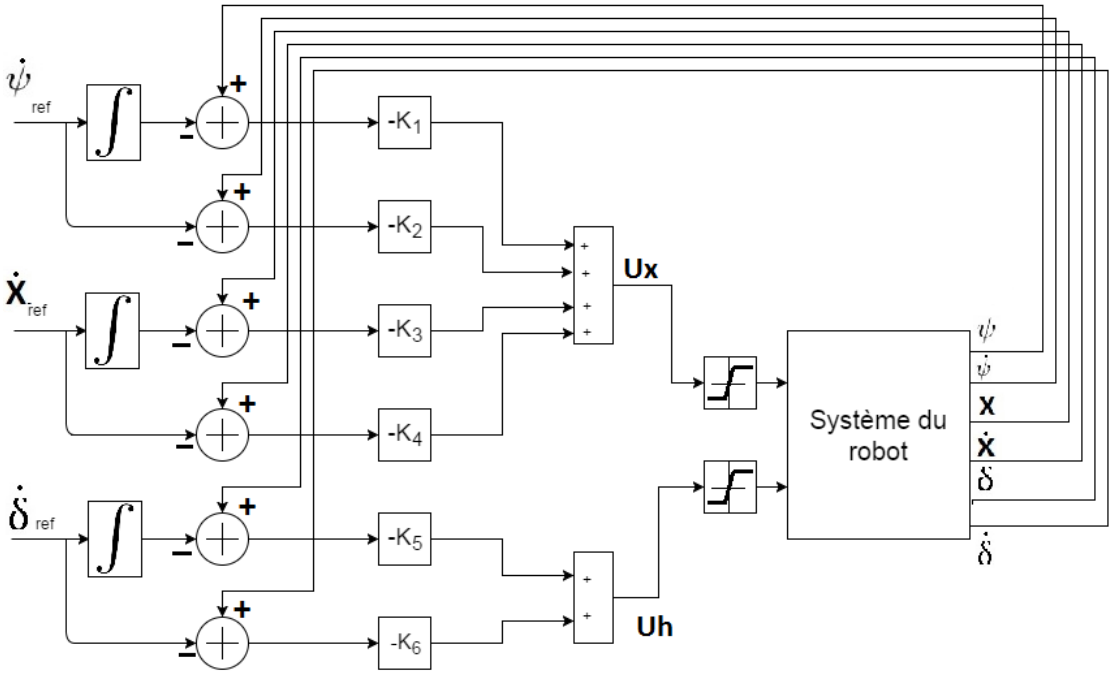

FIGURE 3.16 – Structure de la régulation de vitesse par la commande LQR

A l'aide de Simulink on a simulé le système. La figure 3.17 montre la poursuit de la consigne en vitesse linéaire. On remarque que la vitesse poursuit la consigne et qu'il y a une oscillation pendant le changement de valeur. comme on a dit précédemment la position et l'angle de l'inclinaison sont dépendant et le robot doit se déplacer au sens inverse de la consigne pour incliner le robot dans le sens de consigne.

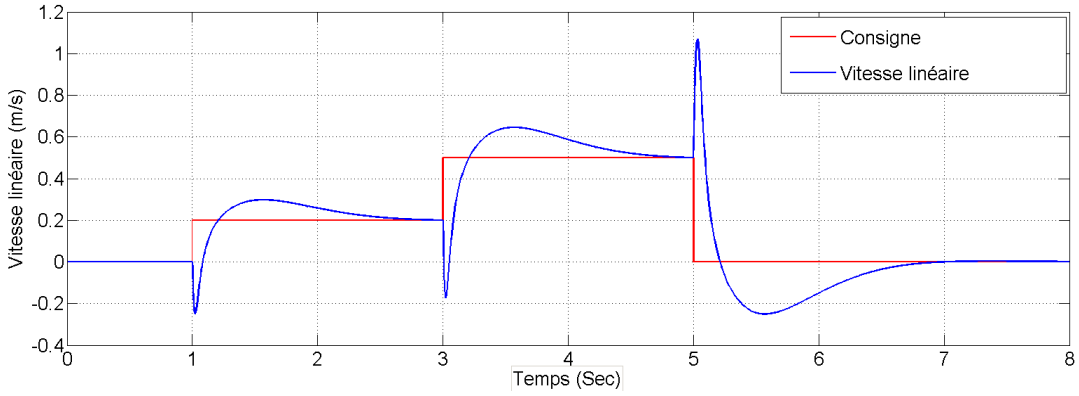

FIGURE 3.17 – Régulation de vitesse par la commande LQR

La figure 3.18 montre la simulation de la régulation de vitesse angulaire de rotation. On observe que la vitesse angulaire suit bien la consigne.

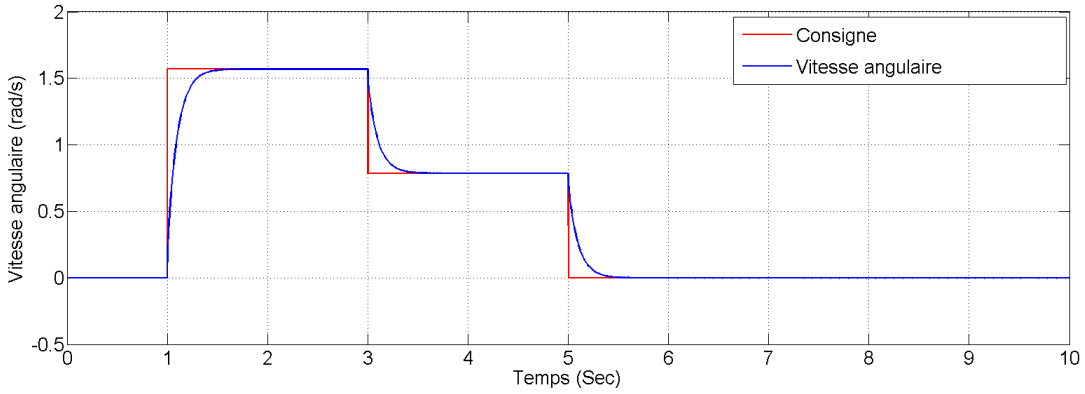

FIGURE 3.18 – Régulation de vitesse angulaire de rotation par la commande LQR

3.8 Conclusion

Dans ce chapitre, on a développé trois lois de commandes, PID, LQR et la commande par mode glissant et on a présenté les simulations de ces commandes sur le robot balanceur.

D'après les résultats obtenus, on conclue que les trois commandes réalisent les objectifs désirés, la stabilisation dans la position vertical et la poursuite de trajectoire, en particulier la commande LQR qui est la plus facile à synthétiser et plus performante que les autres commande. Dans le chapitre suivant on va implémenter ces commandes sur le robot réel et analyser les résultats.

Chapitre 4

Expériences, résultats et analyses

Chapitre 4

Expériences, résultats et analyses

4.1 Introduction

Dans ce chapitre, on va implémenter les commandes sur le robot, puis on fait quatre tests pour voir les perfemances de chaque commande, ces tests sont :
- Équilibrage du robot dans la position verticale.
- Test de la robustesse du robot.
- Test de poursuite de la consigne.

Finalement, on va tester la commande du robot en temps réel.

4.2 Acquisition des données

Avant d'exposer les résultats on va expliquer la méthode d'acquisition des données du robot. On a utilisé le module Bluetooth HC-05 pour la connexion entre Arduino et l'ordinateur. La fonction de Matlab **instrhwinfo('Bluetooth','HC-05')** crée un réseau entre Matlab et Arduino et nous permet d'envoyer ou recevoir les données. Grâce à ce réseau on peut aussi stocker toutes les données du robot ou les afficher en temps réel sur des graphes.

4.3 Tests sur le robot

Après la construction du robot et on a fait la programmation de l'Arduino, La programmation consiste en l'élaboration des codes permettant de :
• Recevoir les données de MPU 6050 et fusionner les données du gyroscope et de l'accéléromètre par un filtre complémentaire.
• Recevoir les implosions des encodeurs comme des signaux d'interruptions et calculer la position et la vitesse de chaque roue ce qui permet de calculer la position et l'orientation du robot et sa vitesse.
• Calculer les commandes u_h, u_x puis calculer les commandes pour chaque moteur u_r, u_l avec limitation $[-12v,\ 12v]$ et les envoyer comme deux signaux PWM, et quatre signaux digitaux pour les directions.
• Envoyer les données du robot par le port série (USB) ou par le Bluetooth et recevoir les consigne à partir du smart phone.

On a implémenté les lois de la commande synthétisées dans le chapitre précédent, on a essayé les commandes plusieurs fois pour obtenir les meilleures performances, qu'on va montrer dans les tests suivants.

4.3.1 Test de la stabilisation du robot

Dans ce test, on va laisser le robot libre avec des références nulles. Le robot doit être stable dans la position verticale (point d'équilibre instable) et aussi il doit rester dans sa position initiale ($x(t) = 0$).

On a appliqué ce test sur la commande LQR et la commande par mode glissant et on a obtenu les résultats suivants :

-Commande LQR :

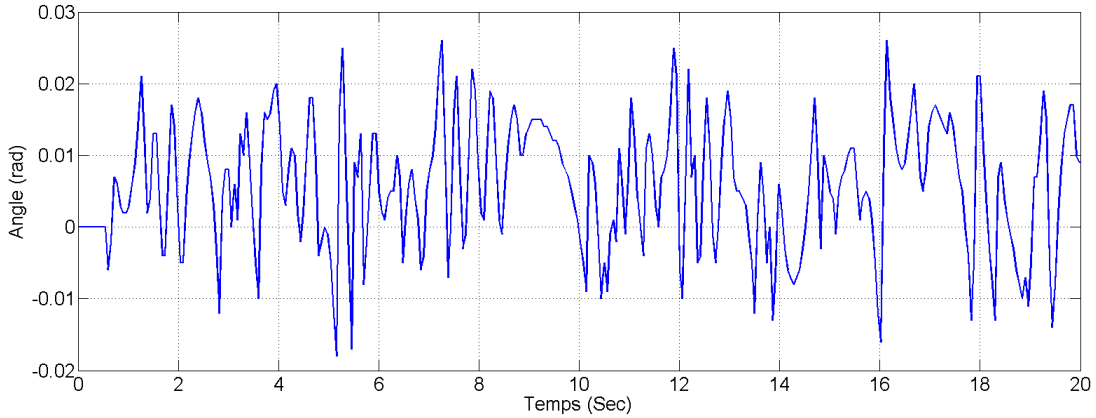

FIGURE 4.1 – Angle d'inclinaison $\psi(t)$: stabilisation du robot avec commande LQR

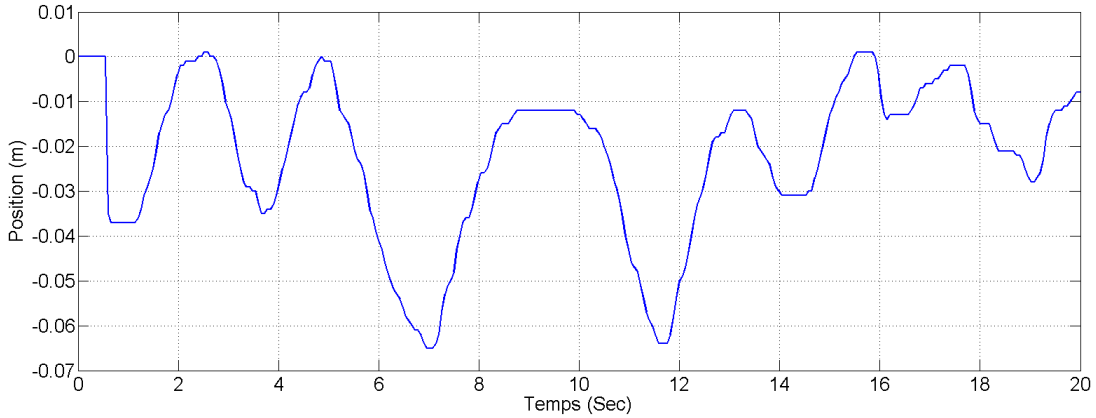

FIGURE 4.2 – Position $x(t)$: stabilisation du robot avec commande LQR

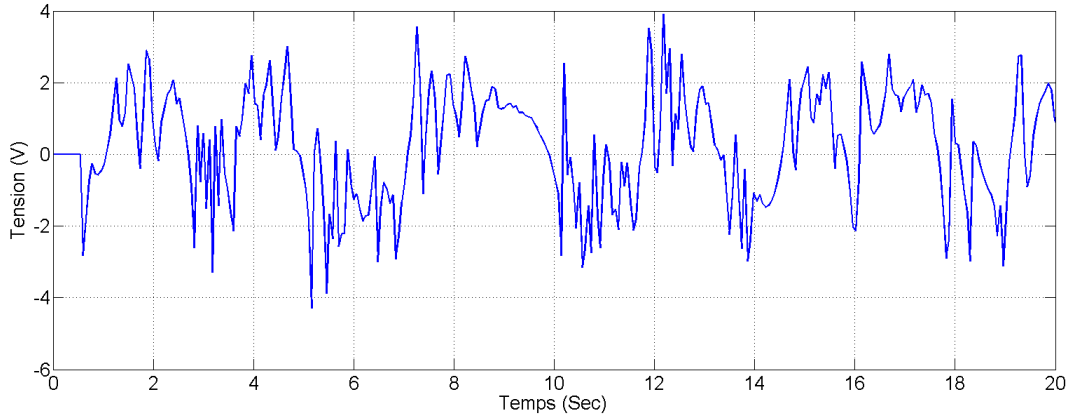

FIGURE 4.3 – Commande $u_x(t)$: stabilisation du robot avec la commande LQR

- Commande par mode glissant :

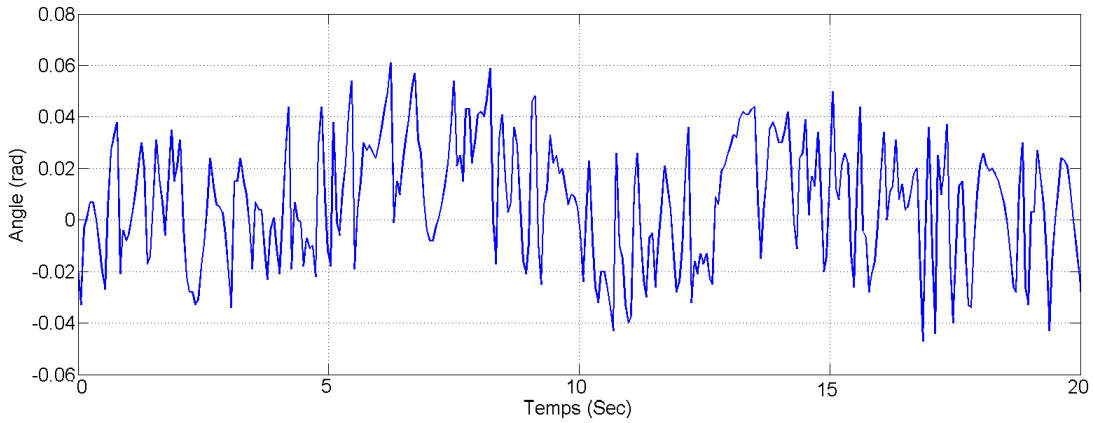

FIGURE 4.4 – Angle d'inclinaison $\psi(t)$: stabilisation du robot avec commande M.G

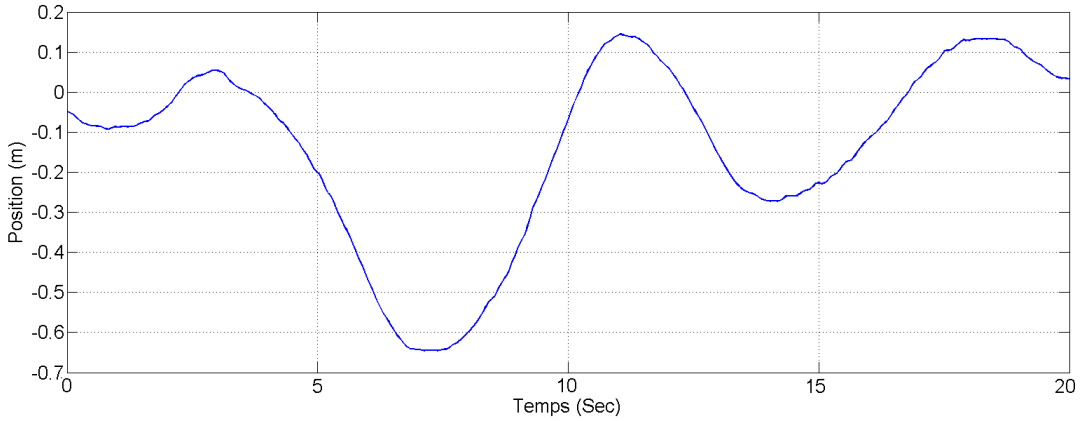

FIGURE 4.5 – Position $x(t)$: stabilisation du robot avec commande M.G

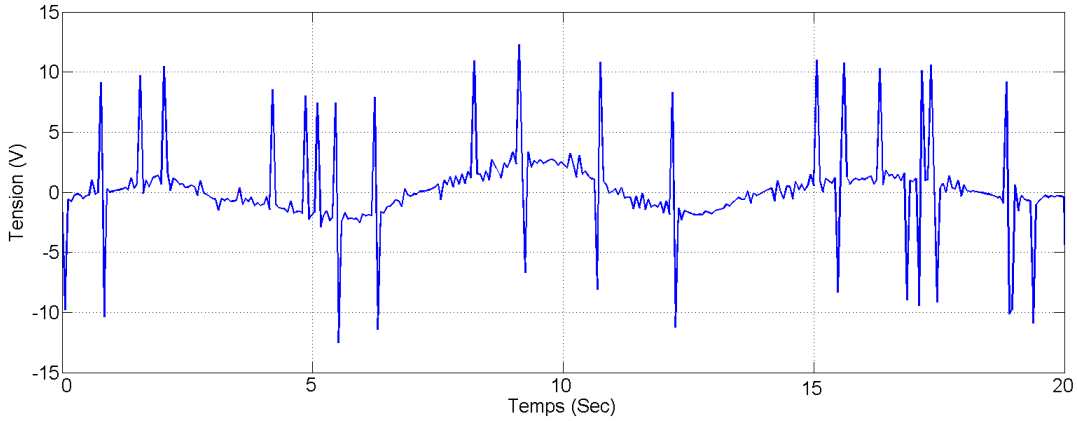

FIGURE 4.6 – Commande $u_x(t)$: stabilisation du robot avec la commande M.G

On peut voir en les figures 4.1 et 4.4, l'angle d'inclinaison obtenu par les deux commandes, que le système est stable, il y a des vibrations parce que il y a une bande de tension des moteurs dans laquelle ils ne fonctionnent pas $[-1.5v, 1.5v]$(limitation des moteurs). On remarque aussi que l'amplitude de vibration est de $-0.02rad$ à $0.03rad$, donc une bande de 3 dégrée dans la commande LQR moins que la valeur dans la commande par mode glissant (de $-0.04rad$ à $0.06rad$), ces sont des résultats acceptables.

Les figures 4.2 et 4.5 montrent les positions du robot pour chaque commande. On remarque que le robot dans les deux commandes se déplace en avant et en arrière pour s'équilibrer et en

même temps pour retourner à sa position initiale (x=0). Il se déplace dans intervalle de $0.07m$ pour la commande LQR et de $0.8m$ pour la commande par mode glissant.

Les figures 4.3 et 4.6 représentent la commande U_x du robot. On peut remarquer que la valeur de commande U_x est entre $4v$ *et* $-4v$ ce qui est considéré comme une bonne valeur parce qu'elle est petite. Pour la commande par mode glissant on peut voir clairement les composantes de la commande u_{eq} *et* u_d, la commande équivalent u_{eq} est sur la forme du signale sinusoïdal dans le graphe de la commande u_x et les piques sont la commande u_d qui se définit comme une fonction de saturation $u_d = \alpha * sat(S, \xi)$.

4.3.2 Test de robustesse

Dans ce test, on va introduire une perturbation sur le robot par un coup de main comme il est montré dans la figure 4.7, le trait noir dans la figure indique la position initiale.

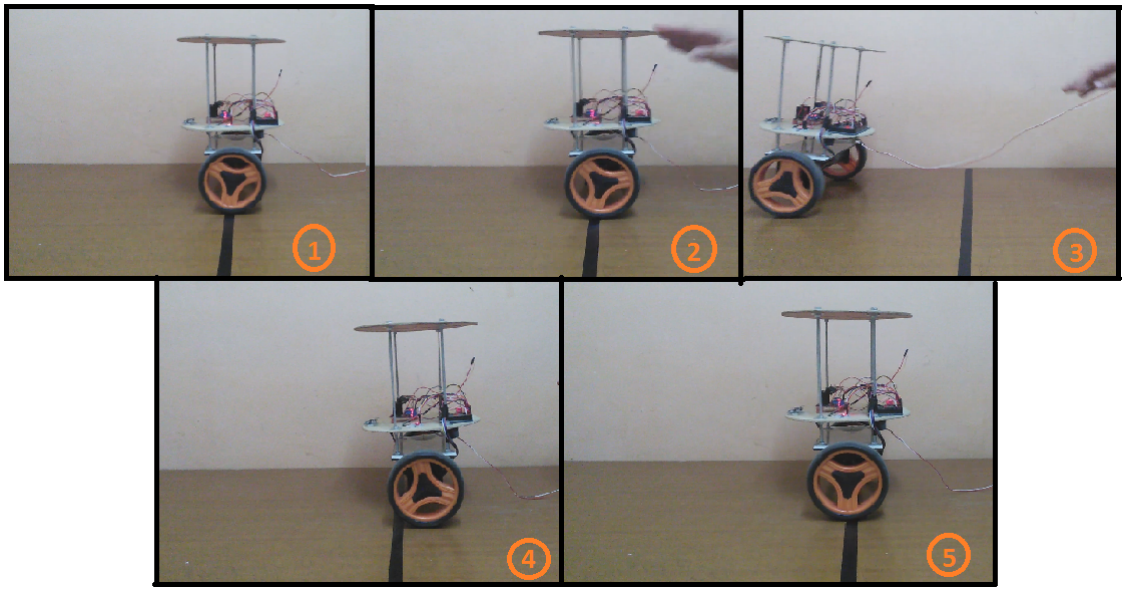

FIGURE 4.7 – Test de robustesse

On a obtenu les résultats suivants :
-Commande LQR : On a fait le coup à l'instant 6.8 seconde.

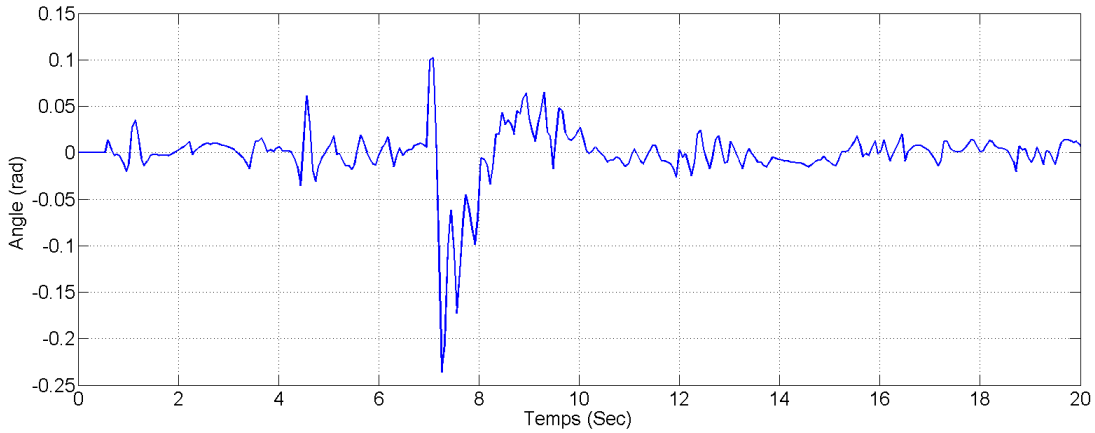

FIGURE 4.8 – Angle d'inclinaison $\psi(t)$: réponse du robot à une perturbation avec commande LQR

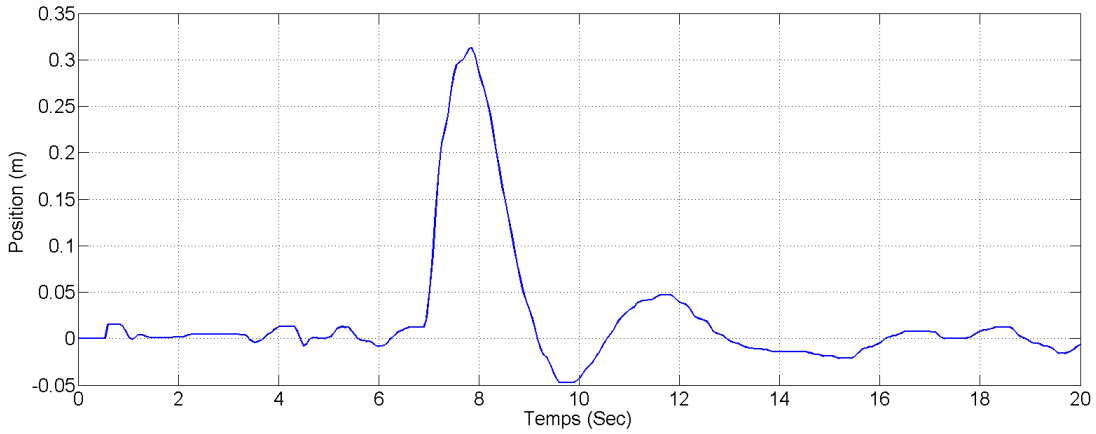

FIGURE 4.9 – Position $x(t)$: réponse du robot à une perturbation avec commande LQR

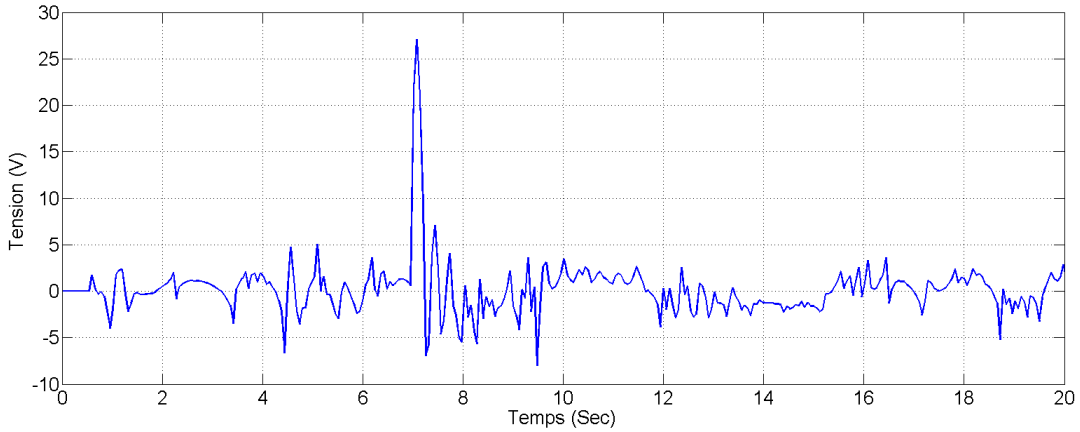

FIGURE 4.10 – Commande $u_x(t)$; réponse du robot à une perturbation avec commande LQR

- Commande par mode glissant :

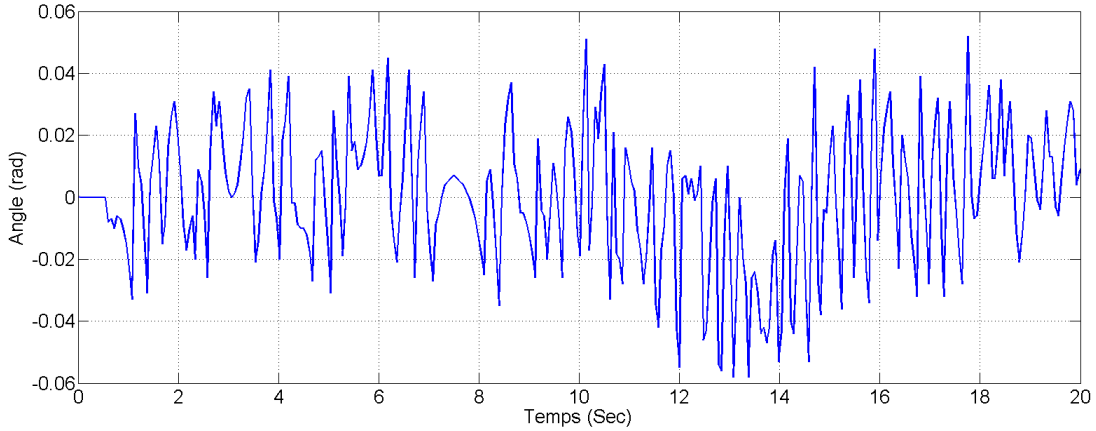

FIGURE 4.11 – Angle d'inclinaison $\psi(t)$: réponse du robot à une perturbation avec commande M.G

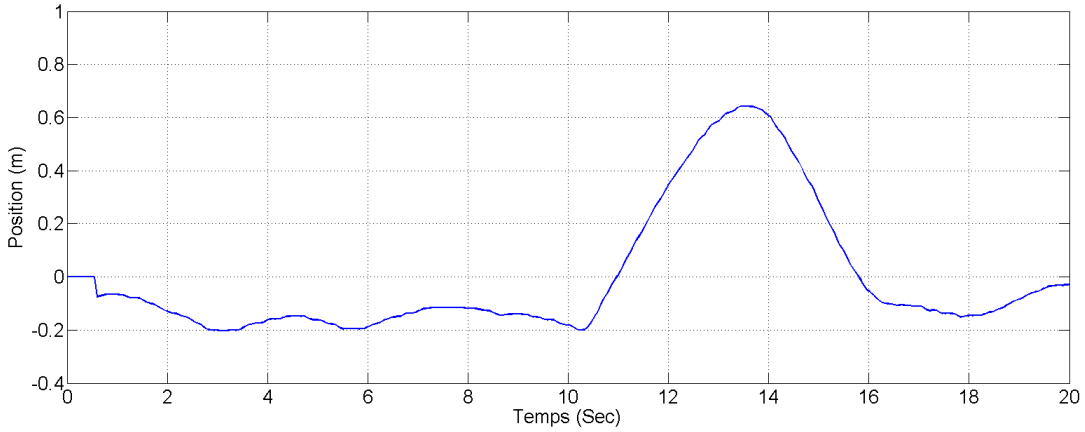

FIGURE 4.12 – Position $x(t)$: réponse du robot à une perturbation avec commande M.G

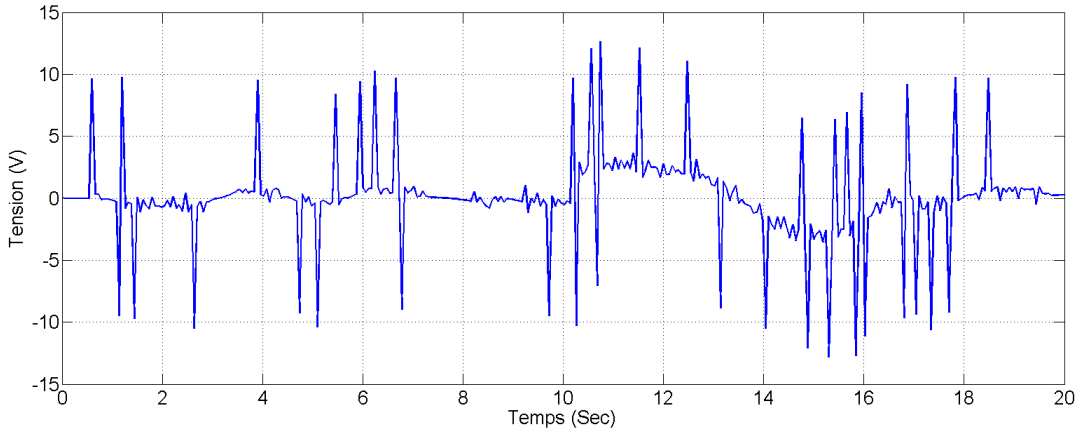

FIGURE 4.13 – Commande $u_x(t)$; réponse du robot à une perturbation avec commande M.G

Les figures 4.8, 4.9 et 4.10 montrent la réponse du robot à la perturbation avec la commande LQR. On remarque que la perturbation à l'instant $t = 6.8s$ incline le robot avec $0.1rad$ dans le sens de la force appliquée, on peut voir dans la figure 4.9 que le robot se déplace jusqu'à $0.32m$ dans le sens de l'inclinaison pour l'équilibrage, ce déplacement incline le robot de $0.2rad$ (12 dégrée) dans le sens inverse de la perturbation, comme on indique dans la figure 4.7 étape 3, puis il retourne à la position initiale. Pour la commande on peut voir dans la figure 4.10 que la commande augment jusqu'à $27v$ pour éliminer la perturbation mais cette tension est hors la limite du moteur ($V_{max} = 12v$ et $V_{min} = -12v$), on a programmé une fonction de saturation dans l'Arduino qui limite la commande avant qu'elle se transforme au signal PWM.

Les figures 4.11, 4.12 et 4.13 montrent la réponse du robot à la perturbation avec la commande par mode glissant. Premièrement, On peut dire que la force de coup appliqué sur le robot pour cette commande est moins que celle dans la commande LQR parce que la régulation de position est très faible. On a appliqué la perturbation a l'instant $t = 10.2s$, on ne peut pas remarquer la variation de l'angle d'inclinaison $\psi(t)$ figure 4.11, mais on peut voir trois piques successifs dans la commande pour retourner le système à la surface de glissement, on peut voir aussi dans la figure 4.12 que le robot se déplace jusqu'à 0.65 m avant qu'il retourne à sa position initial sans grande variation dans l'angle d'inclinaison, ça parce que on a donné une grande importance à la régulation de l'angle par rapport de position, dans un autre sens, on a choisit λ_1 (qui est relié à l'angle ψ) très grand par rapport à λ_2(qui est relié à la position) dans la surface S_1 (voir l'équation 3.17 du chapitre précédent). Si on augment λ_2 le robot va se déstabiliser.

4.3.3 régulation de position

On a donné une consigne de $x_{ref} = 1m$ au robot pour voir la poursuite du robot dans le déplacement linéaire :

-Commande LQR :

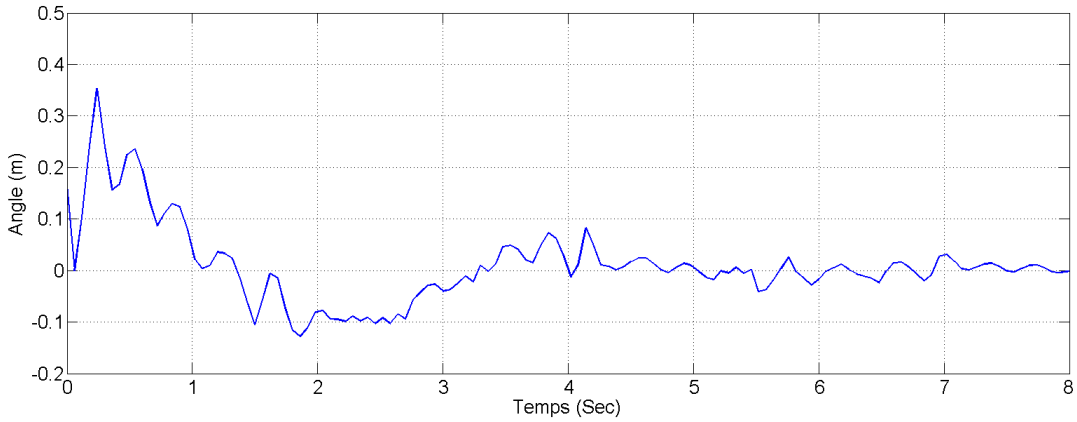

FIGURE 4.14 – Angle d'inclinaison $\psi(t)$: poursuite de la consigne de position linéaire avec la commande LQR

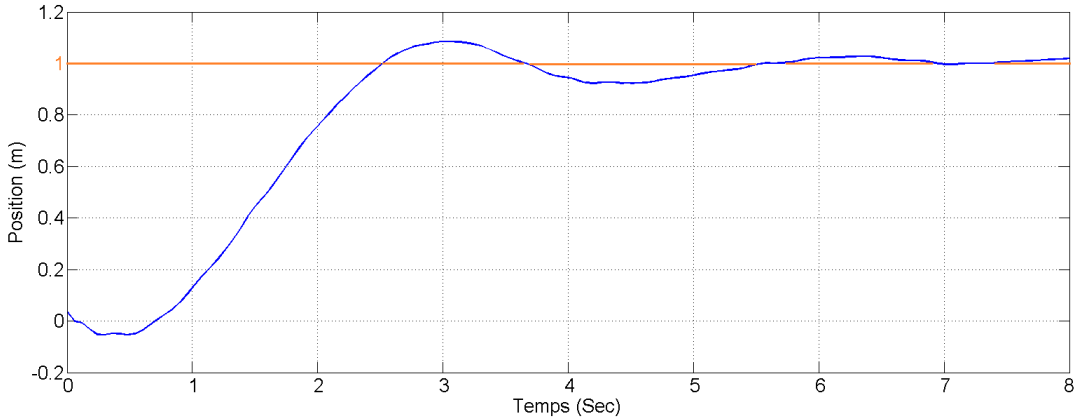

FIGURE 4.15 – Position $x(t)$: poursuite de la consigne de position linéaire avec la commande LQR

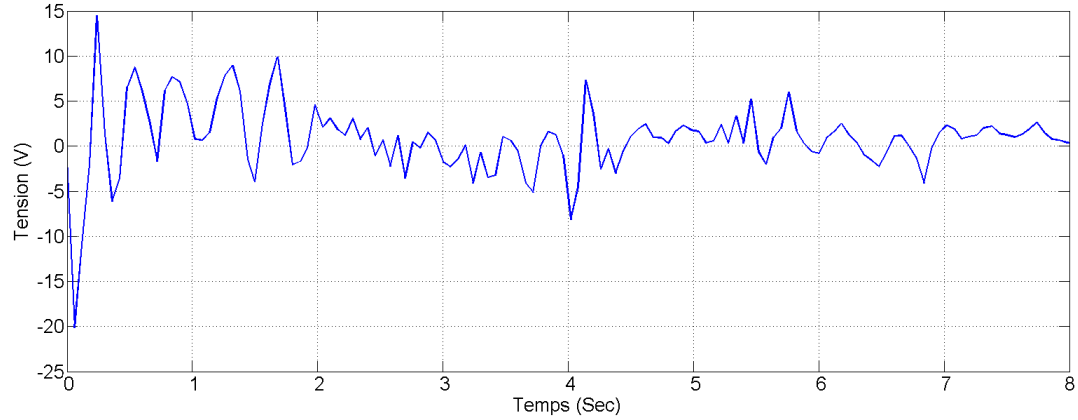

FIGURE 4.16 – Commande $u_x(t)$: poursuite de la consigne de position linéaire avec la commande LQR

- Commande par mode glissant :

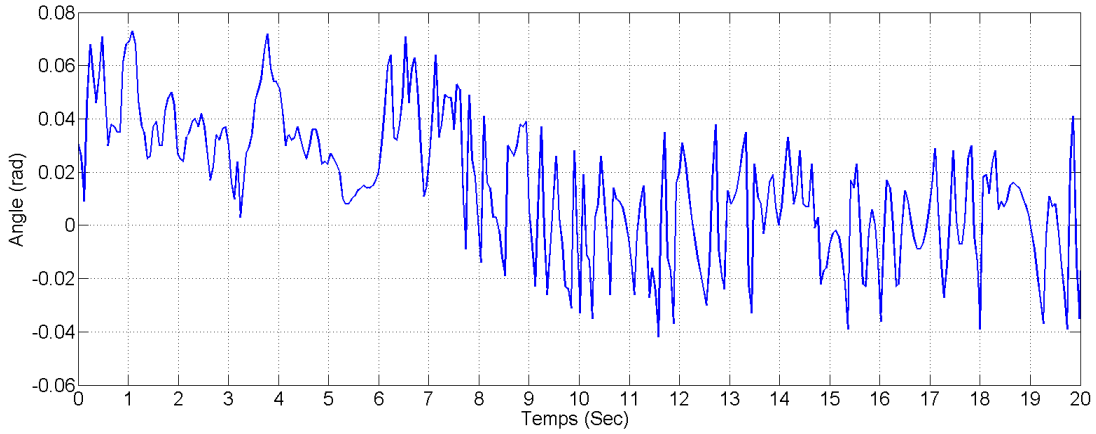

FIGURE 4.17 – Angle d'inclinaison $\psi(t)$: poursuite de la consigne de position linéaire avec la commande M.G

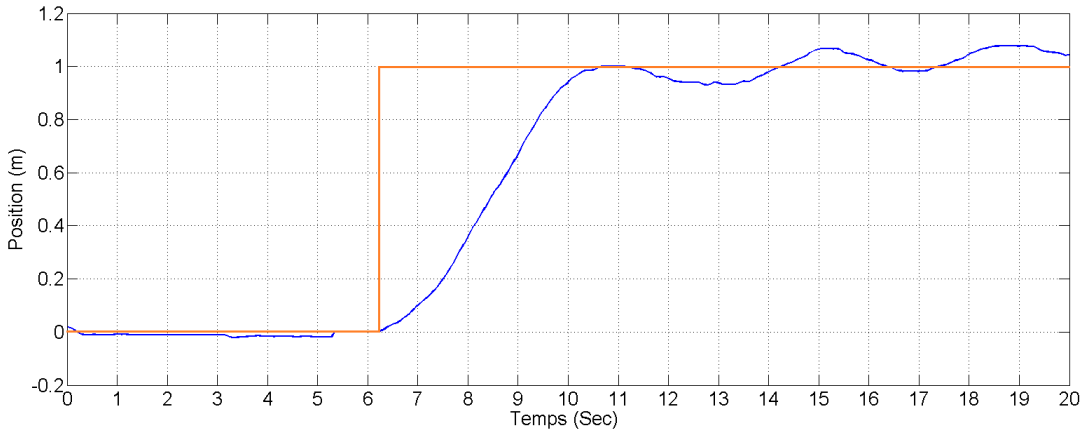

FIGURE 4.18 – Position $x(t)$: poursuite de la consigne de position linéaire avec la commande M.G

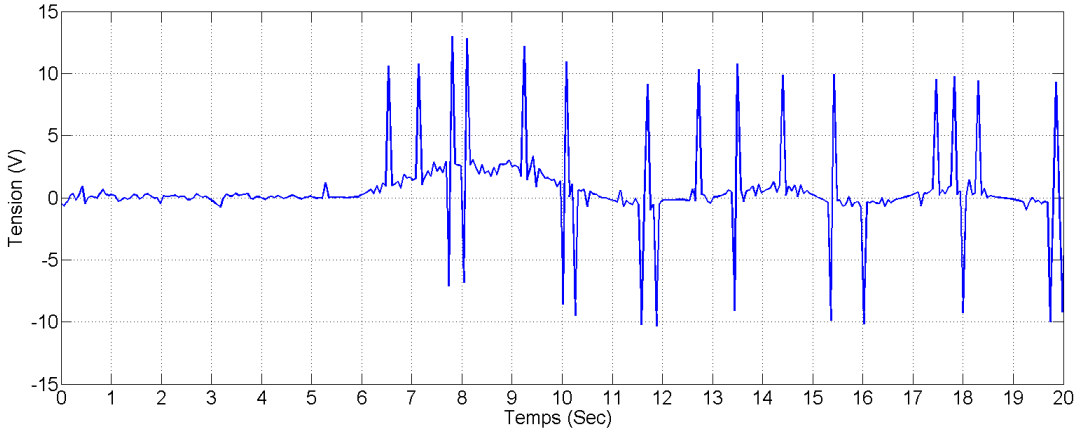

FIGURE 4.19 – Commande $u_x(t)$: poursuite de la consigne de position linéaire avec la commande M.G

Les figures 4.14, 4.15 et 4.16 représentent la poursuite par le robot de la consigne de position pour la commande LQR. Comme pour les résultats de simulation on remarque que le robot se

déplace un peu au sens inverse de la consigne afin d'incliner le corps (jusqu'à $0.35rad$) dans la direction de celle-ci puis il se déplace vers la consigne, ce déplacement dans le sens négatif parce que la dynamique de x a un zéro positif (phase non minimale). Le temps de réponses est $3s$, il est acceptable. On remarque que l'angle ψ devient négatif à l'instant $t = 10.4$ avant que le robot arrive à la consigne, ceci parce que la commande de position pose le robot en avant ce qui affecte l'angle d'inclinaison. On peut voir dans la figure que la commande U_x varie entre $10V$ et $-5V$ pour déplacer le robot à la consigne et l'équilibre parce que l'équilibrage du robot a plus d'importance tel que la valeur de sa pondération est la plus grande parmi les pondérations de la matrice Q.

Les figures 4.17, 4.18 et 4.19 montrent la poursuite par robot de la consigne de position avec la commande par mode glissant, premièrement on peut dire que dans la période de 0 à 6 seconde on a fixé le robot par la main, c'est pour ça il n'y a aucun vibration ou déplacement, on remarque que le robot suit la consigne sans grande inclinaison, et que le temps de réponse est grand 5 seconde. il y a des oscillations autour la consigne variant entre $1.09m$ et $0.94m$.

4.3.4 Test de rotation

Dans ce test on va donner au robot une consigne de rotation ($\delta_{ref} = \frac{\pi}{2}$), on peut voir les réponses de système dans les graphes suivants :

-Commande LQR :

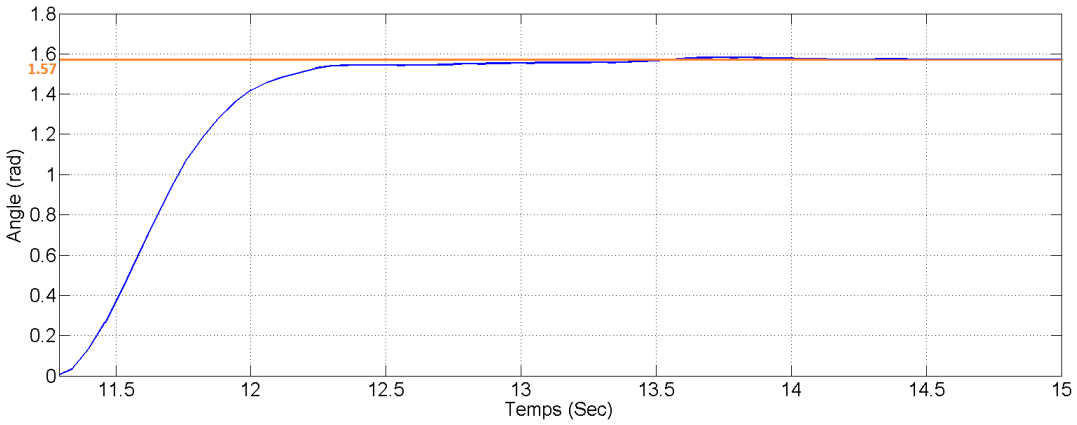

FIGURE 4.20 – Angle de rotation $\delta(t)$: poursuite de la consigne de rotation avec la commande LQR

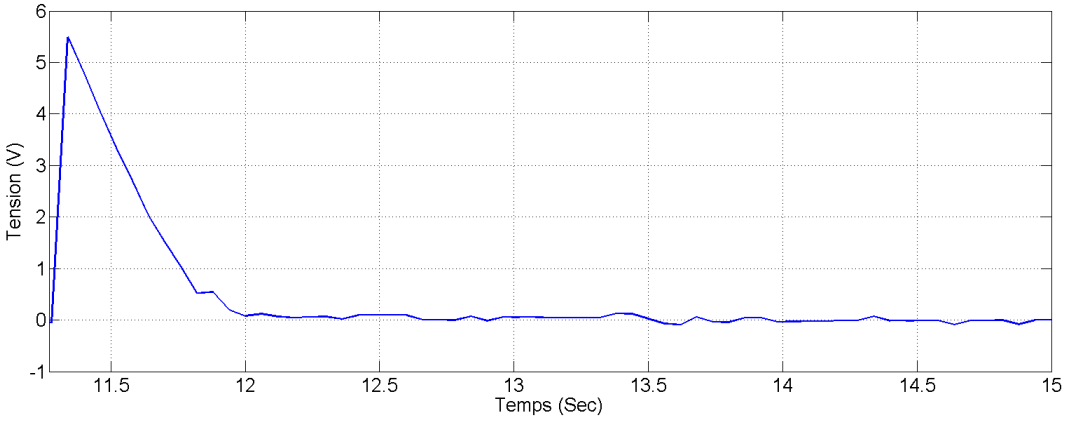

FIGURE 4.21 – Commande $u_h(t)$: poursuite de la consigne de rotation avec la commande LQR

- Commande par mode glissant :

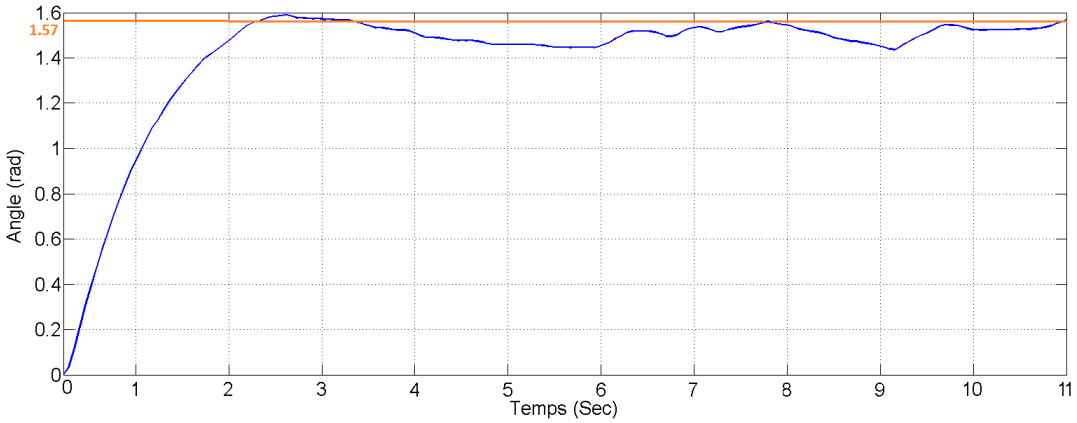

FIGURE 4.22 – Angle de rotation $\delta(t)$: poursuite de la consigne de rotation avec la commande M.G

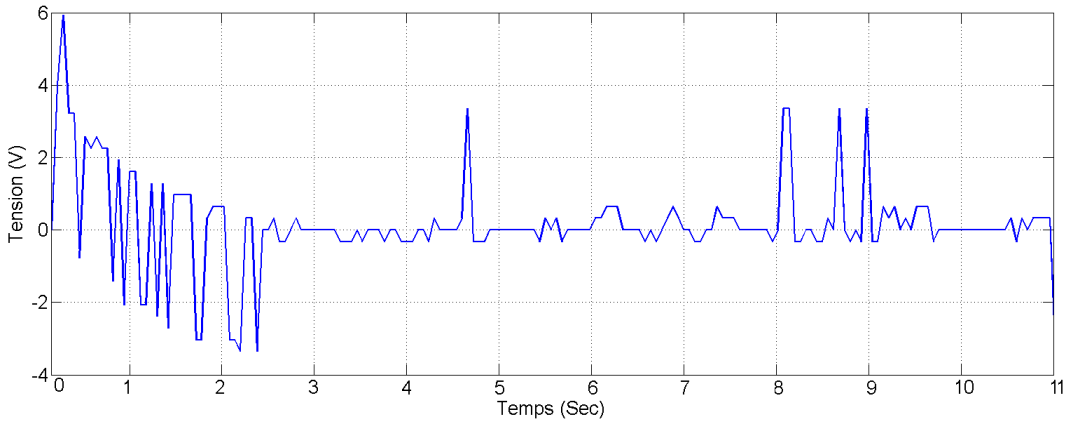

FIGURE 4.23 – Commande $u_h(t)$ poursuite de la consigne de rotation avec la commande M.G

les figures 4.20 et 4.21 montrent la réponse du robot à la consigne de l'angle de la rotation avec une commande LQR. On remarque que le robot a poursuivi la consigne dans un temps de réponse de $1s$ et sans dépassement. Dans la figure de 4.21 on peut voir que la commande u_h ne dépasse pas 6volt, c'est considéré comme une bonne valeur parce que la limite dela tension de chaque moteur est $12v$ et qui se partage avec la commande u_x, donc l'augmentation de la commande u_h affecte l'application de la commande u_x ce qui peut déstabiliser le robot. L'augmentation de la valeur de pondération de delta diminue le temps de réponse et il augmente la valeur de la commande u_h.

Les figures 4.22 et 4.23 montrent la réponse du robot avec la commande par mode glissant. On remarque que le robot poursuit la consigne avec un temps de réponses de $2.2s$, il est lent par rapport à la commande LQR, il y a des oscillations autour de la consigne. La commande u_h par mode glissant comporte beaucoup d'oscillations (chattering) dans le régime transitoire, c'est pour ramener les états à la surface de glissement S_2.

4.4 Application de la commande du robot en temps réel

On a connecté le robot à smart phone qui nous permet d'envoyer la consignes de vitesse linéaire et de vitesse angulaire (rotation) en temps réel.
On peut présenter des résultats sur cet application avec la commande LQR :

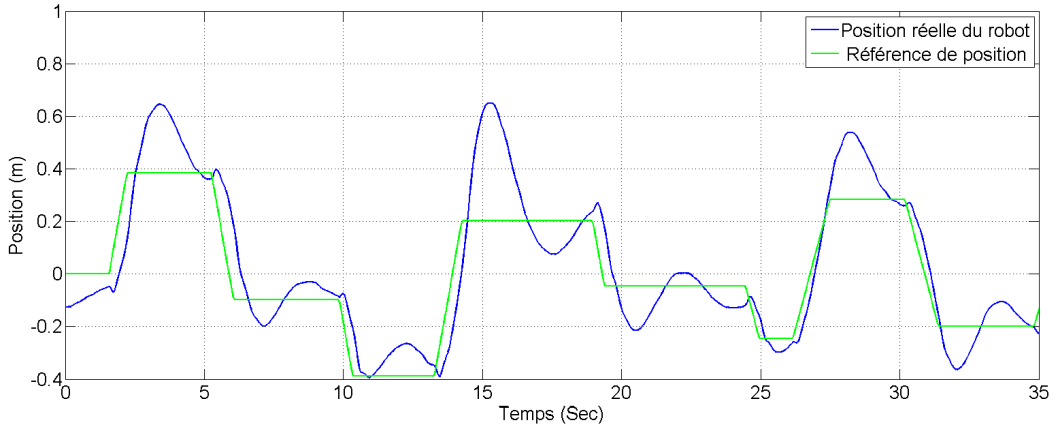

FIGURE 4.24 – Poursuite de la référence de position

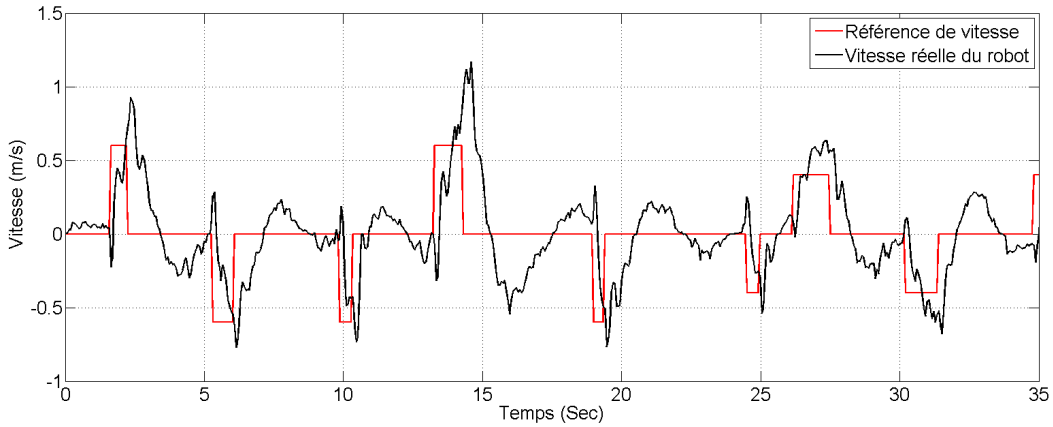

FIGURE 4.25 – Poursuite de la référence de vitesse

Les figures 4.24 et 4.25 montrent la poursuite par le robot de la consigne de vitesse. On remarque que le robot poursuit la consigne de vitesse, il y a des vibrations parce que le robot balance en position vertical. On remarque toujours que le robot se déplace un peu au sens contraire de la consigne parce que la dynamique de x a un zéro positif, ce déplacement crée un retard dans la réponse en position comme on peut le voir dans la figure 4.25, le graphe vert est l'intégration de la consigne de vitesse. On remarque aussi que le robot dépasse la consigne de position parce que sa vitesse est relativement grande, donc il ne peut pas s'arrête directement.

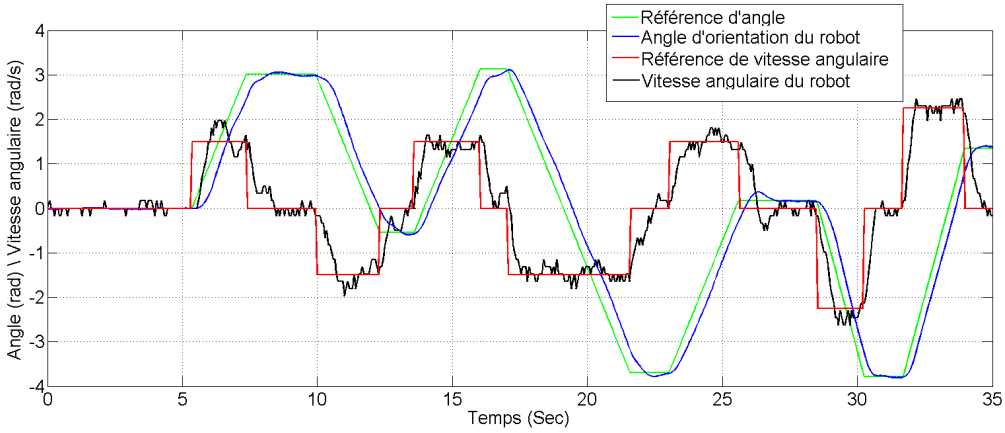

FIGURE 4.26 – Poursuite de la référence de rotation

La figure 4.26 représente la réponse du robot a la consigne de vitesse angulaire avec la

commande LQR. On remarque que robot poursuit bien la consigne. Il y a un retard à cause des moteurs qui ont besoin d'un couple élevé pour le démarrage, il n y a pas les vibrations parce que la dynamique de rotation est indépendant à la dynamique de l'équilibrage.

4.5 Conclusion

Dans ce chapitre, on a présenté les résultats relatifs à l'implémentation de la commande LQR et de la commande par mode glissant. Les résultats sont bonnes, tel que la stabilisation du robot dans la position vertical, convergence du robot vers la position désirée tout en assurant son déplacement dans l'intervalle de temps toléré, les performances vis-à-vis des perturbations, sauf que la régulation du position linéaire par la commande par mode glissant est lente et diverge parfois. On a trouvé que les performances de la commande LQR sont plus intéressantes que celles de la commande par mode glissant soit dans la stabilisation du robot, soit dans la convenance vers la position désirée. On n'a pas étudié en détail la commande par mode glissant pour des raisons de temps, donc il y a possibilité d'améliorer cette commande ou d'appliquer les modes glissants d'ordre supérieur.

Finalement, on a utilisé la commande LQR pour commander le robot en temps réel par un smart phone qui a donné de bonnes réponses.

Conclusion générale

Dans le cadre du projet de fin d'études, on a étudié et réalisé un robot capable de s'équilibrer sur deux roues, et de se déplacer en avant et en arrière, de tourner et de poursuivre une trajectoire par utilisation de la commande optimal LQR et la commande par mode glissant. Ce projet nous a permis d'aborder les aspects d'implémentation expérimentale, et des techniques qu'on a étudiées durant nous formations théorique.

Dans le premier chapitre, on est menés d'abord à connaître la constitution de ce système et comprendre son principe de fonctionnement. La modélisation du système du robot balanceur était nécessaire afin de synthétiser les lois de commande proposées dans notre travail. D'après la modélisation et les simulations en boucle ouverte, nous avons constaté l'instabilité de ce système.

Dans le deuxième chapitre, on a présenté la conception du robot et une description détaillée sur toutes les pièces utilisées pour réaliser le robot. La réalisation du robot contient deux parties, la partie mécanique qui concerne la construction du robot, et la partie électrique qui concerne la réalisation de schéma électrique et la programmation de l'Arduino pour créer un environnement idéal pour implémenter les lois de commande. On a aussi fait la conception du filtre complémentaire, qui nous a permet de fusionner les données de l'accéléromètre et du gyroscope pour mesurer l'angle d'inclinaison avec le minimum des erreurs.

Dans le troisième chapitre, après la confirmation de la commandabilité et l'observabilité du système du robot on a fait la synthèse de trois lois de commande, la commande optimale LQR, la commande PID et la commande par mode glissant. Les résultats de simulation en Matlab montrent que la commande optimale est la plus performante et aussi la plus facile à synthétiser. La synthèse du régulateur PID est très difficile à cause du couplage de la dynamique de l'angle d'inclinaison et de la dynamique de position, donc une variation dans un régulateur affecte les deux, pour des raisons de temps on a satisfait par la simulation, et on propose d'utiliser le régulateur PID en cascade [20] comme un bon solution.

Après la réalisation du robot et la synthèse de lois de commande, on a implémenté les commandes. Les résultats d'implémentation sont représentés dans le quatrième chapitre. La commande optimale LQR a réalisé tous les objectifs du projet avec une bonne performance. La commande par mode glissant a de bonnes performances dans le côté de stabilisation et la rotation du robot, mais il est très faible pour la régulation de position. A cause de temps on n'a pas pu faire une grande étude de la commande par mode glissant donc la commande capable de s'améliorer ou d'appliquer les modes glissants d'ordre supérieur . Finalement, on a développé le programme de l'Arduino, qui nous a permet de commander le robot par le smart phone en temps réel par utilisation de la commande LQR.

Bibliographie

[1] Z. Li, C. Yang, L. Fan, " Advanced Control of Wheeled Inverted Pendulum Systems ", Springer-Verlag London, 2013.

[2] M. Boulouiha, " Techniques de commande avanée ", note de cours, universitaire de Rélizane, 2015.

[3] S. Islam, S. Ul-Mahmud, " A low cost MEMS and complementary filter based attitude Heading Reference System (AHRS) for Low Speed Aircraft ", IEEE, Military Institute of Science and Technology Dhaka, Bangladesh, 2016.

[4] W. An, Y. Li " Simulation and control of a two-wheeled self-balancing robot ", IEEE, International Conference on Robotics and Biomimetics (ROBIO) Shenzhen, China, December 2013.

[5] A. A. Bature, S. Buyamin, M. N. Ahmad, M. Muhammad, " A comparison of controllers for balancing two wheeled inverted pendulum Robot ", International Journal of Mechanical and Mechatronics Engineering IJMME-IJENS Vol :14 No :03, June 2014.

[6] M. Lozeau " Commande par supervision de systèmes mécatroniques via internet " Mémoire présenté en vue de l'obtention du diplôme de maitrises sciences, École Polytechnique de Montréal, Septembre 2009.

[7] M. Ahmed, C, Azer " Commande d'un robot à deux roues ", Projet fin d'étude, Institut supérieur des études technologiques de Nabeul, Tunisie, 2016.

[8] I. Deghboudj " Commade des systémes non lineares par mode glissant d'orde superieur ", thèse de magistère, Université Constantine 1, Novembre 2013.

[9] http ://lei.epfl.ch/page-36905-fr.html

[10] http ://www.segway.com/

[11] https ://techcrunch.com/2016/05/23/ibot-wheelchair/

[12] http ://www.futura-sciences.com/tech/videos/handle-robot-deux-roues-boston-dynamics-devoile-etonnantes-capacites-4507/

[13] https ://www.arduino.cc/en/Main/ArduinoBoardMega2560

[14] https ://wiki.mchobby.be/index.php ?title=Pont-H_L298N

[15] http ://www.futura-sciences.com/magazines/high-tech/infos/dico/d/technologie-gyroscope-11121/

[16] http ://www.simius.be/interfacage/accelerometer.html

[17] http ://www.generationrobots.com/blog/fr/2014/01/encodeurs-robotique-mobile/

[18] http ://www.generationrobots.com/blog/fr/2017/03/capteur-ultrason-capteur-a-ultrason-en-vente-chez-generation-robots/

[19] http ://www.generationrobots.com/blog/fr/2017/03/capteur-ultrason-capteur-a-ultrason-en-vente-chez-generation-robots/

[20] D. Pratama, E. H. Binugroho, F. Ardilla, " Movement Control of Two Wheels Balancing Robot using Cascaded PID Controller ", International Electronics Symposium (IES), Surabaya, Indonesia, 2015.

Annexe

Arduino IDE

L'environnement de programmation Arduino IDE offre une interface simple et pratique pour programmer n'importe quelle carte commande de firme Arduino.

Cependant, il existe quelques logiciels alternatifs qui permettent de programmer la carte Arduino.

Utiliser un langage de programmation qu'on maitrise déjà permet de ne pas avoir a apprendre un nouveau langage pour programmer la carte Arduino. Cela permet aussi de réutiliser les librairies et programmes que l'on a éventuellement déjà développés pour d'autres familles de micro-contrôleurs. Pour les programmeurs confirmes, le langage C/C++ qui est traditionnellement utilisé pour programmer les micro-contrôleurs reste la solution la plus performante. D'autre part, si l'on possède des connaissances et l'on dispose de ressources techniques et de partenaires qui travaillent sur d'autres plateformes, rester sur celles-ci est peut être un choix pertinent.

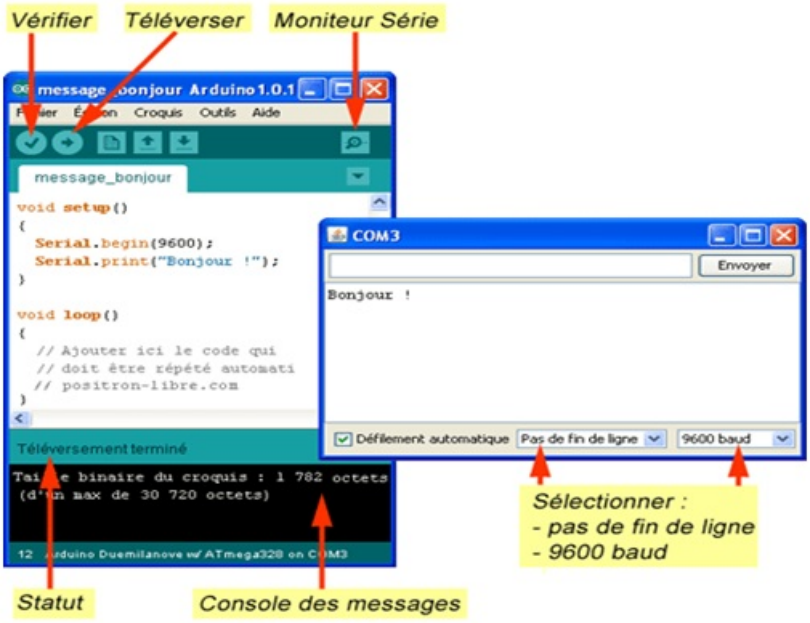

FIGURE 4.27 – Arduino Integrated Development Environment

Caractéristiques du moteur

Gear ratio : 43.7 :1
Free-run speed @ 12V : 251 rpm
Free-run current @ 12V : 0.4A
Rated torque @ 12V : 1.5 kg*cm
Encoder Resolution : 16CPR(motor shaft)/2096CPR(gearbox shaft)
Weight : 205g

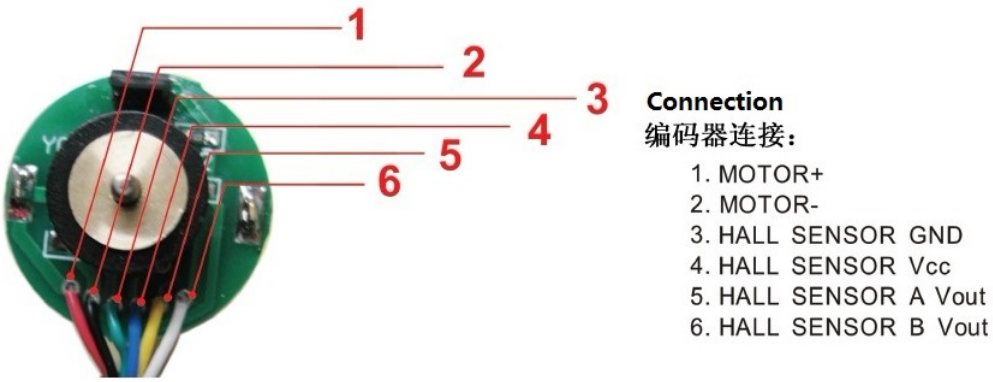

FIGURE 4.28 – Schéma de l'encodeur du moteur

Les paramètres des commandes

Commande LQR

On a utilisé les matrices de pondération suivantes dans la simulation de la commande LQR :

$$Q = \begin{bmatrix} 1500 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1400 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 150 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \qquad R = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

Pour les matrices de pondération de la commande implémenté :

$$Q = \begin{bmatrix} 9000 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 400 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 40 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \qquad R = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

Commande par mode glissant

simulation :
$\lambda_1 = 7,\ \lambda_2 = 1,\ \alpha_1 = 300,\ \xi_1 = 0.8$
$\lambda_3 = 15,\ \alpha_2 = 10,\ \xi_2 = 0.1$
implémentation :
$\lambda_1 = 50,\ \lambda_2 = 2,\ \alpha_1 = 300,\ \xi_1 = 1.8$
$\lambda_3 = 2,\ \alpha_2 = 1000,\ \xi_1 = 0.3$

Caractéristiques du MPU 6050

6 Electrical Characteristics

6.1 Gyroscope Specifications

VDD = 2.375V-3.46V, VLOGIC (MPU-6050 only) = 1.8V±5% or VDD, T_A = 25°C

PARAMETER	CONDITIONS	MIN	TYP	MAX	UNITS	NOTES
GYROSCOPE SENSITIVITY						
Full-Scale Range	FS_SEL=0		±250		°/s	
	FS_SEL=1		±500		°/s	
	FS_SEL=2		±1000		°/s	
	FS_SEL=3		±2000		°/s	
Gyroscope ADC Word Length			16		bits	
Sensitivity Scale Factor	FS_SEL=0		131		LSB/(°/s)	
	FS_SEL=1		65.5		LSB/(°/s)	
	FS_SEL=2		32.8		LSB/(°/s)	
	FS_SEL=3		16.4		LSB/(°/s)	
Sensitivity Scale Factor Tolerance	25°C	-3		+3	%	
Sensitivity Scale Factor Variation Over Temperature			±2		%	
Nonlinearity	Best fit straight line; 25°C		0.2		%	
Cross-Axis Sensitivity			±2		%	
GYROSCOPE ZERO-RATE OUTPUT (ZRO)						
Initial ZRO Tolerance	25°C		±20		°/s	
ZRO Variation Over Temperature	-40°C to +85°C		±20		°/s	
Power-Supply Sensitivity (1-10Hz)	Sine wave, 100mVpp; VDD=2.5V		0.2		°/s	
Power-Supply Sensitivity (10 - 250Hz)	Sine wave, 100mVpp; VDD=2.5V		0.2		°/s	
Power-Supply Sensitivity (250Hz - 100kHz)	Sine wave, 100mVpp; VDD=2.5V		4		°/s	
Linear Acceleration Sensitivity	Static		0.1		°/s/*g*	
SELF-TEST RESPONSE						
Relative	Change from factory trim	-14		14	%	1
GYROSCOPE NOISE PERFORMANCE	**FS_SEL=0**					
Total RMS Noise	DLPFCFG=2 (100Hz)		0.05		°/s-rms	
Low-frequency RMS noise	Bandwidth 1Hz to10Hz		0.033		°/s-rms	
Rate Noise Spectral Density	At 10Hz		0.005		°/s/ √ Hz	
GYROSCOPE MECHANICAL FREQUENCIES						
X-Axis		30	33	36	kHz	
Y-Axis		27	30	33	kHz	
Z-Axis		24	27	30	kHz	
LOW PASS FILTER RESPONSE						
	Programmable Range	5		256	Hz	
OUTPUT DATA RATE						
	Programmable	4		8,000	Hz	
GYROSCOPE START-UP TIME	**DLPFCFG=0**					
ZRO Settling (from power-on)	to ±1°/s of Final		30		ms	

1. Please refer to the following document for further information on Self-Test: *MPU-6000/MPU-6050 Register Map and Descriptions*

6.2 Accelerometer Specifications

VDD = 2.375V-3.46V, VLOGIC (MPU-6050 only) = 1.8V±5% or VDD, T_A = 25°C

PARAMETER	CONDITIONS	MIN	TYP	MAX	UNITS	NOTES
ACCELEROMETER SENSITIVITY						
Full-Scale Range	AFS_SEL=0		±2		*g*	
	AFS_SEL=1		±4		*g*	
	AFS_SEL=2		±8		*g*	
	AFS_SEL=3		±16		*g*	
ADC Word Length	Output in two's complement format		16		bits	
Sensitivity Scale Factor	AFS_SEL=0		16,384		LSB/*g*	
	AFS_SEL=1		8,192		LSB/*g*	
	AFS_SEL=2		4,096		LSB/*g*	
	AFS_SEL=3		2,048		LSB/*g*	
Initial Calibration Tolerance			±3		%	
Sensitivity Change vs. Temperature	AFS_SEL=0, -40°C to +85°C		±0.02		%/°C	
Nonlinearity	Best Fit Straight Line		0.5		%	
Cross-Axis Sensitivity			±2		%	
ZERO-G OUTPUT						
Initial Calibration Tolerance	X and Y axes		±50		mg	1
	Z axis		±80		mg	
Zero-G Level Change vs. Temperature	X and Y axes, 0°C to +70°C		±35			
	Z axis, 0°C to +70°C		±60		mg	
SELF TEST RESPONSE						
Relative	Change from factory trim	-14		14	%	2
NOISE PERFORMANCE						
Power Spectral Density	@10Hz, AFS_SEL=0 & ODR=1kHz		400		µg/ √Hz	
LOW PASS FILTER RESPONSE						
	Programmable Range	5		260	Hz	
OUTPUT DATA RATE						
	Programmable Range	4		1,000	Hz	
INTELLIGENCE FUNCTION INCREMENT			32		mg/LSB	

1. Typical zero-g initial calibration tolerance value after MSL3 preconditioning
2. Please refer to the following document for further information on Self-Test: *MPU-6000/MPU-6050 Register Map and Descriptions*

Code Matlab pour enregistrer les données du robot et les dessiner dans des graphe en temps réel

Bluetooth

```
%CHELLI Kalbeddine
%Ecole Nationale Polytechnique
%Code Matlab  pour connecter Arduino a matlab par Bluetooth et enregistrer
% les donnees et dessiner les graphes en temps réel


t=1:1:100;
i=1;
W=[0.01:0.01:1]; %Initialiser 1er vecteur des données (graphe)
P=[0.01:0.01:1]; %Initialiser 2eme vecteur des données (graphe)
Z=[0.01:0.01:1]; %Initialiser 3eme vecteur des données (graphe)
X=[0.01:0.01:1]; %Initialiser 4eme vecteur des données (graphe)
x=[0 0 0 0 0 0 0 0 0 0 0];%Initialiser 1er vecteur des données(sauvgarder)
z=[0 0 0 0 0 0 0 0 0 0 0];%Initialiser 2eme vecteur des données(sauvgarder)
w=[0 0 0 0 0 0 0 0 0 0 0];%Initialiser 3eme vecteur des données(sauvgarder)
p=[0 0 0 0 0 0 0 0 0 0 0];%Initialiser 4eme vecteur des données(sauvgarder)
instrhwinfo('Bluetooth','HC-05') % Définir le Bluetooth
b = Bluetooth('HC-05',1); % Définir le Bluetooth
 fopen(b); % Ouvrir la connexion
 while(i<600) % nombre des donnees
      %Lire les donner envoyer par arduino
    Psi=fscanf(b,'%f');
    Uh=fscanf(b,'%f');
    dot_Psi=fscanf(b,'%f');
    Psi_c=fscanf(b,'%f');
   %Mise à jour des vecteurs des données
   if(i>10)

    X=[X(2:100),Psi];
    Z=[Z(2:100),Uh];
    W=[W(2:100),dot_Psi];
    P=[P(2:100),Psi_c];
x(i)=Psi;
z(i)=Uh;
w(i)=dot_Psi;
p(i)=Psi_c;
   end
 plot( t,X,'r',t,W,'g',t,Z,t,P,'k'); %dessiner les graphes

  drawnow;
    i=i+1
 end
 fclose(b); % fermer la connexion
```

USB

```
%CHELLI Kalbeddine
%Ecole Nationale Polytechnique
%Code Matlab  pour connecter Arduino a matlab par USB et enregistrer les
% les donnees et dessiner les graphes en temps réel


t=1:1:100;
i=1;
W=[0.01:0.01:1]; %Initialiser 1er vecteur des données (graphe)
P=[0.01:0.01:1]; %Initialiser 2eme vecteur des données (graphe)
Z=[0.01:0.01:1]; %Initialiser 3eme vecteur des données (graphe)
X=[0.01:0.01:1]; %Initialiser 4eme vecteur des données (graphe)
x=[0 0 0 0 0 0 0 0 0 0 0];%Initialiser 1er vecteur des données(sauvgarder)
z=[0 0 0 0 0 0 0 0 0 0 0];%Initialiser 2eme vecteur des données(sauvgarder)
w=[0 0 0 0 0 0 0 0 0 0 0];%Initialiser 3eme vecteur des données(sauvgarder)
p=[0 0 0 0 0 0 0 0 0 0 0];%Initialiser 4eme vecteur des données(sauvgarder)

ard=serial('COM13','BaudRate',115200)% Définir le port USB

fopen(ard) % Ouvrir la connexion



 while(i<1000)  % nombre des donnees
    %Lire les donner envoyer par arduino
     Psi_c=fscanf(ard,'%f');
     Psi_gy=fscanf(ard,'%f');
     Psi_acc=fscanf(ard,'%f');
     Psi=fscanf(ard,'%f');

    %Mise à jour des vecteurs des données
   if(i>10)

        X=[X(2:100),Psi];
        Z=[Z(2:100),Psi_gy];
        W=[W(2:100),Psi_acc];
        P=[P(2:100),Psi_c];
x(i)=Psi;
z(i)=Psi_gy;
w(i)=Psi_acc;
p(i)=Psi_c;
   end
 plot( t,X,'r',t,W,'g',t,Z,t,P,'k'); %dessiner les graphes
  drawnow;
```