الجمهورية الجزائرية الديمقراطية الشعبية
République Algérienne démocratique et populaire

وزارة التعليم العالي و البحث العلمي
Ministère de l'enseignement supérieur et de la recherche scientifique

جامعة سعد دحلب البليدة 1
Université SAAD DAHLAB de BLIDA 1

كلية التكنولوجيا
Faculté de Technologie

قسم الإلكترونيك
Département d'Électronique

Mémoire de Master

Mention Électronique
Spécialité Automatique

présenté par

Zerroukhat Wassim

&

Daoud Noureddine

Conception et réalisation d'un pendule inversé de type Segway

Proposé par : Mr Boualem KAZED

Année Universitaire 2015-2016

Remerciements

Nos premiers remerciements vont tout naturellement au Dieu le tout puissant pour la force, la patience et le courage qui nous a donné pour accomplir ce travail et qui ont été un gage de réussite.

Nous voudrions en deuxième lieu exprimer notre reconnaissance envers notre promoteur Mr KAZED.B pour sa confiance en nous permettant d'effectuer cette étude, pour son implication permanente et presque quotidienne dans le déroulement de ce travail ainsi ses précieux conseils pour mener à bien ce projet dont nous avons apprécié la valeur toute ou longue de cette période.

Que tous les membres de jury soient remerciés pour le soutien, l'intérêt et l'attention qu'ils ont porté a cette thèse.

Un grand Mercie à tous les enseignants du département d'électronique, leurs encouragements et leurs aides, qui ont contribué à nos formations.

En fin nos remerciements vont également à tous nos amis et camarade de labo pour leur soutient, leur appui moral qu'ils nous ont témoigné ainsi que leurs disponibilité durant les jours du travail.

Dédicace

Je tiens à dédier ce modeste travail à :

A mes très chers parents qui ont tellement fait de sacrifices pour moi.

A mon grand père qui était mon soutient toutes ces années d'études.

Ma chère sœur, mes frères, mes oncles, mes tantes, mes cousins et tous les membres de la famille zerroukhat et bouzina , chacun par son nom.

Mon binôme avec qui j'ai partagé les bons et les mouvais moments.

A tous mes amis et surtout : Mohamed , Amine ,Adel ,Hassane et Abdou

A Toute personne ayant participé de prés ou de loin pour la réalisation de ce projet.

A Toute la promotion d'automatique 2015 /2016.

A tous ceux que j'aime et qui m'aiment.

Z.Wassim

DÉDICACES

Je dédie ce modeste travail aux gens qui m'ont soutenu et aidé à préparer ce travail:

A mes chers parents et à toute ma grande famille DAOUD.

Mes frères, mes sœurs,

Mes enseignants,

Mon Binôme,

Mes amis,

Mes collègues.

A tous mes amis de la promotion Master ATM 2015-2016.

A tous ceux que j'aime et qui m'aiment .

DAOUD NOUREDDINE

ملخص:

بسم الله الرحمان الرحيم، والصلاة والسلام على أنبيائه، لقد تم بحمد الله إتمام مشروع التخرج بعنوان السيطرة على نواس مقلوب على شكل Segway بواسطة المنظم PID المنفذ في بطاقة أردوينو.

تحتوي بطاقة أردوينو على المتحكم ATmega2560 الذي يعالج المعلومات الآتية من الملتقط gy-521 MPU-6050 ،المتحكم يستعمل هذه المعلومات للتحكم في المحركات. بهدف الحفاظ على استقرار ال Segway .

كلمات المفاتيح : gy-521 MPU-6050 ; Arduino ; PID ; segway .

Résumé : Au nom de Dieu le miséricordieux, prière et paix soient sur ses Prophètes, avec la louange de Dieu il a été possible de réaliser ce projet de fin d'études intitulé: contrôle d'un pendule inversé de type segway par régulateur PID qui implémenter sur une carte Arduino.

La carte Arduino, muni d'un microcontrôleur ATmega2560, traite les informations acquise par le capteur gy-521 MPU-6050, le microcontrôleur utilisent ces informations pour commander les moteurs. Dont le but de stabiliser la position du segway.

Mots clés : segway ; PID ; Arduino ; gy-521 MPU-6050.

Abstract :

In the name of God the Merciful, prayer and peace be upon His Prophets, with the praise of God it has been possible to realize this project graduation titled : control of an inverted pendulum type segway by PID controller which implement on an Arduino board.

Arduino board, equipped with a ATmega2560 microcontroller processes the information acquired by the gy-521 MPU-6050 sensor, microcontroller use these information to control motors. Which aims to stabilize the position of the segway.

Keywords : segway ; PID ; Arduino ; gy-521 MPU-6050.

Listes des acronymes et abréviations

MCC : Moteur à courant continu.

PID : proportionnel intégral dérivé.

PWM: Pulse Width Modulation.

USB: Universal Serial Bus.

SRAM: Static Random Access Memory.

EEPROM: Electrical Erasable Programmable Read Only Memory.

LED: Light Emitting Diode.

SDA: Serial Data Line.

SCL: Serial Clock Line.

ADC: Analog to Digital Converter.

IDE : environnement de développement.

TWI : Two Wire Interface.

I2C : Interface à deux Conducteur.

DMP : Digital Motion Processing.

TTL : Transistor-Transistor Logic.

RPM : rotations par minute.

M : poids de la plate-forme.

m : poids pendule.

x: La position.

θ: Angle pendule.

b : frottement du chariot.

L : distance au centre de gravité du pendule.

I: inertie du pendule.

F: force appliqué au chariot.

$u(t)$: Tension appliquée au moteur.

$e(t)$: Force contre électromotrice.

$i(t)$: Intensité traversant le moteur.

$\Omega(t)$: Vitesse de rotation du rotor.

K_e: Constante de vitesse.

K_c: Constante de couple.

C_p: Couple de pertes.

C_u: Couple moteur généré.

f: Coefficient de frottement visqueux.

C_r : Couple résistant.

J: Moment d'inertie de l'axe du rotor.

Table des matières

Chapitre 3 : Partie électronique et hardware

Chapitre 4 : Test et résultat

Liste des figures

Liste des tableaux

Introduction générale

Le but de ce projet est de réalisation d'un segway et l'étude des principes nécessaires pour contrôler efficacement ce système robotique, La capacité de balance sur deux roues est très efficace pour la mobilité.

- Faire la connaissance de segway et robots en équilibre, et le principe de l'auto balance.
- Trouver les paramètres dynamiques du véhicule, les équations d'état du modèle.
- L'identification des systèmes.
- Construire la simulation en Matlab.
- Projeter le modèle et fabriquer la structure mécanique du modèle.
- Faire la programmation en arduino.

Le Segway est un moyen de transport personnel dont l'équilibre est assuré par un système automatique, construit autour du principe du pendule inversé.

La description du pendule inversé muni de son actionneur et ses capteurs ainsi que les divers phénomènes physiques présents lors du fonctionnement, montrent la forte complexité due aux nombreux non linéarités ainsi que la difficulté à modéliser parfaitement la dynamique du système pendule inverse -capteurs- actionneur.

Le contrôle du pendule inversé pour le redressement et la stabilisation devient ardu, car la connaissance du système se révèle imprécise et imparfaite, il est largement utilisé comme banc d'essai pour analyser les performances des différents types des contrôleurs. Afin d'assurer la stabilité du segway, deux moteurs à courant continu feront une plate-forme mobile qui va déplacer vers l'avant et vers l'arrière avec une vitesse variable, ce sera calculé dans un système embarqué basé sur un microcontrôleur dédié.

Dans un segway nous nous intéressons à la mesure de l'inclinaison du pendule, Pour mesurer l'inclinaison du système pendule inversé nous avons besoin de mesurer l'accélération dans un axe, avec ces données, il est possible de déterminer l'inclinaison du segway.

Introduction générale

Dans ce projet, nous avons été invité à mettre en œuvre un contrôleur PID pour stabiliser la position du segway en utilisant un microcontrôleur ATmega2560, le capteur mpu6050 mesure la vitesse angulaire et l'accélération de ce dernier, après envoyer cette mesure vers l'arduino par le protocole i2c.

Le microcontrôleur ATmega2560 a transformé le signal de commande en terme de rapport cyclique qui fournira la carte de puissance avec le signal désiré, à la fois les moteurs rendre le segway sur la position vertical stable. Le courant requis pour les moteurs est alimenté par un double pont en H, connecté de sorte que le seul PWM est nécessaire pour faire tourner les moteurs dans un sens ou dans l'autre. Ceci a été rendu possible en utilisant un Schmidt déclenchement de type inverseur qui fournit le signal PWM opposé utilisé en tant que second signal d'entrée du pont en H.

1.1 Introduction

Dans ce chapitre, nous présentons le pendule inversé et ces différents robots à deux roues pour faire la connaissance de segway.

1.2 Pendule inversé

1.2.1 Définition d'un pendule inversé

Un pendule inversé est un pendule qui, sa masse est au dessus de son point de pivotement. Il est souvent mis en œuvre avec le point de pivot monté sur un chariot qui peut se déplacer horizontalement et peut être appelé un panier ou un pôle. Alors qu'un pendule normale est stable lorsqu'il est accroché à la baisse, un pendule inversé est intrinsèquement instable, et doit être activement équilibré afin de rester debout, soit en appliquant un couple au point de pivot ou en déplaçant le point de pivot à l'horizontale dans le cadre d'un système de rétroaction [1].

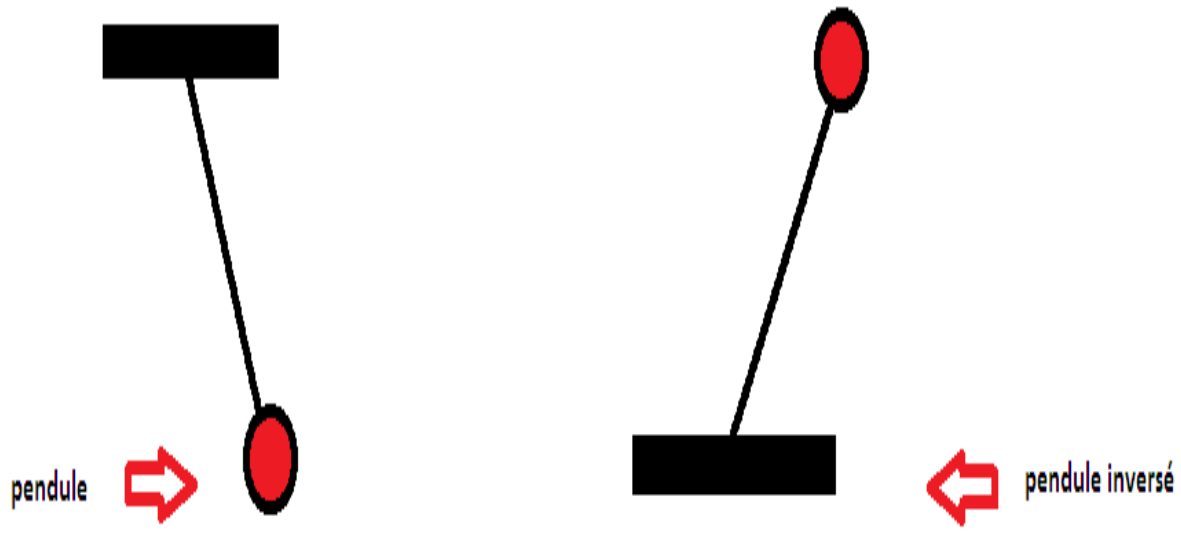


Figure 1.1. Schéma de pendule inversé.

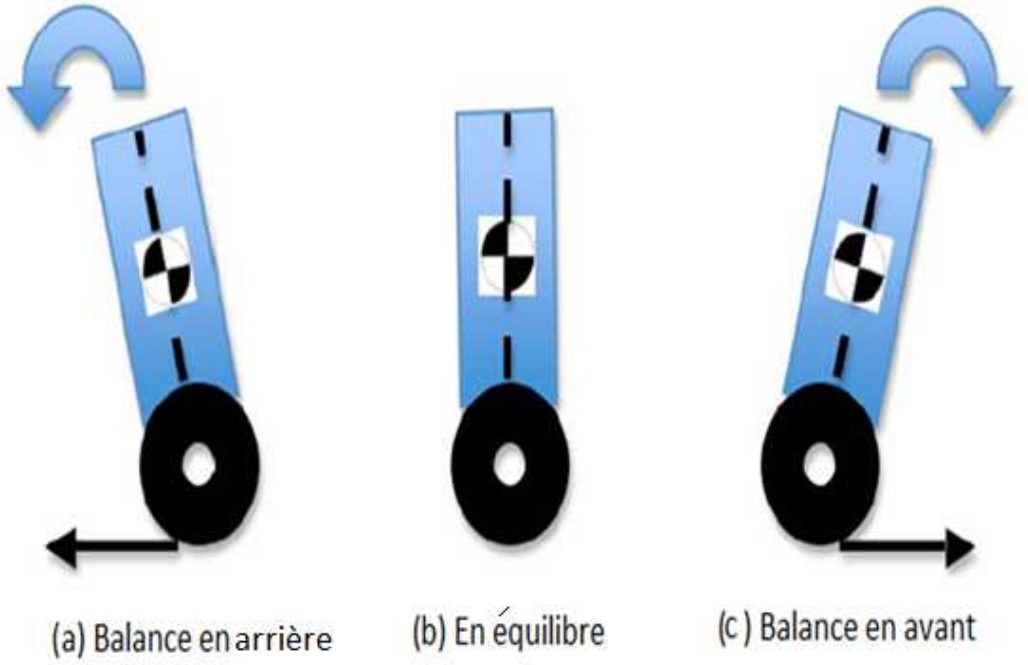


Figure 1.2. Les trois cas du pendule inversé.

1.2.2 Applicatives du pendule inversé

La validation des méthodes de poursuite par retour visuel permettra à une mise en œuvre au sein du projet « Attelage Virtuel » (suite du projet AVIVA) qui consiste à « accrocher » un véhicule meneur (avec conducteur) et un suiveur (autonome) grâce à des capteurs « immatériels » ou « sans contact ».

Par ailleurs, l'intégration de systèmes de navigation tels que GPS (Etats Unis), GLONASS (Russie), GALILEO (Europe) et BEIDOU (Chine) permettra de réaliser la commande à distance, la commande via des réseaux de transports autonomes (guidage, détection et localisation des défaillances).

La figure 1.3 illustre des applications potentielles du pendule inversé.

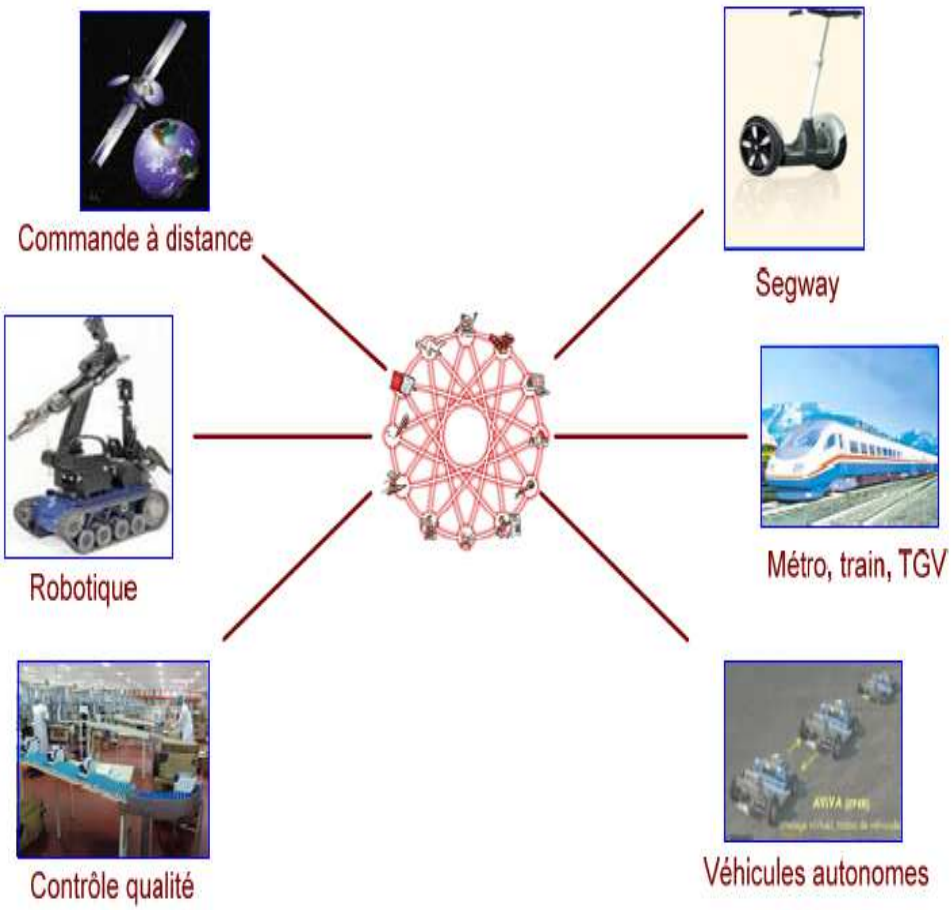


Figure 1.3. Les applications du pendule inversé.

1.3 Les robots d'équilibrage de deux roues

1.3.1 Le robot EMIEW

EMIEW est un robot humanoïde développé par l'équilibrage d'Hitachi groupe de recherche. EMIEW représente une excellente mobilité et l'existence Interactive comme Workmate.

Ce robot est un guide ou un robot de surveillance dans un environnement du monde réel tel que les bureaux, les usines ou les hôpitaux. Hitachi a mis au point deux modèles d'EMIEW. Ces deux modèles peuvent éviter les obstacles et ont une vitesse de pointe de 6 kilomètres par heure.

EMIEW2 a été donné une mobilité accrue en déployant un système de suspension pour chaque jambe et de reconnaissance vocale. La Figure 1.4 illustre plus de photos d'EMIEWs. [2]

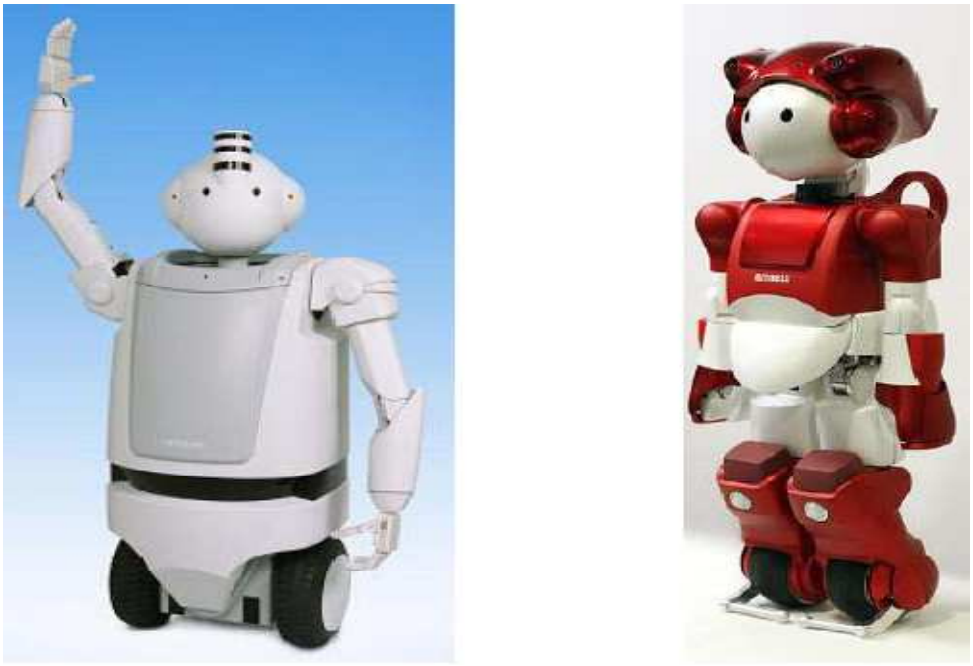


Figure 1.4. Les robots EMIEW (gauche) et EMIEW2 (droite).

1.3.2 Le robot Joe

Joe est un autre robot d'équilibrage de deux roues innovant (Figure 1.5). Les deux contrôles découplés de ce travail de robot côte à côte servent de :

- équilibrer le robot et le contrôler vers l'avant et la marche arrière.
- postes de contrôle autour de son axe vertical.

Cette radio machine contrôlée offre la possibilité de tourner autour de son axe vertical et d'effectuer un demi-tour. Joe filtre les signaux provenant d'un accéléromètre et gyroscope tant pour mesurer l'angle d'inclinaison du robot et de produire un signal d'inclinaison. Elle est utilisée avec les codeurs des moteurs pour déterminer la position du robot. [3]

Figure 1.5. Le robot joe.

1.3.3 Le robot nBot

Le nBot a été développé en 2007 par David Anderson (Anderson 2007), voir la figure 1.6 La stabilité du robot est induite par l'utilisation d'un filtre de Kalman. Ceci fournit une entrée précise pour contrôler la stabilité en fusionnant les sorties du gyroscope et un accéléromètre.

Actuellement, le nBot a été révisé cinq fois. La version actuelle a un système de navigation qui aide à éviter les obstacles. En outre, le robot peut Traverser un terrain accidenté et peut descendre un ensemble d'escaliers tout en maintenant toujours son équilibre. [3]

Figure 1.6. Le robot nBot.

1.3.4 Le robot Anybots

Anybots est une société fondée par Trevor robotique Blackwell en 2001. QA et QB sont deux robots de téléprésence mobiles développés par Anybots qui prennent l'équilibre sur deux roues et manœuvrent en douceur. Ces robots peuvent fournir une communication facile via l'Internet pour un utilisateur, qui ne peu pas assister à la deuxième place. QB est la dernière version du robot de téléprésence d'Anybot. Il peut pivoter autour de son axe vertical et d'entraînement à 5,6 kilomètres par heure. Les figures 1.7 illustre QA et QB, respectivement. [4]

Figure 1.7. Les robots QA (gauche) et QB (droite).

1.3.5 Le robot Segway

a Définition d'un segway

Le Segway PT pour Transporteur Personnel Segway (anciennement appelé Segway HT) est un gyropode, c'est-à-dire 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'une manche de maintien et de conduite.

Segway est un transporteur bien connu par l'auto équilibrage personnel, qui a été produit par Segway Inc du New Hampshire, États-Unis [5]. Il a été inventé par Dean Kamen en 2001.

Segway est entraîné par des servomoteurs et il peut accélérer jusqu'à 20 kilomètres par heure. Un capteur d'inclinaison et de plusieurs gyroscopes sont employés dans Segway pour détecter le mouvement angulaire du robot. Segway coureur est capable d'accélérer et de décélérer en se penchant en avant ou en arrière. En 2006 IBot était un autre produit développé par Kamen [6]. IBot est un fauteuil roulant électrique mobile, Il monte les escaliers et se tient en équilibre.

IBot augmente sa hauteur quand il tient en équilibre sur deux roues. L'augmentation de la hauteur offre une vision du niveau des yeux pour une personne désactivée pour communiquer avec d'autres personnes. Figure 1.8 illustre les fonctionnalités IBOT.

Figure 1.8. Les robots Segway i180 series (gauche) et iBot 4000 (droite).

b Principe de fonctionnement d'un segway

Le Segway est le premier moyen de transport fondé sur le principe de l'équilibre dynamique.

Pour avancer ou reculer, il suffit de se pencher en avant ou en arrière. Le Segway s'occupe de maintenir l'équilibre du pilote !

Le fonctionnement du Segway est en effet calqué sur le corps humain : Lorsque vous vous penchez en avant, votre oreille interne indique à votre cerveau que vous êtes en train de perdre l'équilibre. Vos muscles reçoivent alors l'ordre de mettre un pied en avant pour rétablir les choses. Vous ne tombez pas, vous marchez, en mettant un pied devant l'autre ! Le Segway fonctionne de la même manière. Ses pieds sont ses roues, son cerveau est un ensemble de micro processeurs, et son oreille interne est constituée des gyroscopes et des capteurs très sophistiqués.

Le Segway détecte votre inclinaison et donne aux roues le mouvement nécessaire pour que vous restiez en équilibre tout en déplaçant.

Quand le Segway s'incline en avant, les deux roues tournent vers l'avant pour l'empêcher de tomber. Et inversement si le Segway s'incline en arrière.

Quand l'utilisateur penche le LeanSteer dans une direction, les moteurs font tourner une roue plus vite que l'autre ou dans des directions opposées pour faire pivoter le Segway. [7]

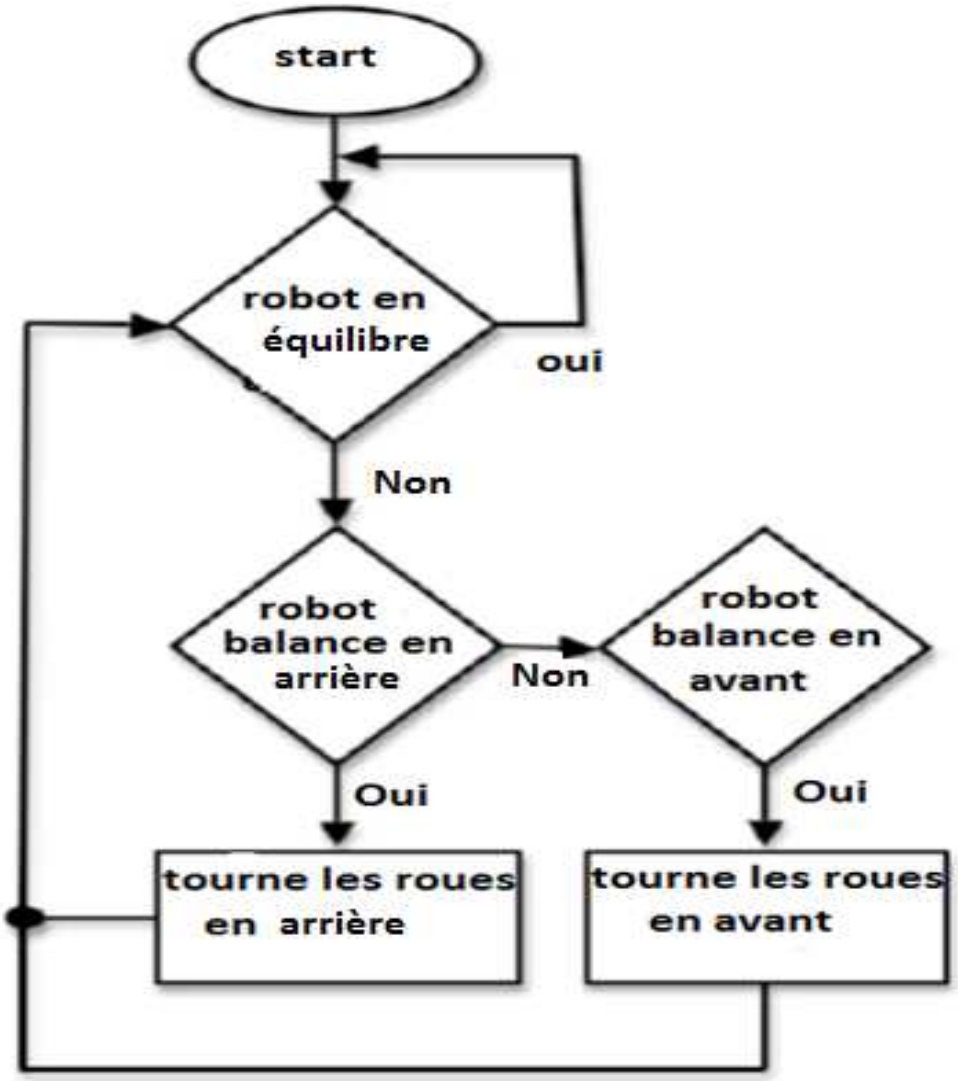


Figure 1.9. Diagramme de robot pendule inversé.

1.4 Les autres robots

L'internet fournit des exemples de diverses constructions de robots, de systèmes de contrôle et de matériaux qui peuvent être utilisés. par exemple, des ressources telles que lego Bloks et d'autres matériaux domestiques ont été utilisés.

L'incorporation de différents mécanismes tels que l'infrarouge (IR) appariés ont été utilisés pour déterminer les mesures d'inclinaison. L'inclinaison du robot dans une direction particulière sur la base des comparaisons que les deux capteurs fournissent.

Les systèmes de contrôle linéaires sont utilisés dans un grand nombre de ces conceptions et dans la plupart du temps fournissent un robot stable. [3]

1.5 Conclusion

Les robots d'équilibrage à deux roues viennent dans de nombreuses formes. Leurs gammes d'utilisation du transport est à l'entrainement. Comme il a été découvert dans ce chapitre un segway n'est pas très différent des autres robots mentionnés. Il utilise des gyroscopes et des accéléromètres, comme la plupart des robots.

2.1 Introduction

Dans ce chapitre on va commencer par la modélisation d'un pendule inversé et sa fonction de transfert, et la deuxième partie on parle sur la modélisation et la régulation d'un moteur à courant continu.

2.2 Modélisation du pendule inversé

Un pendule inversé est un problème de contrôle classique. Le processus est non linéaire et instable avec un signal d'entrée et plusieurs signaux de sortie.

Le système auquel on s'intéresse est le système mécanique représenté par la figure suivante. Il s'agit d'un axe, articulé en rotation dans un plan, surmonté d'une masselotte et porté par un chariot en translation, sur lequel on exerce une force F.

L'objectif est d'équilibrer un pendule verticalement sur un chariot.

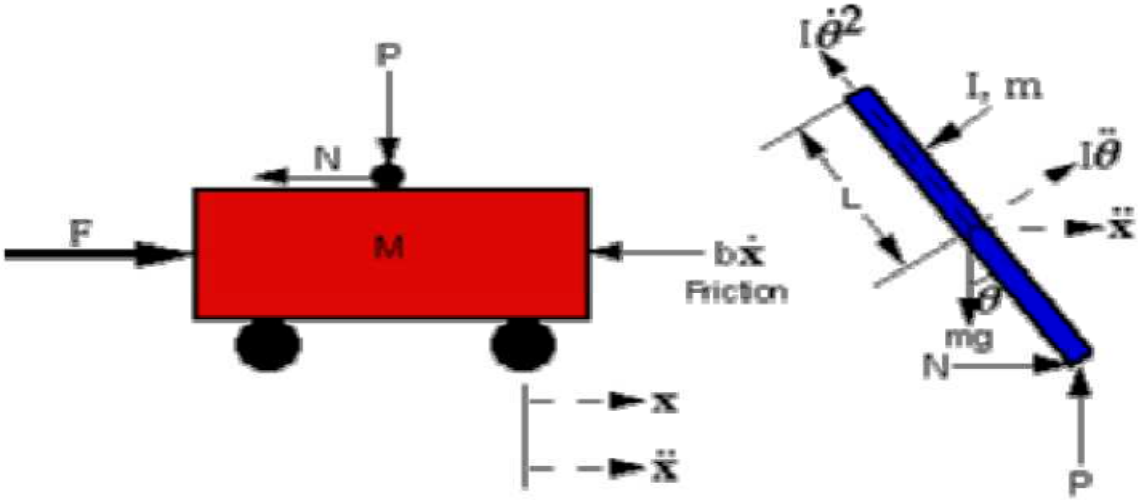


Figure 2.1. Représentation de la force de pendule inversé.

Avec:

M: Poids de la plate-forme.

m: Poids de pendule.

L : distance au centre de gravité du pendule.

θ: Angle de pendule.

b : frottement du chariot.

x: La position.

I: inertie du pendule.

F: force appliqué au chariot.

La somme des forces dans le diagramme du corps libre du chariot dans le sens horizontal, nous obtenons l'équation de mouvement suivante:

$$\mathbf{M\ddot{x} + b\dot{x} + N = F} \text{.....(2.1)}$$

La force exercée dans la direction horizontale en raison du moment sur le pendule est déterminé comme suit:

$$\boldsymbol{\tau = r \times F = I\ddot{\theta}} \text{.....(2.2)}$$

$$\mathbf{F = \frac{I\ddot{\theta}}{r} = mI\ddot{\theta}} \text{.....(2.3)}$$

Composant de cette force dans la direction de N est: $\mathbf{mI\ddot{\theta}cos\theta}$

La composante de la force centripète agissant le long de l'axe horizontal est la suivante :

$$\mathbf{F = \frac{I\dot{\theta}^2}{r} = mI\dot{\theta}^2} \text{.....(2.4)}$$

Cette composante de force dans la direction de N est : $\mathbf{mI\dot{\theta}^2 sin\theta}$

La somme des forces dans le diagramme du corps libre du pendule dans la Direction horizontal, nous obtenons une équation pour N:

$$\mathbf{N = m\ddot{x} + mI\ddot{\theta}cos\theta - mI\dot{\theta}^2 sin\theta} \text{ .....(2.5)}$$

Si nous remplaçons cette équation (2.5) dans la première équation (2.1), nous obtenons la première équation du mouvement pour ce système:

$$\mathbf{(M + m)\ddot{x} + b\dot{x} + mI\ddot{\theta}cos\theta - mI\dot{\theta}^2 sin\theta = F} \text{.....(2.6)}$$

Pour obtenir la deuxième équation du mouvement, la somme des forces perpendiculaires à la pendule. Cet axe est choisi pour simplifier la complexité

mathématique. Résoudre le système le long de cet axe finit par sauver beaucoup d'algèbre. Tout comme l'équation précédente est obtenue, les composantes verticales de ces forces sont considérées ici pour obtenir l'équation suivante:

$$\mathbf{P\sin\theta + N\cos\theta - mg\sin\theta = mI\ddot{\theta} + m\ddot{x}\cos\theta} \ldots\ldots\ldots\ldots\ldots\ldots\ldots (2.7)$$

Pour débarrasser les termes P et N de l'équation ci-dessus, nous somme les moments autour du barycentre du pendule pour obtenir l'équation suivante:

$$\mathbf{-PI\sin\theta - NI\cos\theta = I\ddot{\theta}} \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (2.8)$$

La combinaison de ces deux dernières équations, nous obtenons la deuxième équation dynamique:

$$\mathbf{(I + mI^2)\ddot{\theta} + mgI\sin\theta = -mI\ddot{x}\cos\theta} \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (2.9)$$

L'ensemble des équations définissant complètement la dynamique du pendule inversé sont:

$$\mathbf{(M + m)\ddot{x} + b\dot{x} + mI\ddot{\theta}\cos\theta - mI\dot{\theta}^2\sin\theta = F} \ldots\ldots\ldots\ldots\ldots\ldots\ldots (2.10)$$

$$\mathbf{(I + mI^2)\ddot{\theta} + mgI\sin\theta = -mI\ddot{x}\cos\theta} \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (2.11)$$

Ces deux équations sont non linéaires et doivent être linéarisée pour la plage de fonctionnement. Étant donné que le pendule est en cours de stabilisation à une position d'équilibre instable, qui est $\boldsymbol{\pi}$ radians de la position d'équilibre stable, cet ensemble d'équations doivent être linéarisé environ $\boldsymbol{\theta = \pi}$

Supposons que : $\boldsymbol{\theta = \pi + \phi}$;(où $\boldsymbol{\phi}$ représente un petit angle de la direction verticale vers le haut). Par conséquent $\boldsymbol{\cos\theta = -1; \sin\theta = -\phi}$, et $\boldsymbol{\dot{\theta}^2 = 0}$.

Après linéarisation les deux équations du mouvement deviennent (où u représente l'entrée):

$$\mathbf{(M + m)\ddot{x} + b\dot{x} - mI\ddot{\emptyset} = u} \ldots\ldots\ldots\ldots\ldots\ldots\ldots (2.12)$$

$$\mathbf{(I + mI^2)\ddot{\emptyset} - mgI\emptyset = mI\ddot{x}} \ldots\ldots\ldots\ldots\ldots\ldots\ldots (2.13)$$

Pour obtenir la fonction de transfert des équations du système linéarisées analytiquement, nous devons d'abord prendre la transformée de Laplace des équations du système.

Les transformées de Laplace sont:

$$(\boldsymbol{M}+\boldsymbol{m})\boldsymbol{S}^{2}\boldsymbol{X}(\boldsymbol{S})+\boldsymbol{bSX}(\boldsymbol{S})-\boldsymbol{mIS}^{2}\emptyset(\boldsymbol{S})=\boldsymbol{U}(\boldsymbol{S})\text{.....(2.14)}$$

$$(\boldsymbol{I}+\boldsymbol{mI}^{2})\boldsymbol{S}^{2}\emptyset(\boldsymbol{S})-\boldsymbol{mgI}\emptyset(\boldsymbol{S})=\boldsymbol{mIS}^{2}\boldsymbol{X}(\boldsymbol{S})\text{.....(2.15)}$$

Donc, on obtient:

$$\boldsymbol{X}(\boldsymbol{S})=\left(\frac{\boldsymbol{I}+\boldsymbol{mI}^{2}}{\boldsymbol{mI}}-\frac{\boldsymbol{g}}{\boldsymbol{S}^{2}}\right)\emptyset(\boldsymbol{S})\text{ .....(2.16)}$$

Lorsque, en remplaçant dans la deuxième équation donne:

$$(\boldsymbol{M}+\boldsymbol{m})\left[\frac{\boldsymbol{I}+\boldsymbol{mI}^{2}}{\boldsymbol{mI}}-\frac{\boldsymbol{g}}{\boldsymbol{S}^{2}}\right]\boldsymbol{S}^{2}.\emptyset(\boldsymbol{S})+\boldsymbol{b}\left[\frac{\boldsymbol{I}+\boldsymbol{mI}^{2}}{\boldsymbol{mI}}-\frac{\boldsymbol{g}}{\boldsymbol{S}^{2}}\right]\boldsymbol{S}.\emptyset(\boldsymbol{S})-\boldsymbol{mIS}^{2}\emptyset(\boldsymbol{S})=\boldsymbol{U}(\boldsymbol{S})\text{....(2.17)}$$

Le réagencement, la fonction de transfert est la suivante:

$$\frac{\emptyset(\boldsymbol{S})}{\boldsymbol{U}(\boldsymbol{S})}=\frac{\frac{\boldsymbol{mI}}{\boldsymbol{q}}\boldsymbol{S}^{2}}{\boldsymbol{S}^{4}+\frac{\boldsymbol{b}(\boldsymbol{I}+\boldsymbol{mI}^{2})}{\boldsymbol{q}}\boldsymbol{S}^{3}-\frac{\boldsymbol{mgL}(\boldsymbol{M}+\boldsymbol{m})}{\boldsymbol{q}}\boldsymbol{S}^{2}-\frac{\boldsymbol{bmgL}}{\boldsymbol{q}}\boldsymbol{S}}\text{ ..... (2.18)}$$

Où :

$$\boldsymbol{q}=(\boldsymbol{M}+\boldsymbol{m})(\boldsymbol{I}+\boldsymbol{mI}^{2})-(\boldsymbol{mI})^{2}$$

À partir de la fonction de transfert ci-dessus, on peut voir qu'il y a à la fois un pôle et un zéro à l'origine. Ceux-ci peuvent être annulés et la fonction de transfert devient:

$$\frac{\emptyset(\boldsymbol{S})}{\boldsymbol{U}(\boldsymbol{S})}=\frac{\frac{\boldsymbol{mI}}{\boldsymbol{q}}\boldsymbol{S}}{\boldsymbol{S}^{3}+\frac{\boldsymbol{b}(\boldsymbol{I}+\boldsymbol{mI}^{2})}{\boldsymbol{q}}\boldsymbol{S}^{2}-\frac{\boldsymbol{mgL}(\boldsymbol{M}+\boldsymbol{m})}{\boldsymbol{q}}\boldsymbol{S}-\frac{\boldsymbol{bmgL}}{\boldsymbol{q}}}\text{ ..... (2.19)}$$

La fonction de transfert peut donc être simplifiée comme suit : [8]

$$\frac{\emptyset(\boldsymbol{S})}{\boldsymbol{U}(\boldsymbol{S})}=\frac{\boldsymbol{mIS}}{\boldsymbol{q}.\boldsymbol{S}^{3}+\boldsymbol{b}(\boldsymbol{I}+\boldsymbol{mI}^{2})\boldsymbol{S}^{2}-\boldsymbol{mgL}(\boldsymbol{M}+\boldsymbol{m})\boldsymbol{S}-\boldsymbol{bmgL}}\text{ ..... (2.20)}$$

2.3 Modélisation du MCC

Un moteur à courant continu est une machine électrique. Il s'agit d'un convertisseur électromécanique permettant la conversion bidirectionnelle d'énergie entre une installation électrique parcourue par un courant continu et un dispositif mécanique, d'où l'énergie électrique est transformée en énergie mécanique. Dans la suite nous considérons le moteur à excitation séparée. Son schéma équivalent est donné par la figure suivante :

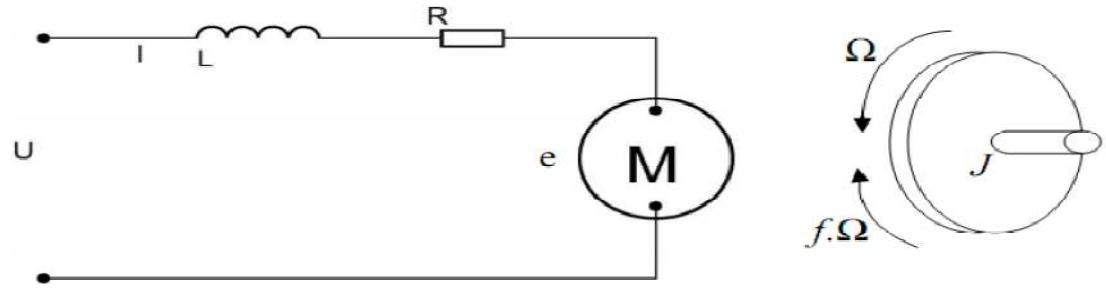


Figure 2.2. Schéma équivalent d'un moteur à courant continu.

Un moteur électrique à courant continu est régit par les équations physiques découlant de ses caractéristiques électriques, mécaniques et magnétiques. D'après la loi de Newton, combiné à des lois de Kirchhoff, On peut écrire les équations différentielles de premiers ordres suivantes :

$U(t) = Ri(t) + L\frac{di}{dt}(t) + e(t)$ ……(2.21)

$e(t) = K_e\Omega(t)$……(2.22)

D' après le principe fondamental de la dynamique on a :

J $\frac{d\Omega}{dt} = C_u - C_r$……(2.23)

$C_u = K_c i(t) - C_p$……(2.24)

$C_r = f\Omega(t)$……(2.25)

$u(t)$: Tension appliquée au moteur

$e(t)$: Force contre électromotrice

$i(t)$: Intensité traversant le moteur

$\Omega(t)$: Vitesse de rotation du rotor

K_e: Constante de vitesse

K_c: Constante de couple

C_p: Couple de pertes C_u: Couple moteur généré

f: Coefficient de frottement visqueux C_r : Couple résistant

J: Moment d'inertie de l'axe du rotor

2.3.1 Fonction de transfert du moteur

On passe en Laplace :

$$U(p) = Ri(p) + Lpi(p) + E \text{.....(2.26)}$$

$$E = K_e\Omega(p) \text{.....(2.27)}$$

$$Jp\Omega(\mathrm{p}) = C_u - C_r \text{.....(2.28)}$$

En combinant (2.26) et (2.27) on obtient :

$$U(p) = Ri(p) + Lpi(p) + K_e\Omega(p) \text{.....(2.29)}$$

En modifiant (2.28) on a :

$$Jp\Omega(\mathrm{p}) = K_c i(p) - \mathrm{f}\,\Omega(\mathrm{p}) - C_p \text{.....(2.30)}$$

On en déduit l'expression de $\Omega(\mathrm{p})$:

$$\Omega(\mathrm{p}) = \frac{K_c \mathrm{i}(\mathrm{p}) - C_p}{\mathrm{Jp} + \mathrm{f}} \text{ ... ... (2.31)}$$

On peut en sortir l'expression de i(p):

$$I(p) = \frac{f + Jp}{K_c}\left(\Omega(\mathrm{p}) + \frac{C_p}{\mathrm{f} + \mathrm{Jp}}\right) \text{ ... ... (2.32)}$$

On l'injecte à présent dans (2.26)

$$U(p) = \Omega(\mathrm{p})\left(\frac{(\mathrm{R} + \mathrm{Lp})(\mathrm{f} + \mathrm{Jp})}{K_c} + K_e\right) + \frac{R + Lp}{f + Jp} C_p \text{ ... ... (2.33)}$$

On suppose que le moment du couple de pertes (qui est vu comme une perturbation) soit négligeable devant le moment du couple électromagnétique $(K_c i(t))$ on peut alors prend C_p nul pour simplifier le système.

On obtient donc:

$$U(p) = \Omega(\mathrm{p})\left(\frac{(\mathrm{R}+\mathrm{Lp})(\mathrm{f}+\mathrm{Jp})}{K_c} + K_e\right) \ldots\ldots\ldots\ldots (2.34)$$

La fonction de transfert cherchée H(p) est entre la tension entrant dans le moteur U(p) et la vitesse de sortie *Ω(p) :* [9],[10]

$$H(p) = \frac{\Omega(p)}{U(p)} = \frac{K_c}{(\mathrm{R}+\mathrm{Lp})(\mathrm{f}+\mathrm{Jp}) + K_c K_e} \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (2.35)$$

On peut établir le modèle mathématique de la réponse en vitesse du moteur électrique qui est donné par la figure suivante (figure 2.3):

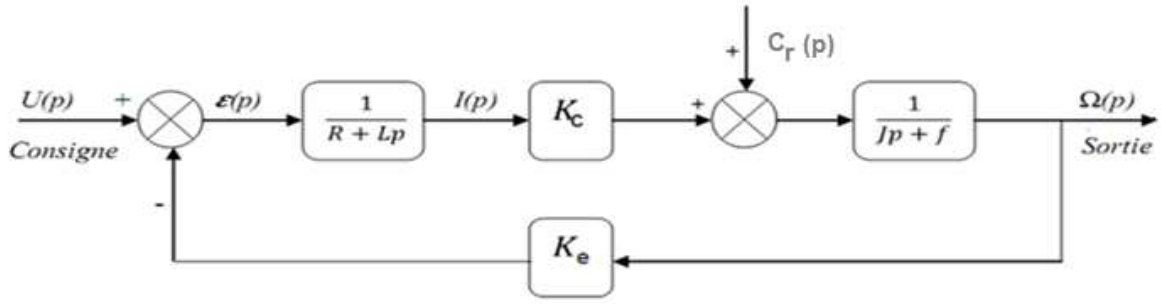


Figure 2.3. Schéma bloc du Modèle de moteur électrique en vitesse.

D'après l'équation (2.35), le système modélisé soit du second ordre, lorsque l'inductance interne est négligeable devant la résistance interne (ce qui est généralement le cas) il s'apparente à un système du premier ordre. On observe bien sur la (figure 2.3) ci-dessus que le moteur change de vitesse de rotation pour chaque valeur de la tension d'entrée, d'où la vitesse de rotation est proportionnelle à la tension d'entrée.

On peut établir le modèle mathématique de la réponse en position du moteur électrique qui est donné par la figure suivante (figure 2.4):

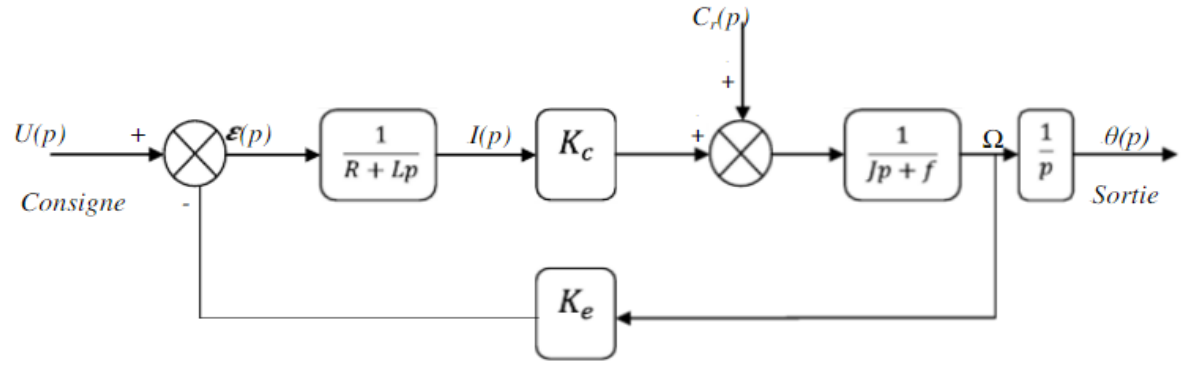


Figure 2.4. Schéma bloc du Modèle de moteur électrique en position.

2.4 La régulation par PID

2.4.1 Introduction

Le régulateur le plus utilisé dans l'industrie est le régulateur PID (proportionnel intégral dérivé). Il permet de régler à l'aide de ses trois actions les performances d'un système en boucle fermée.

Les régulateurs PID répondent à plus de 95% des besoins industriels et le nombre de régulateurs installés dans une usine pétrolière, par exemple, se compte par milliers. Malheureusement, malgré l'expérience acquise au fil des ans, les valeurs choisies pour les paramètres des actions P, I et D ne sont pas toujours satisfaisantes, ni adaptées au processus à régler.[11]

Le régulateur PID est une simple implémentation de retour d'information (Feedback). Il a la capacité d'éliminer la compensation de l'état d'équilibre grâce à l'action intégrale, et il peut anticiper le futur grâce à une action dérivée, [12].

En 1942, Ziegler et Nichols [13] ont proposé deux démarches permettant de trouver facilement les paramètres optimums pour une installation donnée. Au fil des ans, les propositions de Ziegler et Nichols ont été adaptées ou modifiées selon les besoins.

En 1963, Horowitz [14] a ajouté un degré de liberté supplémentaire au régulateur PID afin de mieux contrôler les dépassements obtenus lors d'une réponse indicielle. Ce

nouveau degré de liberté consiste, en particulier, à ne réinjecter vers le terme proportionnel qu'une partie du signal de sortie.

Au début des années 1990 et dans le but de fournir des règles d'ajustement simples mais plus performantes que celles de Ziegler-Nichols, Aström [15] et ses collaborateurs ont analysé le comportement dynamique d'un grand nombre de processus. Cette analyse a conduit à l'établissement de tableaux servant aux calculs des paramètres des actions P, I et D à partir de mesures simples.

2.4.2 Correction des systèmes

Le rôle le plus important des correcteurs est la commande des processus pour avoir une sortie désirée. Le correcteur est sélectionné de façon à ce que les nouveaux caractéristiques du système en boucle fermée répondent à des critères de performances exigés. Les critères les plus importants dans une commande sont les marges de stabilité, le dépassement, le temps de réponse et l'erreur en régime permanent (l'erreur statique).

On peut distinguer deux approches essentielles pour la correction des systèmes, la première est basée seulement sur la valeur désirée à la sortie du processus (commande en boucle ouverte) et l'autre sur la valeur désirée mais aussi sur la sortie du système réellement obtenue (commande en boucle fermée).

Dans notre cas on utilise la régulation en boucle fermée.

2.4.3 Régulation en boucle fermée

Le schéma classique d'une correction en boucle fermée est présenté dans la figure (2.5). Le régulateur, dont la fonction de transfert est désignée par $G_c(s)$, est situé en amont du système à régler $G_a(s)$.

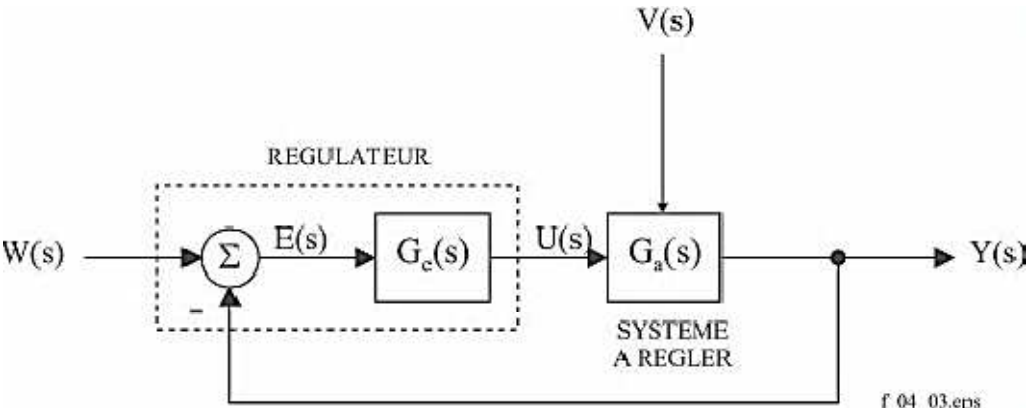


Figure 2.5. Schéma fonctionnel d'un système asservi mono-variable (correction en boucle fermée).

Le système à régler $G_a(s)$ comprend, outre le processus, l'amplificateur de puissance, l'actionneur, le capteur et l'électronique de traitement de la mesure associée. Appliquée au système à régler, la commande u(t) provoque donc une modification de la grandeur réglée y(t). Le régulateur en tenant compte pour former u(t), on constate que y(t) apparaît :

- à l'origine de l'action entreprise par le régulateur.
- comme conséquence de cette action [16].

2.4.4 Eléments constitutifs d'une boucle de régulation

Une boucle de régulation doit comporter au minimum les éléments suivants :

- Une variable de sortie: qui doit être contrôlée.
- Un capteur pour mesurer la variable de sortie.
- Un régulateur pour calculer l'action de contrôle. C'est-à-dire la valeur de la variable manipulée à partir de l'erreur.
- Un actionneur pour appliquer physiquement l'action de contrôle dont le but de changer ou de modifier le procédé, et de le ramener à son point de consigne.

2.4.5 Description des régulateurs PID

Un régulateur PID remplit essentiellement trois fonctions :

- Il fournit un signal de commande u(t) en tenant compte de l'évolution du signal de sortie y(t) par rapport à la consigne w(t).
- Il élimine l'erreur statique grâce au terme intégrateur.
- Il anticipe les variations de la sortie grâce au terme dérivateur. [11]

2.4.6 La commande PID en régulation de vitesse et de position

La commande proportionnel-intégral-dérivé (PID) est insérée dans la chaine directe de l'asservissement, en série avec le processus. Ce régulateur élabore à partir du signal d'erreur ε(t) une commande u(t) en fonction de trois actions proportionnelle, intégrale, dérivée. [17]

$$\mathrm{U(t)} = \mathrm{K_p}\varepsilon(\mathrm{t}) + \frac{1}{\mathrm{T_i}}\int_0^{\mathrm{t}} \varepsilon(\mathrm{t})\,\mathrm{dt} + \mathrm{T_d}\frac{\mathrm{d}\varepsilon(\mathrm{t})}{\mathrm{dt}} \text{.....(2.36)}$$

$$\mathrm{U(t)} = \mathrm{K_p}\varepsilon(\mathrm{t}) + \mathrm{K_i}\int_0^{\mathrm{t}} \varepsilon(\mathrm{t})\,\mathrm{dt} + \mathrm{K_d}\frac{\mathrm{d}\varepsilon(\mathrm{t})}{\mathrm{dt}} \text{.....(2.37)}$$

$$\overset{TL}{\rightarrow} \mathbf{U(p)} = \mathbf{K_p}\boldsymbol{\varepsilon}(\mathbf{t}) + \mathbf{K_i}\frac{\mathbf{d}\boldsymbol{\varepsilon}(\mathbf{p})}{\mathbf{p}} + \mathbf{K_d}\boldsymbol{\varepsilon}(\mathbf{p})\mathbf{p} \text{ .....(2.38)}$$

K_p : Gain d'action proportionnelle.

K_i=1/T_i : Gain d'action intégrale.

K_d=T_d : Gain d'action dérivée.

T_i : Constant de temps, dite temps d'action intégrale.

T_d : Constant de temps, dite temps d'action dérivée.

Le régulateur PID est donc conçu dans le domaine temporel comme la somme des trois actions. On obtient alors un asservissement composée d'un terme proportionnel, un terme intégral et un terme dérivé, mises en parallèle, on parle d'asservissement PID:

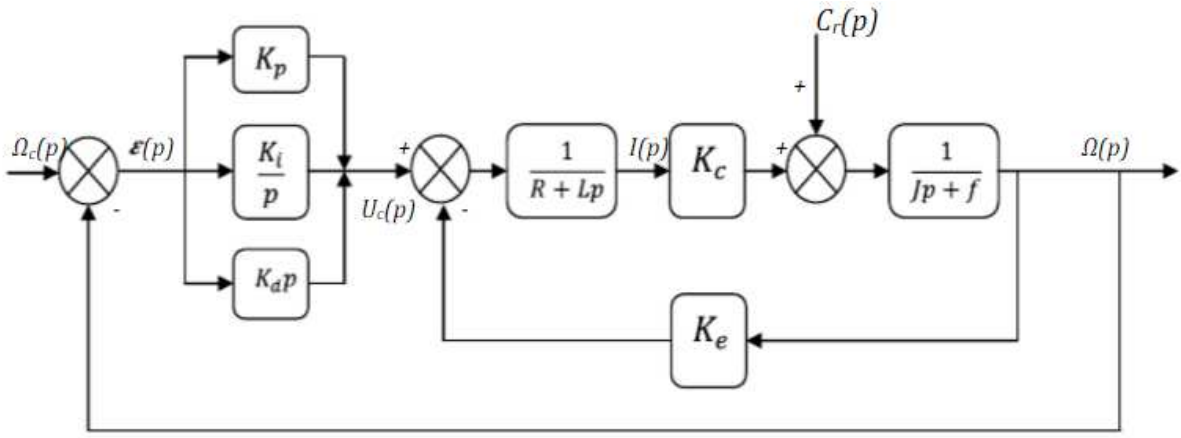


Figure 2.6. Schéma bloc du correcteur PID en régulation de vitesse.

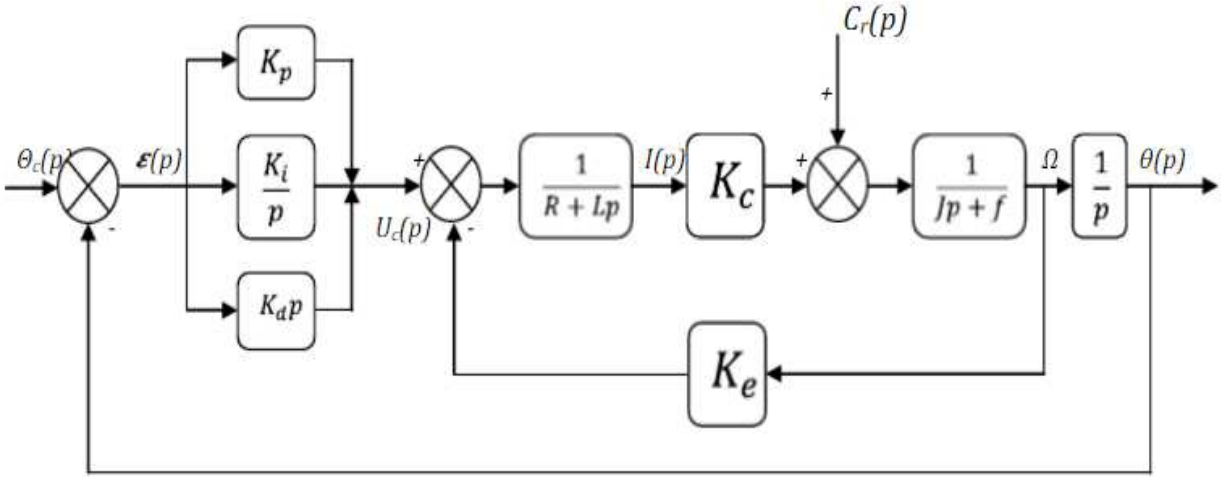


Figure 2.7. Schéma bloc du correcteur PID en régulation de position.

2.5 L'implémentation d'un PID sur Arduino

Le régulateur PID est défini par l'équation différentielle suivante :

$$U_{PID}(t) = K_P \times e(t) + K_i \times \int_0^t e(\tau)d\tau + K_d \times \frac{d}{dt}e(t) \quad \text{.....(2.39)}$$

L'équation de contrôle PID numérique peut être exprimée en de diverses manières, mais une formulation générale est donnée par l'équation suivante :

$$PID = K_P \times erreur + K_i \times \sum(erreur \times \Delta t) + K_d \times \frac{(erreur - erreur_{précédent})}{\Delta t} \text{.....(2.40)}$$

L'implémentation du régulateur PID sur la carte Arduino se fera de la manière suivante:

PID :

Error = Setpoint − Actual

Integral = Integral + (Error ∗dt)

Derivative = (Error − Previous_ error)/ dt

Drive = (Error ∗kP) + (Integral ∗ kI) + (Derivative ∗kD)

Previous_ error = Error

wait (dt)

GOTO PID

2.6 Conclusion

Dans ce chapitre nous avons présenté la modélisation d'un pendule inversé et moteur à courant continu, et le principe général de la régulation, ainsi que les caractéristiques de régulateur PID. Pour réaliser ce dernier, nous avons utilisé une carte Arduino comme élément principal, qui sera détaillée dans le prochain chapitre.

Chapitre 3 Partie électronique et hardware

3.1 Introduction

Dans ce chapitre, on va représenter les différentes cartes électroniques et la partie hardware de notre projet.

3.2 Cartes Arduino

3.2.1 Présentation de la Carte

Arduino est un ensemble d'outils matériel et logiciel pour le prototypage électronique et l'apprentissage de la programmation des microcontrôleurs. Les schémas électroniques des cartes Arduino et les codes sources pour la partie développement des programmes, sont distribués librement et téléchargeables gratuitement. L'environnement Arduino a été conçu pour être utilisé facilement par les débutants sans expérience de la programmation ou de connaissances en électronique [18].

Figure 3.1. Exemple de carte électronique.

3.2.2 Différents modèles d'Arduino

Il existe plusieurs types de carte Arduino, chaque carte a sa propre dimension, taille de mémoire, intensité du courant …etc. voir les Tableau 3.1 et Tableau 3.2 ; [19]

	UNO	**UNO R3**	**PRO**	**LEONARDO**
Dimensions	53 mm × 68 mm	53 mm × 68 mm	53 mm × 68 mm	53 mm × 68 mm
Microcontrôleur	ATmega328	ATmega328	ATmega328	ATmega32u4
Tension de fonctionnement	5V	5V	5V	5V
Tension d'alimentation (recommandée)	7V-12V	7V-12V	7V-12V	7V-12V
Tension d'alimentation (limites)	6V-20V	6V-20V	6V-20V	6V-20V
Broches E/S numériques	14 (dont 6 disposent d'une sortie PWM)	14 (dont 6 disposent d'une sortie PWM)	14 (dont 6 disposent d'une sortie PWM)	14 (dont 6 disposent d'une sortie PWM)
Broches d'entrée analogique	6	6	6	12
Intensité maxi disponible par broche E/S (5V)	40mA	40mA	40mA	40mA

Intensité maxi disponible par broche E/S (3.3V)	50mA	50mA	50mA	50mA
Mémoire programme Flash	16KB(ATmega 168) ou 32KB (ATmega328) dont 2KB	16KB(ATmega 168) ou 32KB (ATmega328) dont 2KB	16KB(ATmeg a168) ou 32KB (ATmega328) dont 2KB	32KB (ATmega32u4) dont 4KB
Mémoire SRAM (mémoire volatile)	1KB (ATmega 168) ou 2KB (ATmega328)	1KB (ATmega 168) ou 2KB (ATmega328)	1KB (ATmega 168) ou 2KB (ATmega328)	2.5KB (ATmega32u4)
Mémoire EEPROM (mémoire non volatile)	512 bytes (ATmega168) ou 1KB (ATmega328)	512 bytes (ATmega168) ou 1KB (ATmega328)	512 bytes (ATmega168) ou 1KB (ATmega328)	KB (ATmega32u4)
Vitesse d'horloge	16 MHz	16 MHz	16 MHz	16 MHz

Tableau 3.1. Caractéristiques Cartes Arduino «UNO-UNO R3-PRO-LEONARDO».

	PRO mini	**MEGA**	**DUE**
Dimensions	33 mm × 18 mm	53 mm × 101 mm	53 mm × 101 mm
Microcontrôleur	ATmega168	ATmega2560	AT91SAM3X8E
Tension de fonctionnement	3.3V ou 5V	5V	3.3V
Tension d'alimentation	3.35V-12V	7V-12V	7V-12V

(recommandée)	(3.3V model)		
Tension d'alimentation (limites)	5V-12V (5V model)	6V-20V	6V-20V
Broches E/S numériques	14 (dont 6 disposent d'une sortie PWM)	54 (dont 15 disposent d'une sortie PWM)	54 (dont 12 disposent d'une sortie PWM)
Broches d'entrée analogique	6	16	12
Intensité maxi disponible par broche E/S (5V)	40 mA	40 mA	130 mA
Intensité maxi disponible par broche E/S (3.3V)	50 mA	50 mA	800 mA
Mémoire programme Flash	16 KB(ATmega168) dont 2KB sont utilisé par le bootloader	256 KB dont 8 KB sont utilisé par le bootloader	512KB (application)
Mémoire SRAM (mémoire volatile)	1KB (ATmega168)	8 KB	96 KB
Mémoire EEPROM (mémoire non volatile)	512 bytes (ATmega168)	4 KB	--------
Vitesse d'horloge	8 MHz	16 MHz	84 MHz

Tableau 3.2. Caractéristiques Cartes Arduino «PRO mini-MEGA-DUE».

3.2.3 Constitutions de la carte Arduino

Dans notre projet nous avons utilisé une carte Arduino Méga qui est représentée dans La figure ci-dessous :

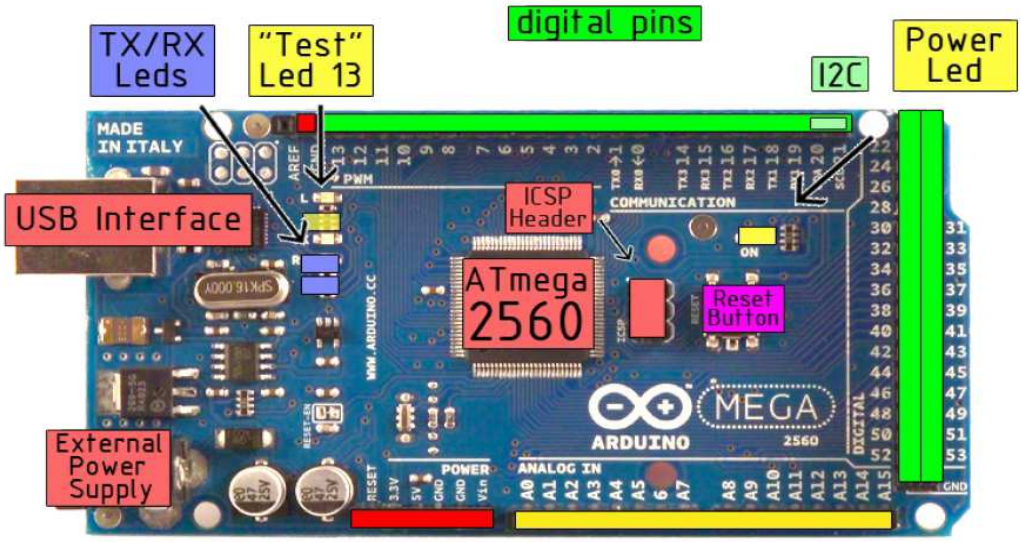


Figure 3.2. Carte Arduino Méga.

a Alimentation de la carte

La carte Arduino Méga peut-être alimentée soit via la connexion USB (qui fournit 5V jusqu'à 500mA) ou à l'aide d'une alimentation externe. La source d'alimentation est sélectionnée automatiquement par la carte.

L'alimentation externe (non-USB) peut être soit un adapteur secteur (pouvant fournir typiquement de 3V à 12V sous 500mA) ou des piles (ou des accus).

La carte peut fonctionner avec une alimentation externe de 6 à 20 volts. Cependant, si la carte est alimentée avec moins de 7V, la broche 5V pourrait fournir moins de 5V et la carte pourrait être instable. Si on utilise plus de 12V, le régulateur de tension de la carte pourrait chauffer et endommager la carte. Aussi, la plage idéale recommandée pour alimenter la carte Méga est entre 7V et 12V.

Pour alimenter la carte par un port USB, il faut assurer la protection du port et pour cela la carte Arduino Méga intègre un fusible réinitialisable qui protège le port USB de votre ordinateur contre les surcharges en intensité (le port USB est généralement limité à 500mA en intensité). Bien que la plupart des ordinateurs aient

leur propre protection interne, le fusible de la carte fournit une couche supplémentaire de protection. Si plus de 500mA sont appliqués au port USB, le fusible de la carte coupera automatiquement la connexion jusqu'à ce que le court-circuit ou la surcharge soit stoppé.

b Connecteurs des sources de tension (broches d'alimentation)

- **Reset** : qui a la même fonction de réinitialisation que le bouton de remise à zéro (reset). Pour forcer un redémarrage avec cette broche, il faut la forcer momentanément à 0 V (état bas). Typiquement, cette broche est utilisée pour ajouter un bouton de réinitialisation sur le circuit qui bloque celui présent sur la carte.
- **3.3V** : Une alimentation de 3.3V fournie par le circuit intégré FTDI (circuit intégré faisant l'adaptation du signal entre le port USB de votre ordinateur et le port série de l'ATmega) de la carte, ceci est intéressant pour certains circuits externes nécessitant cette tension au lieu du 5V, L'intensité maximale disponible sur cette broche est de 50mA.
- **5V** : La tension régulée utilisée pour faire fonctionner le microcontrôleur et les autres composants de la carte (la "tension régulée" obtenue à l'aide d'un composant appelé un régulateur et qui est intégré à la carte Arduino). Le 5V régulé fourni par cette broche peut donc provenir soit de la tension d'alimentation VIN via le régulateur de la carte, ou bien de la connexion USB (qui fournit du 5V régulé) ou de tout autre source d'alimentation régulée.
- **GND** : Broche de masse (ou 0V).
- **VIN** : La tension d'entrée positive lorsque la carte Arduino est utilisée avec une source de tension externe (à distinguer du 5V de la connexion USB ou autre source 5V régulée). Vous pouvez alimenter la carte à l'aide de cette broche, ou si l'alimentation est fournie par le jack d'alimentation, accéder à la tension d'alimentation sur cette broche.

c Entrées analogiques

La carte Méga dispose de 16 entrées analogiques (numérotées de A0 à A15) permettent de mesurer une tension pour pouvoir en utiliser les valeurs dans les programmes. Ces entrées sont désignées comme étant de type analogique, et elles le sont par défaut, mais vous pouvez également les exploiter en tant qu'entrées numériques ou même comme sorties numériques, ces broches mesurent entre le 0V (valeur 0) et le 5V (valeur 1023).

d Entrées et sorties numériques

Les 54 broches numériques de la carte Méga (numérotées des 0 à 53) peut être utilisée soit comme une entrée numérique, soit comme une sortie numérique, Ces broches fonctionnent en 5V et qu'elles peuvent être activées et désactivées depuis le programme. Si vous activez une de ces broches, la tension présentée sera à 5V. Si vous désactivez la broche, la tension sera à 0V. 15 broches sont configurées en sortie PWM.

Les broches 0, 1, 19, 18, 16, 17,14, 15 sont réservées à la communication série, recevoir (RX) et transmettre (TX) les données séries.

Les Broches 2, 3, 18,19, 20, 21 peuvent être configurées pour déclencher une interruption sur une valeur basse, sur un front montant ou descendant, ou sur un changement de valeur.

e Les LEDs (visualisation)

Il y a une LED incluse dans la carte connectée à la broche 13. Lorsque la broche est au niveau HAUT, la LED est allumée, lorsque la broche est au niveau BAS, la LED est éteinte.et va servir pour tester le matériel.

Les deux autre LEDs servent à visualiser l'activité sur la voie série, une pour l'émission et l'autre pour la réception.

f Bouton de remise à zéro

Une pression sur cet interrupteur envoie une impulsion logique sur la broche RESET du microcontrôleur, ce qui le forcer à redémarrer en effaçant la mémoire. Sachez que les programmes stockés sur la carte ne sont pas effacés, parce qu'ils sont stockée dans

la mémoire flash, c'est-à-dire que le contenu n'est pas perdu en cas de mise hors tension.

g Microcontrôleur

Le microcontrôleur est l'élément principal de la carte Arduino, un microcontrôleur est littéralement un petit ordinateur contenu dans une seule puce. Il embarque même plus de fonctions que les premiers ordinateurs personnels, puisqu'il réunit un processeur central, huit kilo-octet de mémoire vive (SRAM) pour stocker les données, quatre kilo-octets de mémoire en lecture seule effaçable (EEPROM) ou en mémoire flash, dans lesquels vous pouvez stocker les programmes, ainsi que des broches d'entrée et de sortie. Ces broches permettent de relier de microcontrôleur à vos composants électroniques.

Le microcontrôleur utilisé sur la carte Arduino Méga est un microcontrôleur ATMega2560, C'est un microcontrôleur ATMEL de la famille AVR.[20]

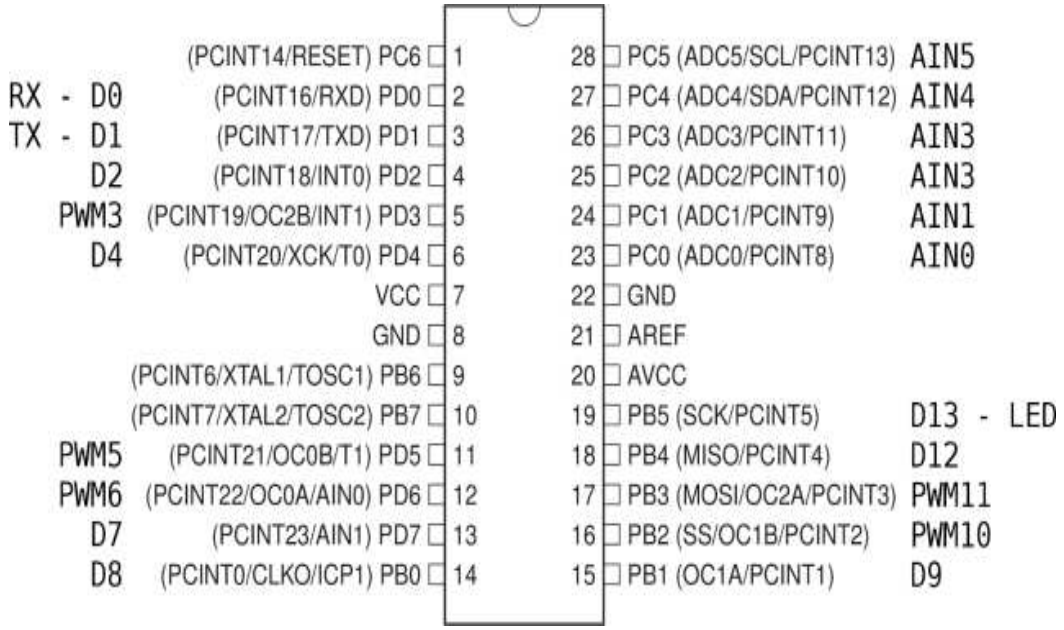


Figure 3.3. Boitier de l'ATMega2560.

3.2.4 Programmation de la carte

La programmation est un code qui va dicter des ordres au microcontrôleur, on la télécharge dans ce dernier pour qu'il puisse suivre les consignes dictées par le programme. Pour ce faire on utilise un logiciel nommé Arduino qui peut vérifier et compiler plusieurs programmes, l'avantage est de pouvoir vérifier le programme édité avant de le compiler vers le microcontrôleur.

Ce programme est codé en langage C, c'est un langage de programmation impératif pour la carte. Son avantage est qu'il intègre des fonctions préinstallées dans une seule ligne de code grâce à des librairies. Leur programmation est décomposée en trois étapes comme nous montre (Figure 3.4) : [21]

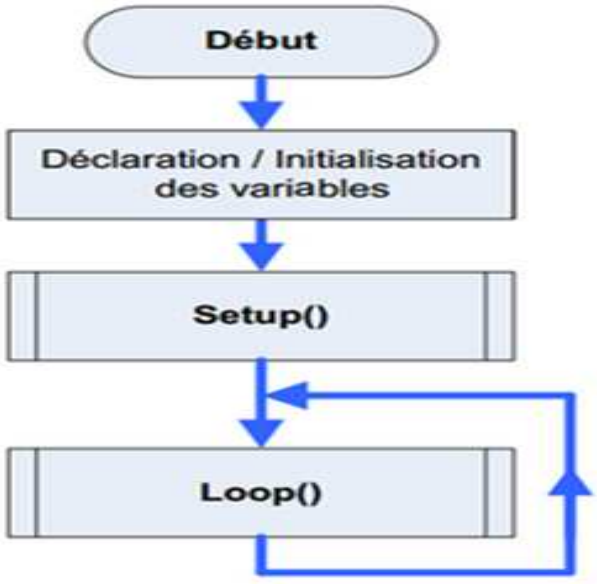


Figure 3.4. Les Etapes de Programmation sous Arduino en Langage C.

L'interface de l'IDE Arduino est plutôt simple (voir Figure 3.5), Elle offre une interface minimale et épurée pour développer un programme sur les cartes Arduino. Il est doté d'un éditeur de code avec **coloration syntaxique** (1) et d'une **barre d'outils rapide** (2). Ce sont les deux éléments les plus importants de l'interface, c'est ceux que l'on utilise le plus souvent. On retrouve aussi une **barre de menus** (3) plus classique qui

est utilisée pour accéder aux fonctions avancées de l'IDE. Enfin, une **console** (4) affichant les résultats de la compilation du code source, des opérations sur la carte, etc.

Figure 3.5. IDE Arduino.

3.2.5 Fonctionnalité de base

a Les entrées/sorties

Le langage Arduino vient avec un nombre important de **fonction de base** permettant d'interagir avec son **environnement**. Les fonctions les plus utilisée sont les fonctions d'entrée/sorties. Ce sont elles qui permettent **d'envoyer** ou de **mesurer** une tension sur une des broches de la carte.

Dans un premier temps, avant d'effectuer une mesure ou d'envoyer une commande. Il est nécessaire de définir la **direction** des broches utilisées. Pour cela on fait appel à la fonction :

- ✓ ***pinMode*** *:* Cette commande qui va dans la fonction **setup ()**, est utilise pour définir la direction d'une broche d'entrée / sortie numérique.

Pour mesurer ou envoyer une commande il faut les quatre fonctions suivants:

- ***digitalRead(pin)*** : mesure une donnée numérique sur une des broches, la broche en question doit être réglée en entrée.
- ***digitalWrite(pin, value)*** : écrit une donnée numérique sur une des broches, la broche concernée doit être réglée en sortie. Le paramètre ***value*** doit être égal à ***HIGH*** (état 1 soit 5V) ou ***LOW*** (état 0 soit 0V).
- ***analogRead(pin)*** : mesure une donnée analogique sur une des broches (compatible seulement), la broche doit être réglée sur entrée.
- ***analogWrite(pin, value)*** : écrit une donnée sous forme de **PWM** sur une des broches (compatible uniquement), la broche doit être réglée en sortie. Le paramètre **value** doit être compris dans l'intervalle *[0;255].*

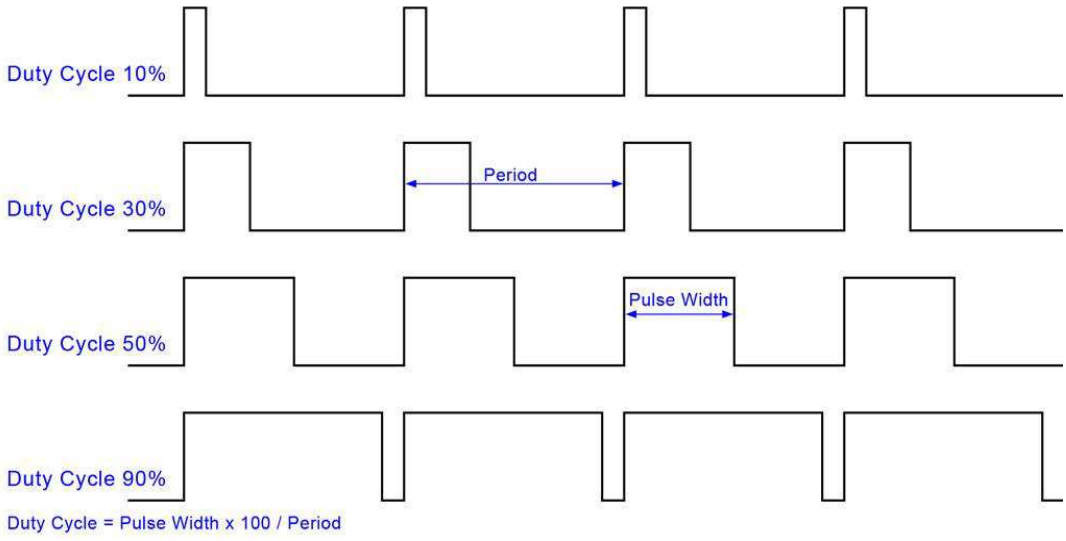


Figure 3.6. Signal PWM.

b La gestion du temps

Le langage Arduino fournit quelques fonctions permettant de gérer le temps, Il est possible d'insérer une **pause** dans son programme pendant un instant. Pour cela, on utilise les fonctions ***delay*** et ***delayMicroseconds*** qui insère une pause suivant le paramètre passé (en millisecondes pour l'un, en microseconde pour l'autre). Cependant ces fonctions **bloquent** le microcontrôleur, on ne peut alors plus effectuer aucune action.

En plus d'insérer une pause, il est possible de **mesurer le temps**. De la même manière que les fonctions de délai, on utilise les fonctions ***millis*** et ***micros*** qui donnent le nombre de **milliseconde** (respectivement **microseconde**) depuis le lancement de la carte.

c Les Interruptions

Les interruptions sont des portions de code (fonctions) appelé lorsqu'un événement (interne ou externe) survient et à besoin d'être traité sur le champ. il est prioritaire par rapport au reste du code. Vu qu'il est possible de mesurer les événements ponctuellement (via les fonctions d'entrées/sorties) on utilise généralement les interruptions pour du code critique (arrêt d'urgence par exemple) ou des événements non-ponctuels (transmissions de données depuis un ordinateur par exemple).

Les interruptions sont utilisables sur les broches compatibles seulement (broches 2, 3, 18, 19, 20, 21 sur l'Arduino Méga). Pour choisir la fonction et la broche utilisée pour l'interruption, on utilise la fonction ***attachInterrupt***. On peut utiliser ***detachInterrupt*** pour supprimer l'interruption. Il est possible de partir en interruptions sur 4 types d'événements :

-***LOW*** : Lorsque la broche est à l'état 0 (0V)

- ***RISING*** : Lorsque la broche passe de l'état 0 (0V) à l'état 1 (5V) (front montant).

- ***FALLING*** : Lorsque la broche passe de l'état 1 (5V) à l'état 0 (0V) (front descendant).

- ***CHANGE*** : Lorsque la broche change d'état (front montant et front descendant). [22]

3.3 Gyroscope et accéléromètre

Gyroscope est un dispositif qui est utilisé pour mesurer la vitesse angulaire. Ce type de dispositif peut être utilisé pour mesurer la rotation de la terre. Il fonctionne plus moins la même chose avec un accéléromètre, mais il donne des informations plus précises, par exemple pour savoir exactement comment un objet est orienté. Ils mesurent la vitesse angulaire en unités de rotations par minute (RPM), ou en degrés par seconde (° / s).

Un accéléromètre est un capteur qui fixé à un mobile ou tout autre objet, permet de mesurer l'accélération linéaire. [23]

3.3.1 Gyroscope et accéléromètre 3 axes – MPU-6050

Le module MPU-6050 embarque un microsystème électromécanique (gyroscope 3 axes et un accéléromètre 3 axes) au sein du même circuit DMP pour vous fournir les informations de navigation les plus précises et diriger sans erreur votre robot mobile, il contient 16 bits analogique au matériel de conversion numérique pour chaque canal. Par conséquent, il saisit les coordonnées x, y et z canal en même temps. Le capteur utilise le bus I2C à l'interface avec l'Arduino.[24]

Ce composant peut-être utilisé dans tous les projets d'objets volants ou de robots nécessitant un asservissement d'équilibre, comme le robot gyropode. [25]

Figure 3.7. MPU-6050.

3.3.2 Spécifications techniques du module gyroscope 3 axes – MPU-6050

- Tension d'alimentation : 2,3 – 3,4 V
- Consommation: 3,9 mA maxi
- Accéléromètre 3 axes avec intervalle de pleine échelle programmable de ±2g, ±4g, ±8g et ±16g
  - Tolérance de calibration : ±3%
- Gyroscope 3 axes avec sensibilité jusqu'à 131 LSB/(deg/s) et intervalle de pleine échelle programmable de ±250, ±500, ±1000, et ±2000 deg/s
  - Tolérance de calibration: ±3%
- Interface I2C
- Capteur de température intégré
- Cavaliers sélectionnables sur CLK, FSYNC et AD0
- Température de service: -40°C à +85°C
- Dimensions : 25,5 x 15,2 x 2,48 mm
- Digital Motion Processing (DMP) permettant de décharger le processeur principal de votre projet pour réaliser des calculs de mouvement complexes. [24]

3.3.3 Le protocole I2C

I2C (signifie : Inter-Integrated Circuit, en anglais) est un bus informatique, il permet de relier facilement un microprocesseur et différents circuits.

Dans notre projet en relie l'arduino Méga (Maitre) avec le capteur MPU-6050 (esclave) par le protocole I2C.

L'Arduino Méga utilisé les connecteurs 20 pour SDA (les données) et 21 pour SCL (l'horloge).

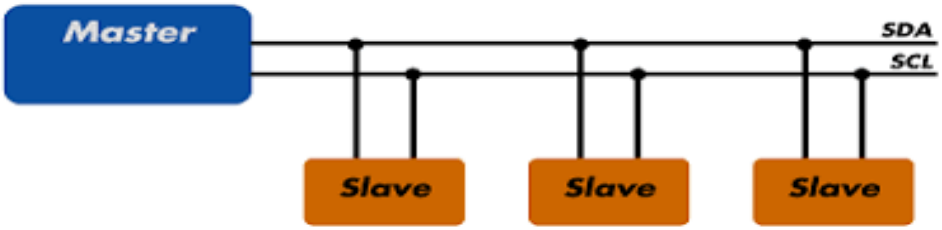


Figure 3.8. Schéma d'un protocole I2C.

L'intérêt est que I2C permet de brancher plusieurs composants à ces deux pins. Par exemple, sur le schéma ci-dessus, trois composants slaves sont reliés au master. Les deux fils SDA et SCL sont donc appelés le bus I2C. Chaque périphérique branché sur le bus I2C a sa propre adresse qui permet de l'identifier (comme des numéros de maisons dans une rue).

Le bus I2C permet d'envoyer et de recevoir des données du capteur. On parle d'écriture et de lecteur. Pour chaque opération (écriture ou lecture), il faut préciser l'adresse du capture et un registre. Ce registre est un numéro (on le notera en hexadécimal) qui permettra de signifier ce que l'on veut lire ou écrire, par exemple lire la coordonnée X de l'accélération (c'est une lecture) ou mettre le capteur en veille (c'est une écriture). [26]

3.4 Driver Moteur L298

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 contrôle de robots et les ensembles mécanisés. Il peut contrôler deux moteur courant continu ou un moteur pas-à-pas 4 fils 2 phases. 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, solénoïdes, 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. [27]

Le pont en H permet de réaliser 2 fonctions qui sont d'inverser le sens de rotation du moteur, en inversant la tension aux bornes du moteur et la variation de la vitesse du moteur en modulant la tension aux bornes du moteur.

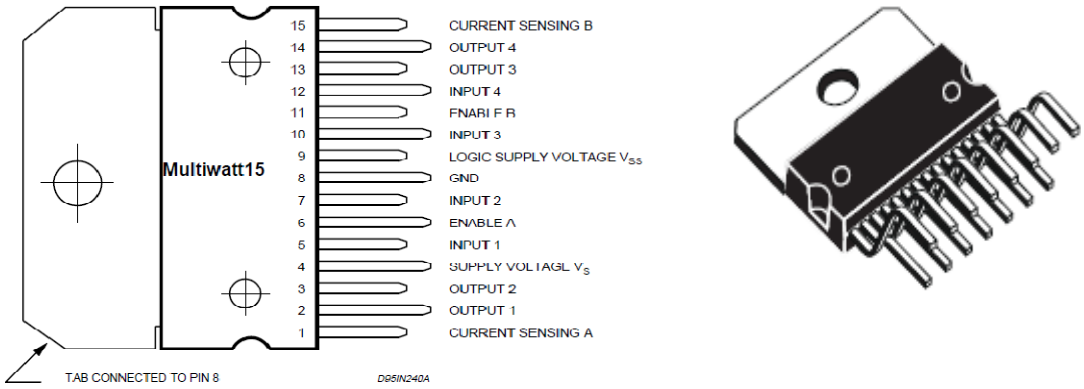


Figure 3.9. Brochage du circuit intégré L298*,* Hacheur L298.

3.4.1 Les caractéristiques du L298

- ✓ Intensité maximale : 2A par pont
- ✓ Alimentation de puissance de 5.5V à 50V
- ✓ Type de boîtier : Multiwatt15 ;
- ✓ Dissipation puissance total : 25W
- ✓ Trois entrées par pont : In1, In2 et ENABLE.

a Activation moteur

- ENA raccordés à un niveau haut (HIGH) activée le MOTEURA.
- ENB raccordés à un niveau haut (HIGH) activera MOTEURB.

Si vous voulez contrôler la vitesse, vous pouvez connecter ENA(ENB) sur une sortie PWM.

b Rotation Moteur A

- IN1 raccordés à 5V et IN2 à GND MOTEURA tournera dans le sens horlogique.
- IN1 raccordés à GND et IN2 à 5V MOTEURA tournera dans le sens Anti-horlogique.

c Rotation Moteur B

- IN3 raccordés à 5V et IN4 a GND MOTEURB tournera dans le sens horlogique.
- IN3 raccordés à GND et IN4 à 5V MOTEURB tournera dans le sens Anti-horlogique. [28]

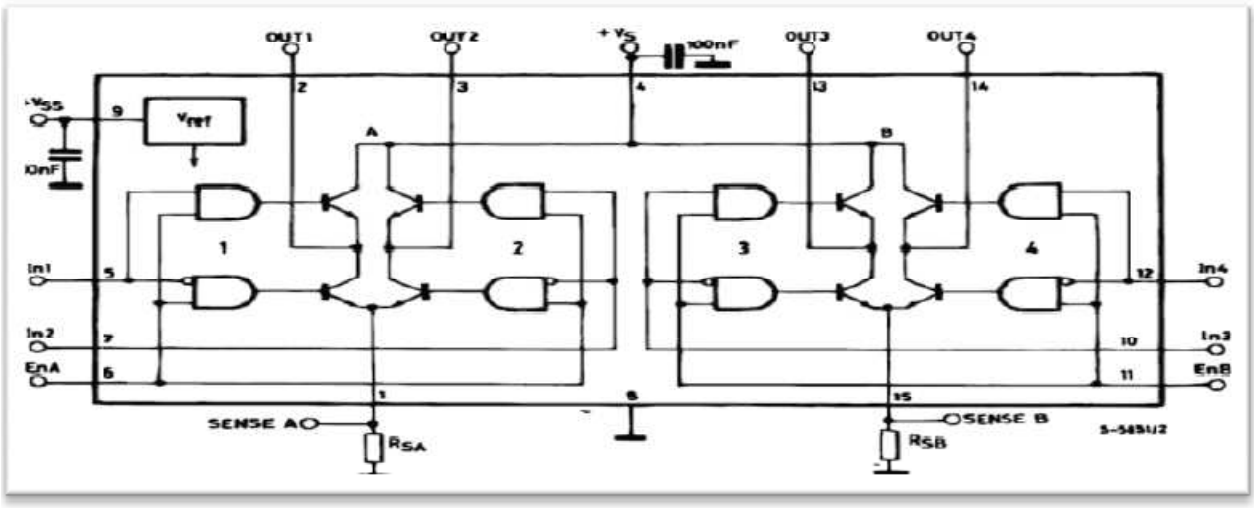


Figure 3.10. Diagramme bloque du L298.

3.5 Partie Hardware de projet

3.5.1 Outils mécaniques et électriques

Les matériels globales se composent des éléments suivants :

a Carte Arduino Méga

La carte Arduino Méga utilisée pour commander le robot (implémenter le correcteur PID, lire les données de capteur, envoyer les données vers les moteurs, etc…), et on utilise cette carte parce que nous avons besoin de quatre interruptions pour les encodeurs (on a 360 impulsions par tour)

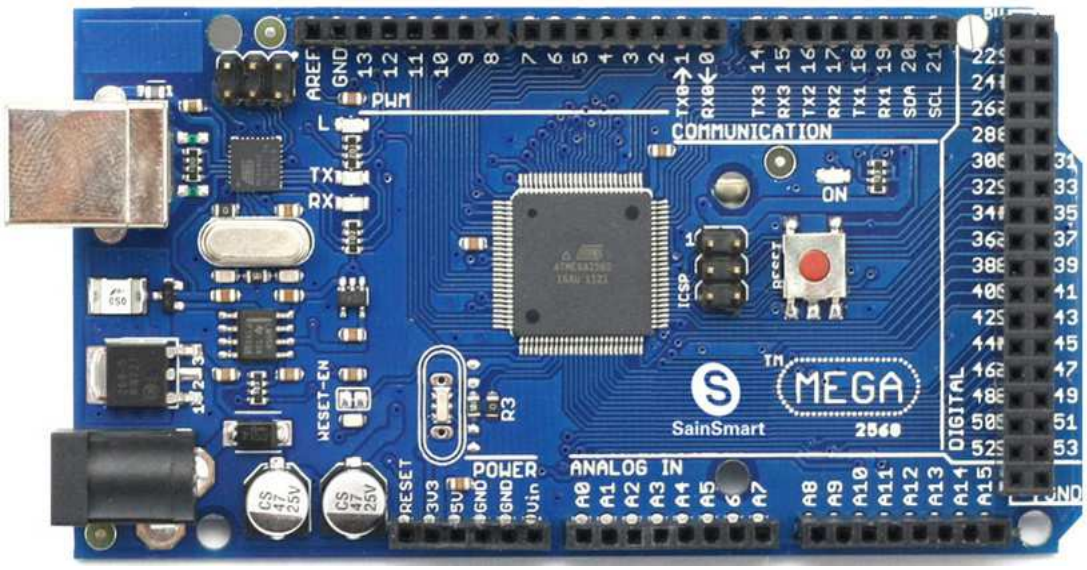


Figure 3.11. Carte Arduino Méga.

b Moteur DC EMG30

Le EMG30 (codeur, moteur, boîte de vitesses 30:1) est un moteur de 12v entièrement équipée avec des encodeurs et 30 boîte de vitesses 1 de réduction.

Il est idéal pour les petites ou moyennes applications robotiques, fournissant un entraînement efficace des coûts et des commentaires pour l'utilisateur. Il comprend également un condensateur de suppression de bruit standard à travers les enroulements du moteur.

***Figure 3.12*.** moteur à courant continu EMG30.

- **Spécification**
  - ✓ Tension nominale 12v.
  - ✓ Démultiplication 30:1.
  - ✓ Couple nominal 1,5 kg / cm.
  - ✓ vitesse nominale 170 tr/min.
  - ✓ courant nominale 530mA (150mA à vide).
  - ✓ Vitesse à vide 216 tr/min.
  - ✓ Courant de décrochage 2.5A.
  - ✓ Puissance nominale 4.22W.
  - ✓ nombre d'encodeurs par tour 360 impulsion.

c Capteur MPU-6050

Capteur MPU-6050 pour mesurer l'accélération et la vitesse angulaire sur un seul axe (l'axe x).

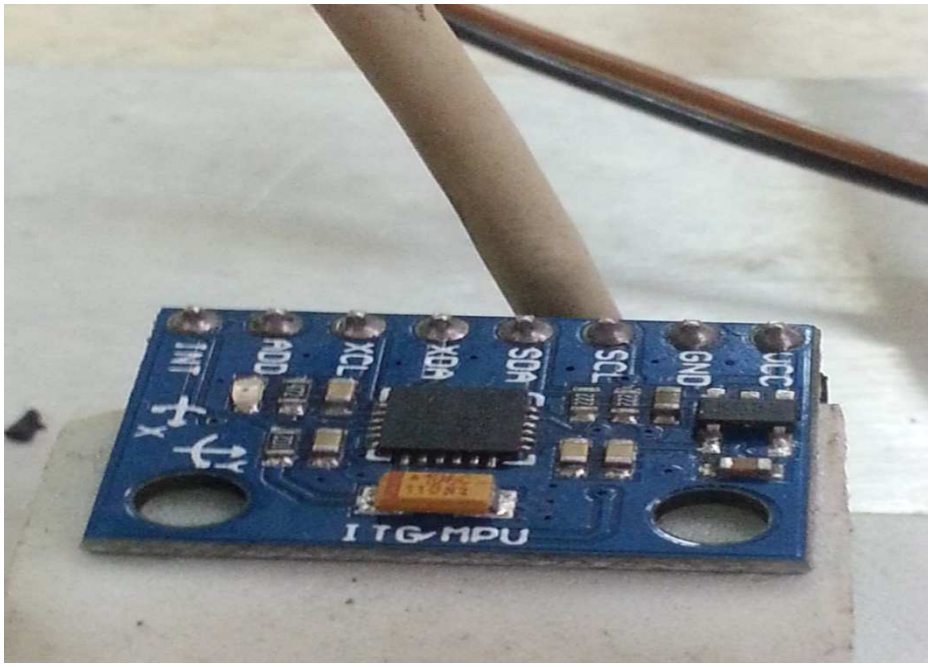


***Figure 3.13*.** Capteur MPU-6050.

d La plate-forme de robot

Deux roues entraînées par le moteur à courant continu EMG30 tourner vers l'avant et vers l'arrière en fonction du signal de commande.

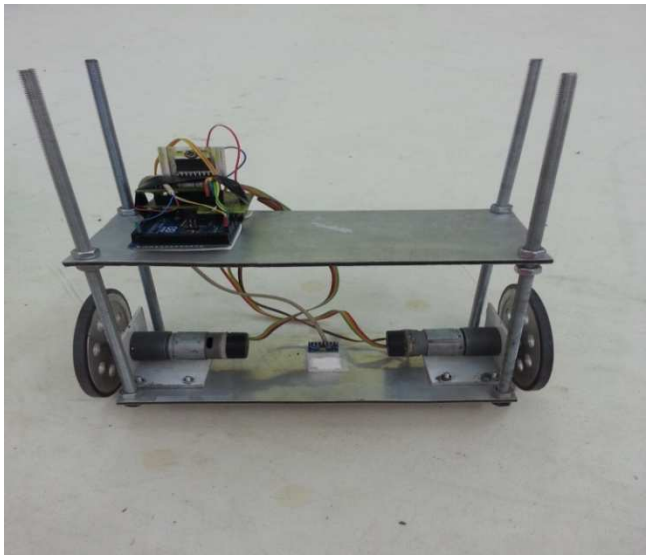


Figure 3.14. Plate-forme de robot.

e Batterie

Une batterie de 12 volts de 4,5 ampères pour alimenter les deux moteurs à courant continu à travers la carte de puissance et de contrôle des conseils (Arduino).

f carte de puissance

Nous avons besoin d'une carte de puissance entre la carte Arduino et les actionneurs (les moteurs à courant continu EMG 30), le schéma suivant représente la carte de puissance utilisée dans le projet qui est fait de L298 double pont en H.

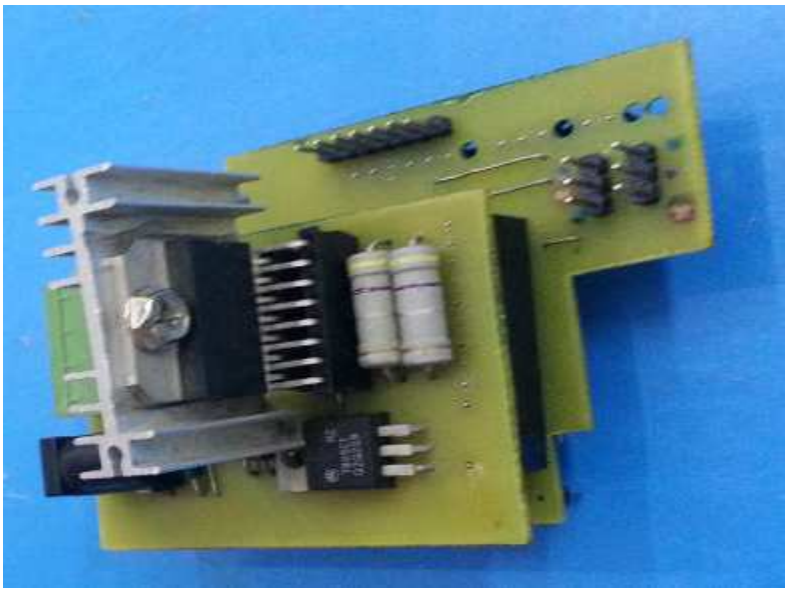


Figure 3.15. Carte de puissance.

3.6 Conclusion

Dans ce chapitre, nous avons bien détaillé notre projet, qui consiste à une étude de conception des différentes cartes électroniques, et aussi d'une partie de programmation sur logiciel ARDUINO, et la partie hardware. Le chapitre suivant sera donc consacré à la réalisation de ces cartes électronique.

4.1 Introduction

La partie la plus importante de notre projet sera discutée dans ce chapitre. Cela inclura les résultats expérimentaux et les problèmes auxquels nous nous sommes confrontés pour faire fonctionner le système global d'une manière satisfaisante.

4.2 Les tests et résultats

4.2.1 L'organigramme général du projet

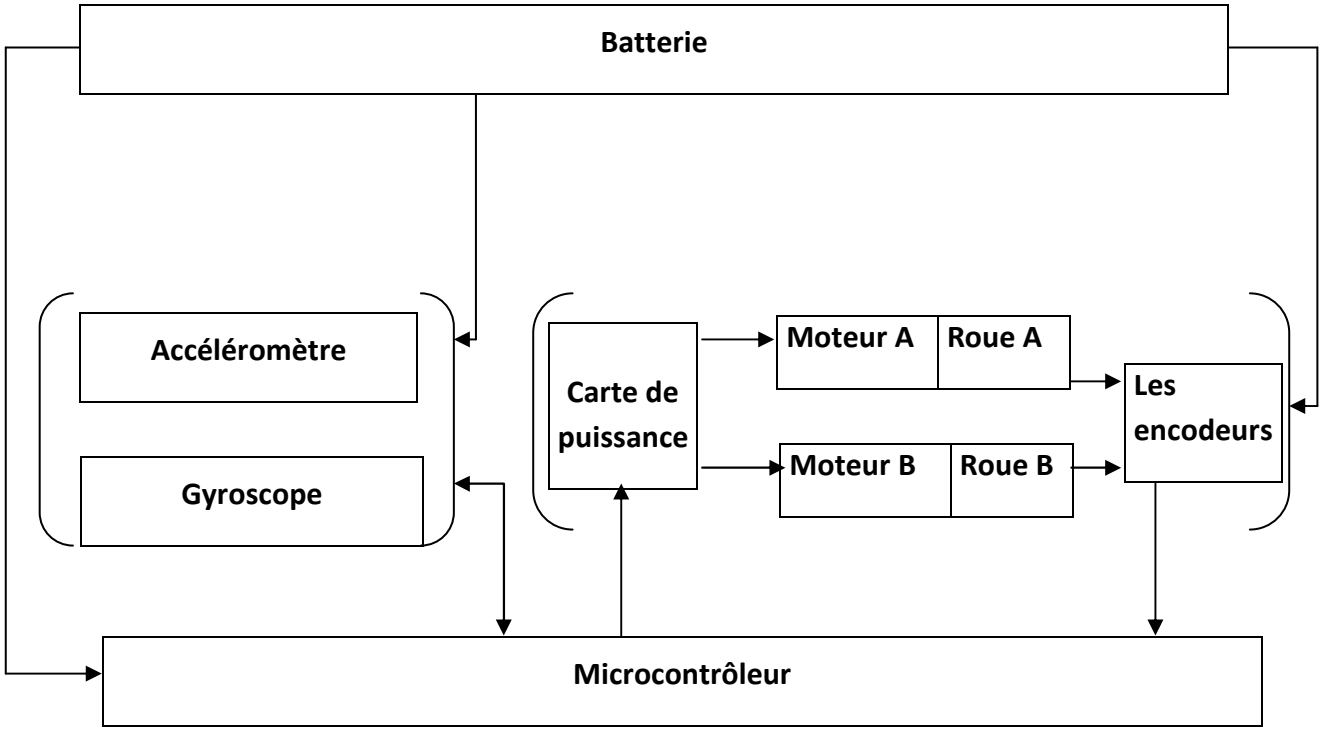


Figure 4.1. Organigramme de matérielle du robot.

4.2.2 L'identification du moteur à courant continu

Le logiciel Matlab contient la « toolbox **ident »** pour faire l'identification des systèmes. Dans notre projet on a utilisé cette toolbox pour trouver le bon modèle du moteur et du pendule inversé.

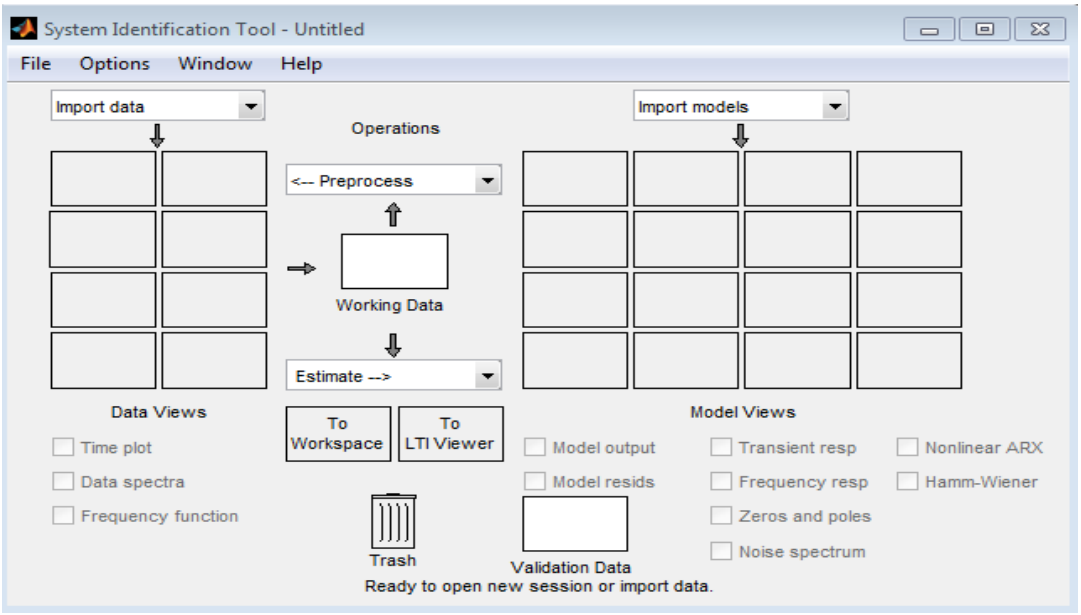


Figure 4.2. Interface d'identification sur Matlab.

Cette figure représente l'entrée (signale PWM) et la sortie (la vitesse du moteur à partir de l'encodeur).

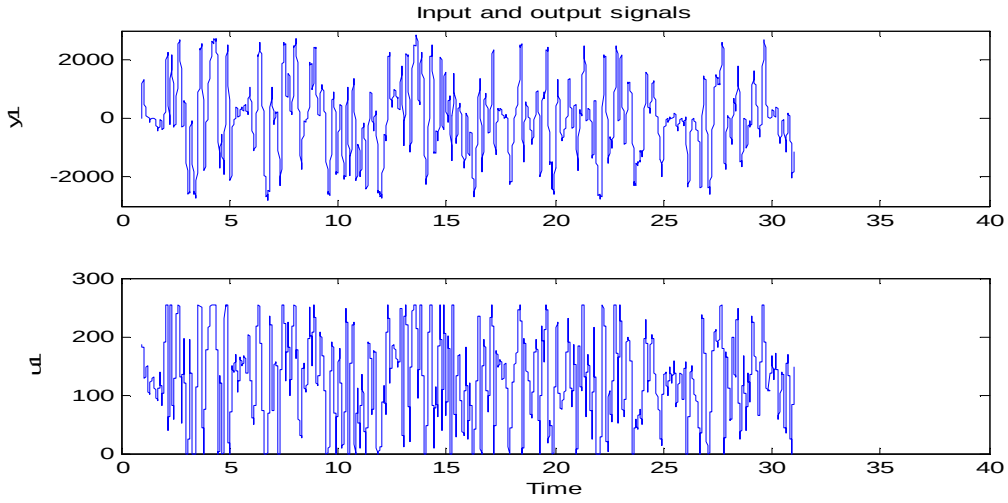


Figure 4.3. Signale de donnée de moteur (u1 l'entrée, y1 la sortie).

Etant donné le nombre important de données recueillies lors de l'étape d'acquisition nous partagé ce nombre en deux parties ; Le premier paquet de ces données sera utilisé pour l'estimation des paramètres et le second pour l'étape de validation.

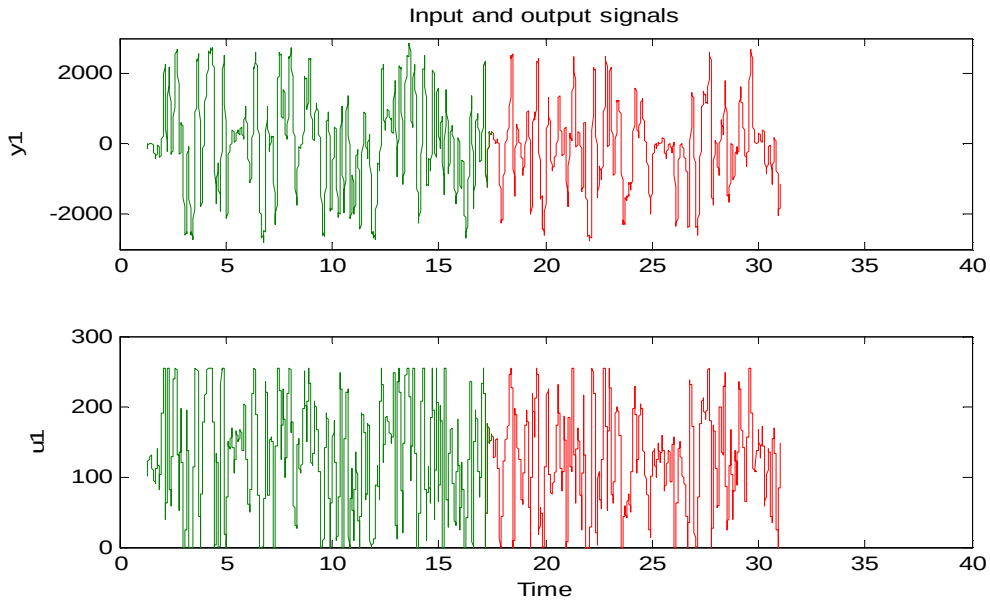


Figure 4.4. Signaux des données échantillonnées (période d'échantillonnage Ts=0.01 sec).

Après plusieurs essais nous avons trouvé un modèle non-linéaire de type Hammerstein Wiener (nlhw7).

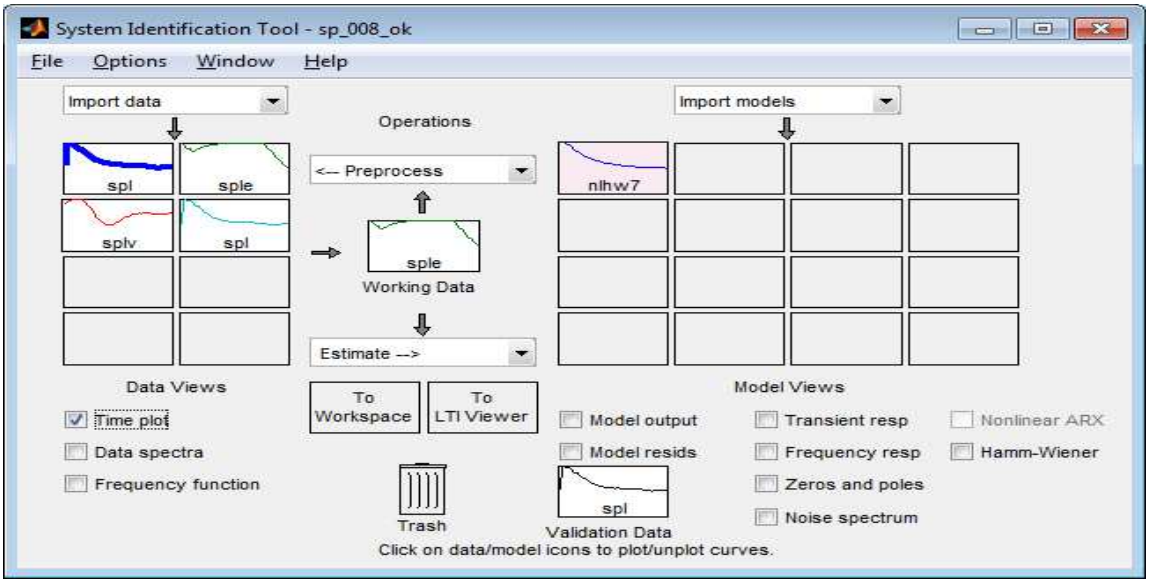


Figure 4.5. L'outil d'identification pour le modèle de moteur.

La figure 4.6 montre que la sortie du modèle suit la sortie mesurée avec un pourcentage de 94.31%, ceci confirme la bonne représentativité de ce modèle pour notre moteur à courant continu.

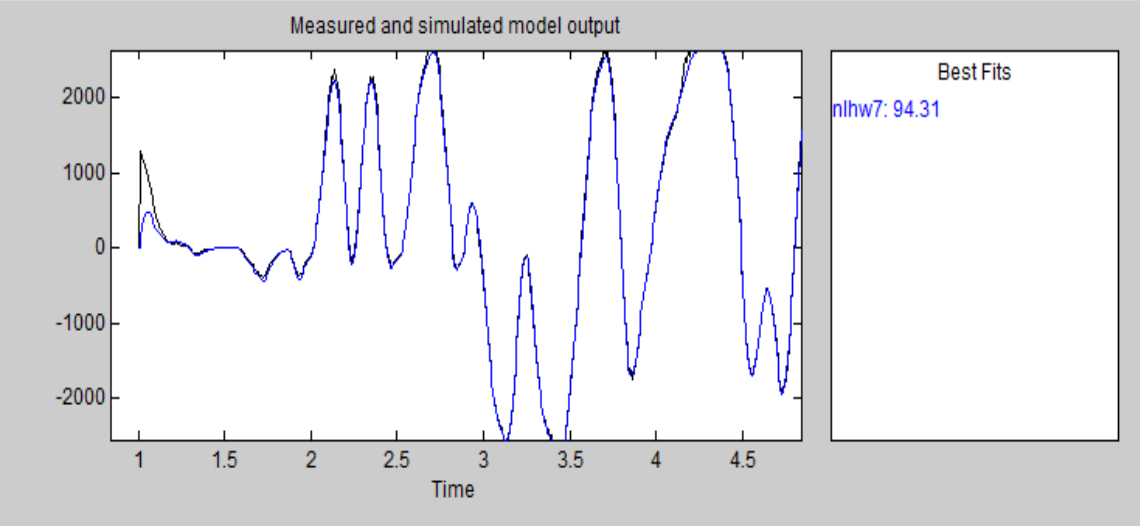


Figure 4.6. La sortie mesurée et la sortie du modèle.

Modèle de moteur avec le contrôleur PID

Après l'identification nous avons utilisé un contrôleur de type PID pour commander la vitesse du moteur.

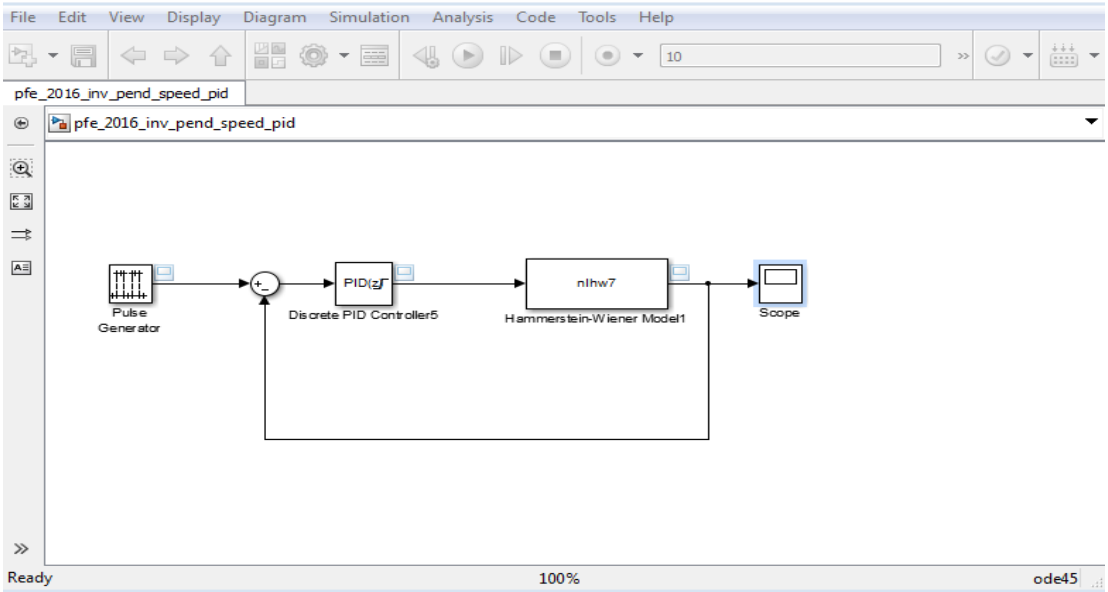


Figure 4.7. Modèle de moteur avec la régulation.

Les paramètres du PID pour réguler la vitesse des moteurs sont :

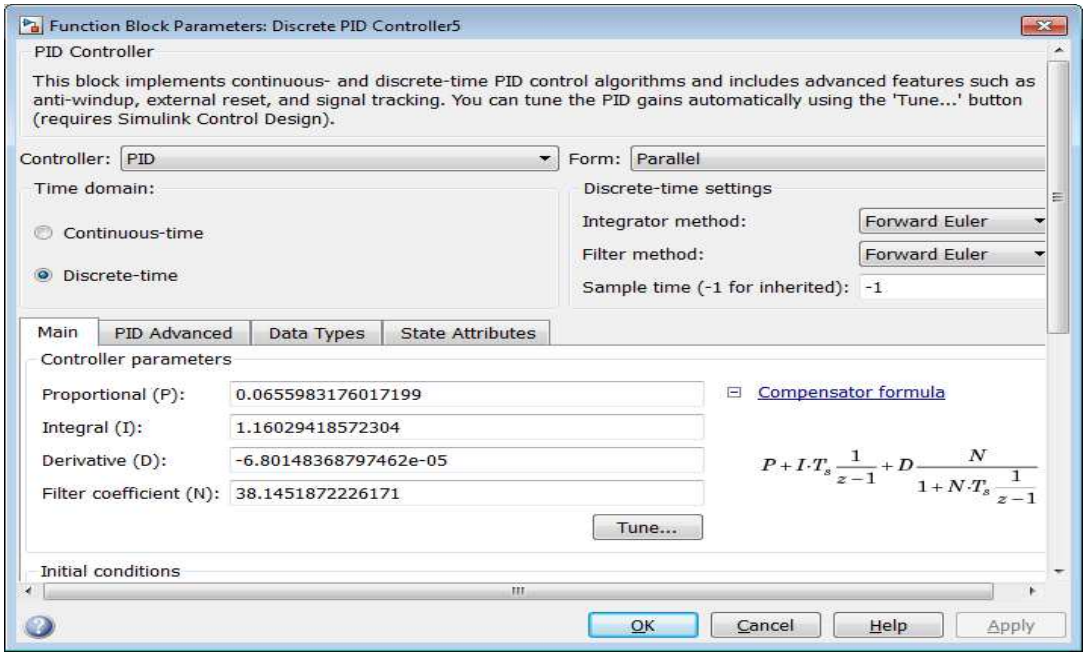


Figure 4.8. Les paramètres PID du moteur.

La figure ci dessous représente la vitesse régulé du moteur, on observe que la vitesse obtenue suit d'une manière très claire celle choisie comme consigne.

Figure 4.9. La vitesse de moteur (en rouge) et la consigne (en bleu).

4.2.3 L'identification du pendule inversé

Cette figure représente l'entrée (vitesse sinusoïdale) et la sortie du pendule inversé (le signal du gyroscope (vitesse angulaire) combiné avec l'accéléromètre).

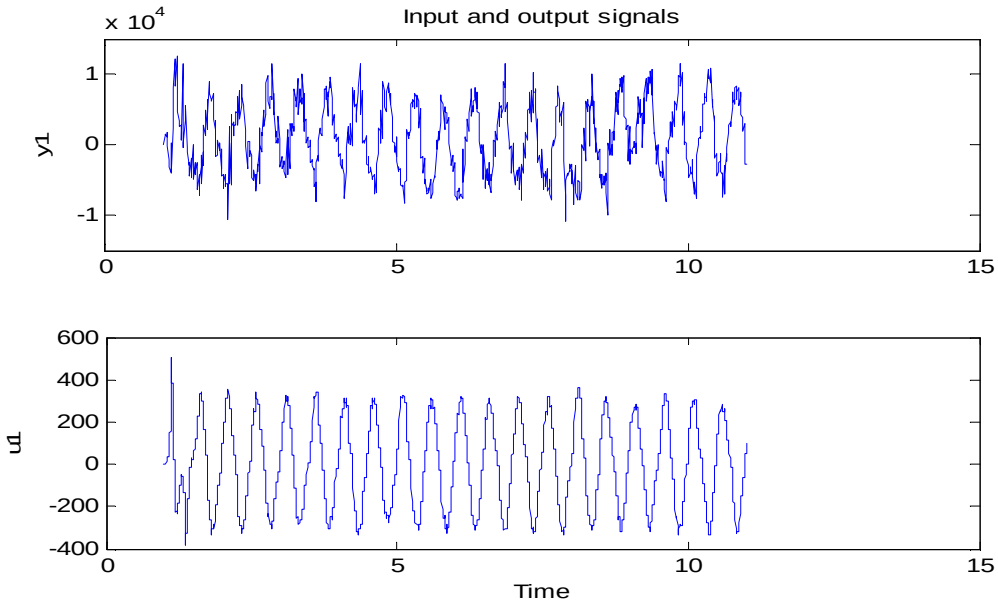


Figure 4.10. Signale de donnée du pendule.

Pour l'identification on a essayé plusieurs modèles pour trouver le bon model. La figure 4.11 montre qu'a l'issu de ces test nous avons obtenu le modèle nlarx19 qui répond à notre besoin.

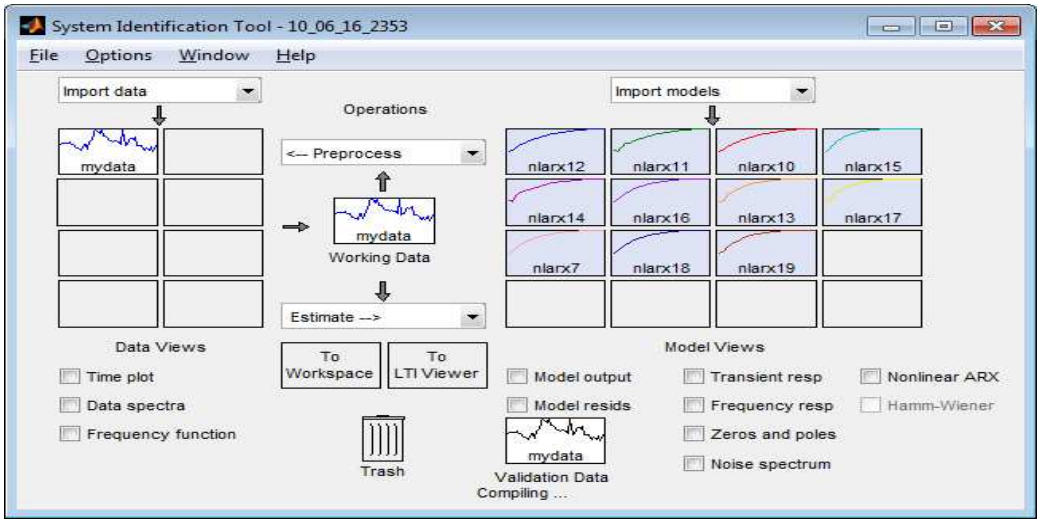


Figure 4.11. L'outil d'identification pour le modèle du pendule inversé.

On observés que la sortie du modèle obtenu est très proche de la sortie mesurée avec un pourcentage de 51.92%, alors c'est le bon modèle pour notre pendule inversé.

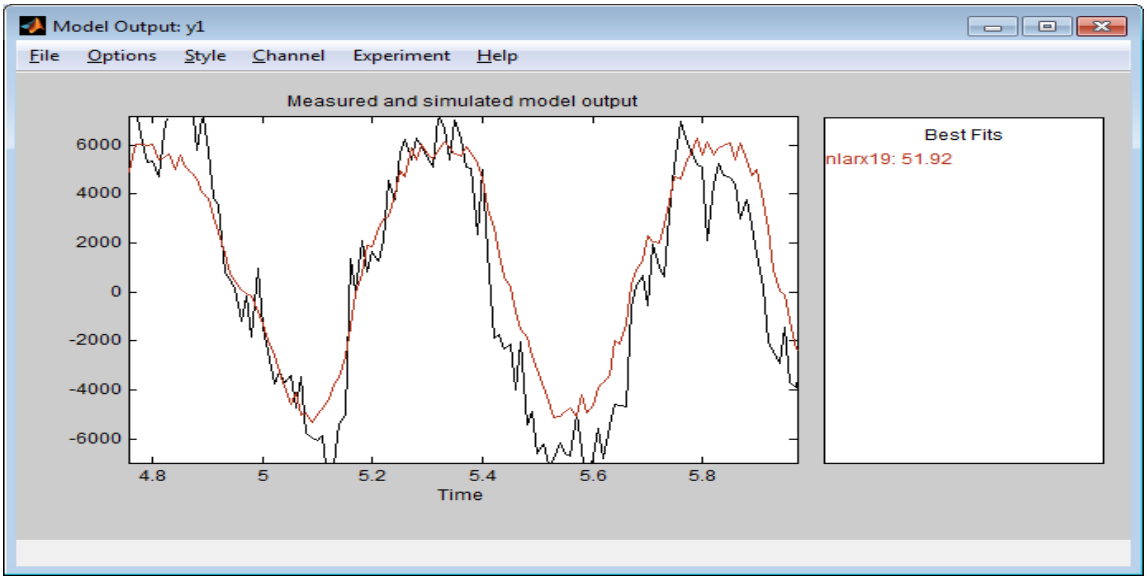


Figure 4.12. La sortie mesurée et la sortie du modèle.

Le modèle du pendule avec le régulateur PID

De même que pour le control de la vitesse des deux moteurs, après l'identification nous avons appliqué un régulateur PID pour stabiliser le pendule inversé sur la position vertical, ceci en donnant une consigne nulle comme signal de commande pour la vitesse des deux moteurs de notre pendule.

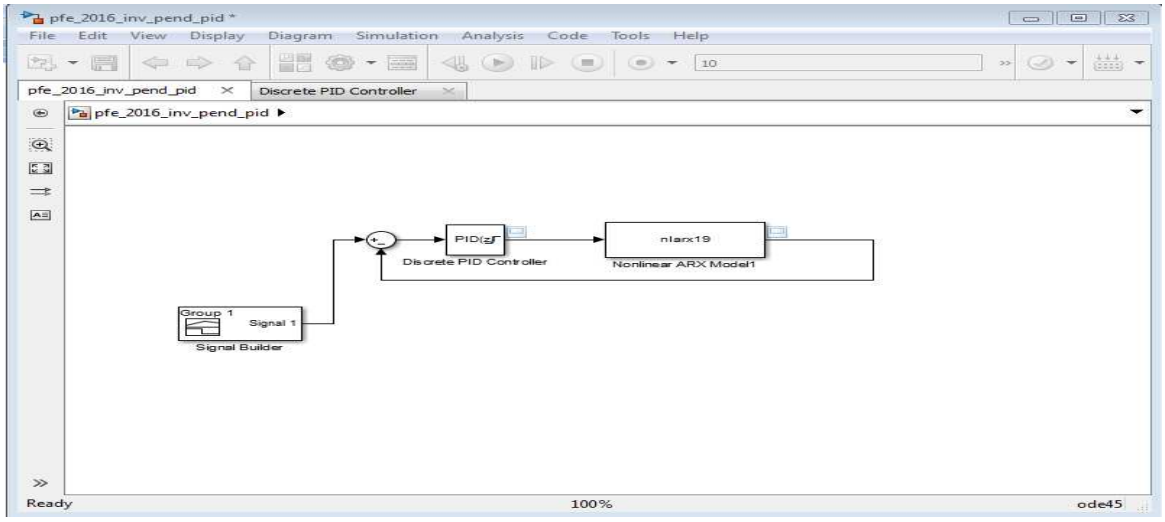


Figure 4.13. Modèle du pendule inversé avec la régulation.

Les paramètres de régulateur PID sont :

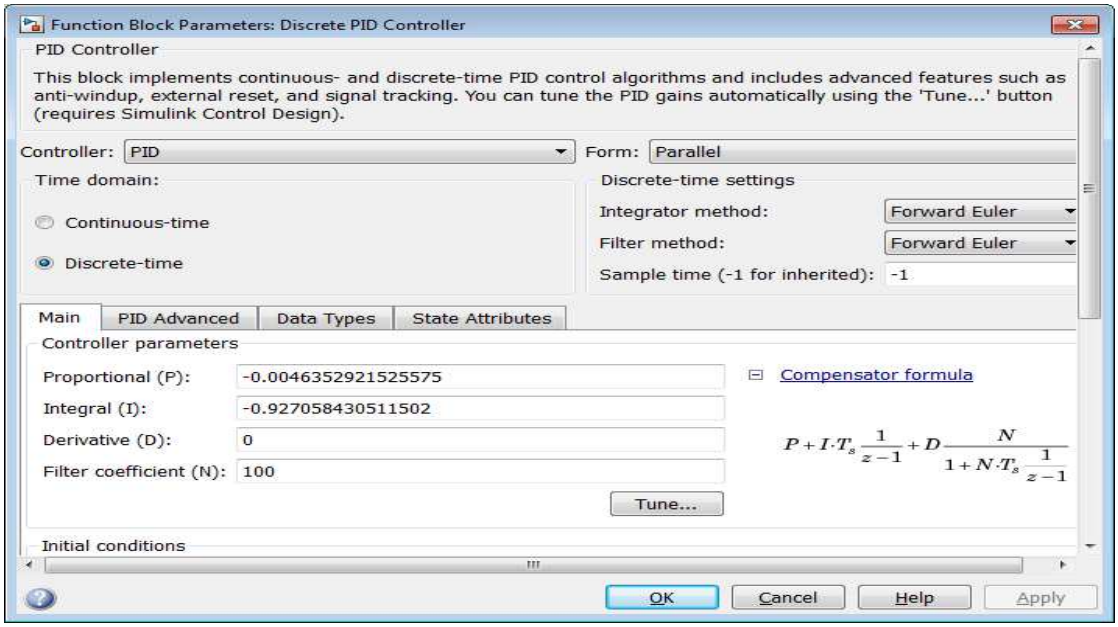


Figure 4.14.Les paramètres PID du pendule inversé.

Cette figure représente la vitesse régulée du pendule inversé.

Figure 4.15. La vitesse du pendule inversé avec l'entrée.

On remarque que le pendule inversé retourne à l'équilibre rapidement lorsqu'on applique une impulsion, Le pendule fonctionne correctement et répond à la consigne désirée.

La figure ci-dessous représente les trois cas de notre robot.

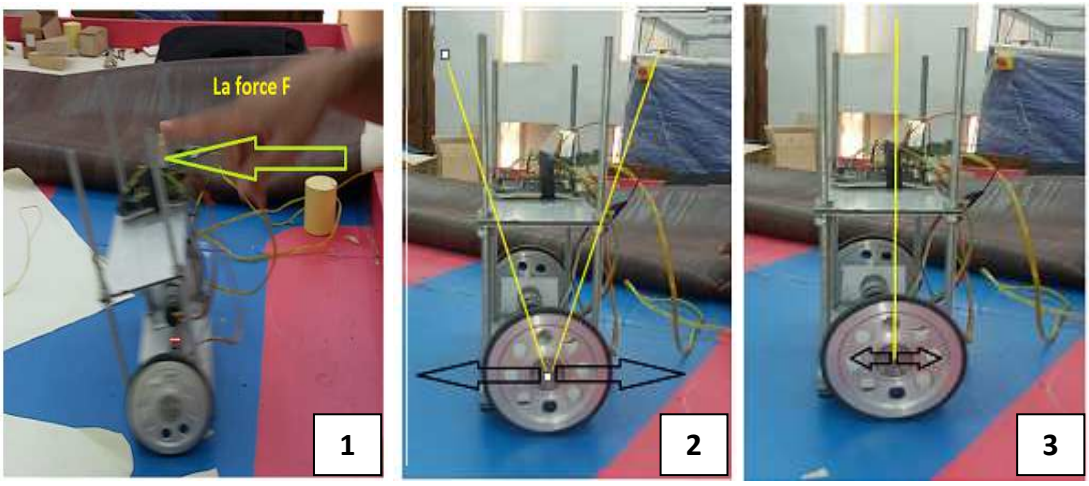
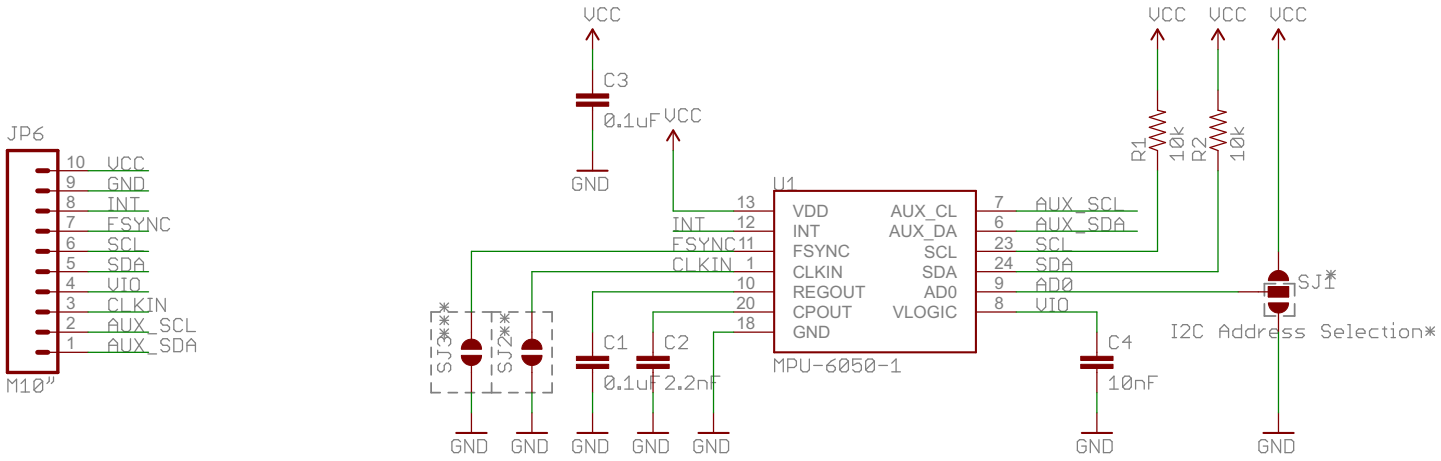


Figure 4.16. Représentation les comportements de notre robot après une perturbation.

4.3. Conclusion

Dans ce chapitre nous avons présenté les différents résultats obtenus par l'identification et simulation pour chaque modèle. Les résultats de simulations présentés au cours de ce chapitre ont été réalisés à l'aide de l'outil **Simulink.**



* Two MPU-60X0s can be connected to the same I2C bus
The LSB bit of the 7 bit address is determined by the logic level on pin AD0.
Default Address = 0x68 (pin AD0 is logic low)
Alternative Address =0x69 (pin AD0 is logic high)

** Optional external reference clock input. Connected to GND by default.
Cut trace for external clock

*** Frame synchronization digital input. Connected to GND by default
Cut trace for external sync

Design by: Joel Bartlett

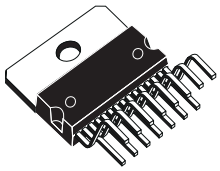
open hardware

TITLE: MPU-6050_Breakout V11	
Document Number:	REV:
Date: not saved!	Sheet: 1/1

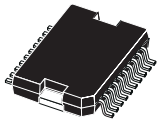
L298

DUAL FULL-BRIDGE DRIVER

- OPERATING SUPPLY VOLTAGE UP TO 46 V
- TOTAL DC CURRENT UP TO 4 A
- LOW SATURATION VOLTAGE
- OVERTEMPERATURE PROTECTION
- LOGICAL "0" INPUT VOLTAGE UP TO 1.5 V (HIGH NOISE IMMUNITY)



Multiwatt15



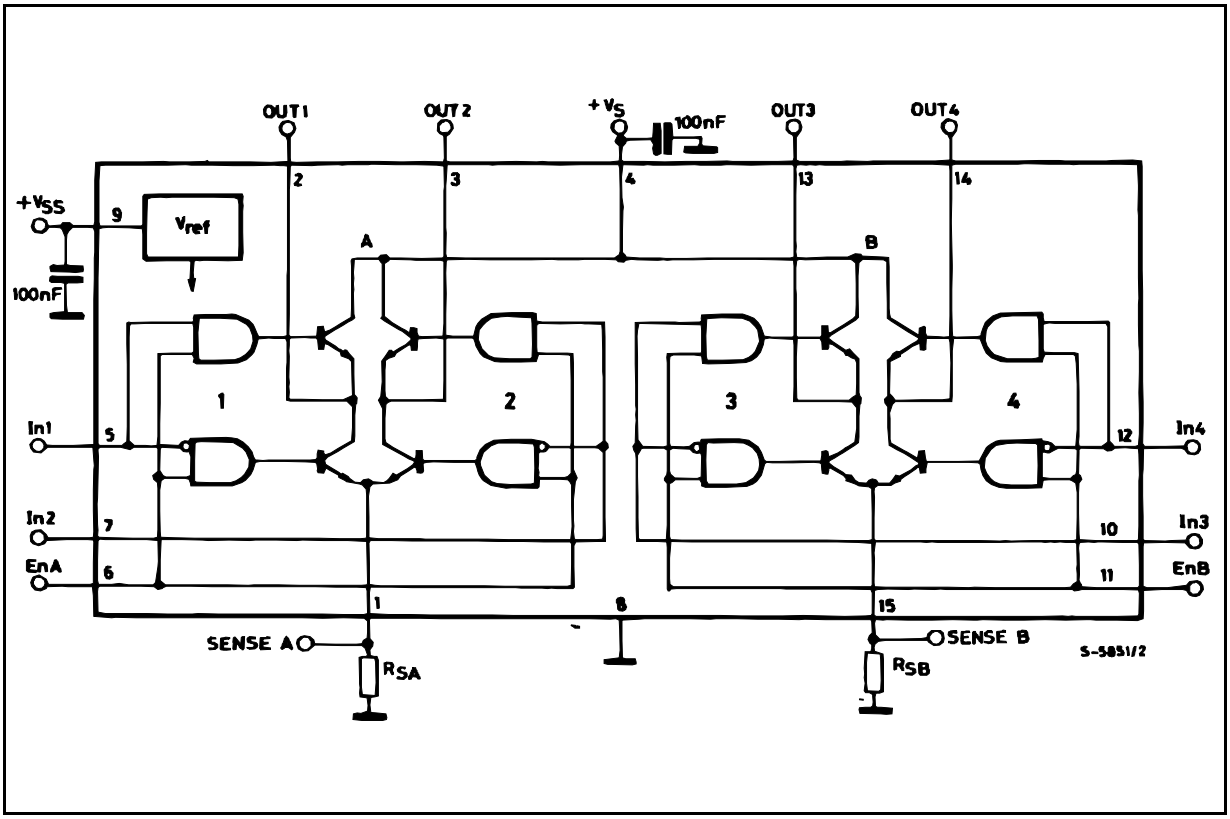
PowerSO20

ORDERING NUMBERS : L298N (Multiwatt Vert.)
L298HN (Multiwatt Horiz.)
L298P (PowerSO20)

DESCRIPTION

The L298 is an integrated monolithic circuit in a 15-lead Multiwatt and PowerSO20 packages. It is a high voltage, high current dual full-bridge driver designed to accept standard TTL logic levels and drive inductive loads such as relays, solenoids, DC and stepping motors. Two enable inputs are provided to enable or disable the device independently of the input signals. The emitters of the lower transistors of each bridge are connected together and the corresponding external terminal can be used for the connection of an external sensing resistor. An additional supply input is provided so that the logic works at a lower voltage.

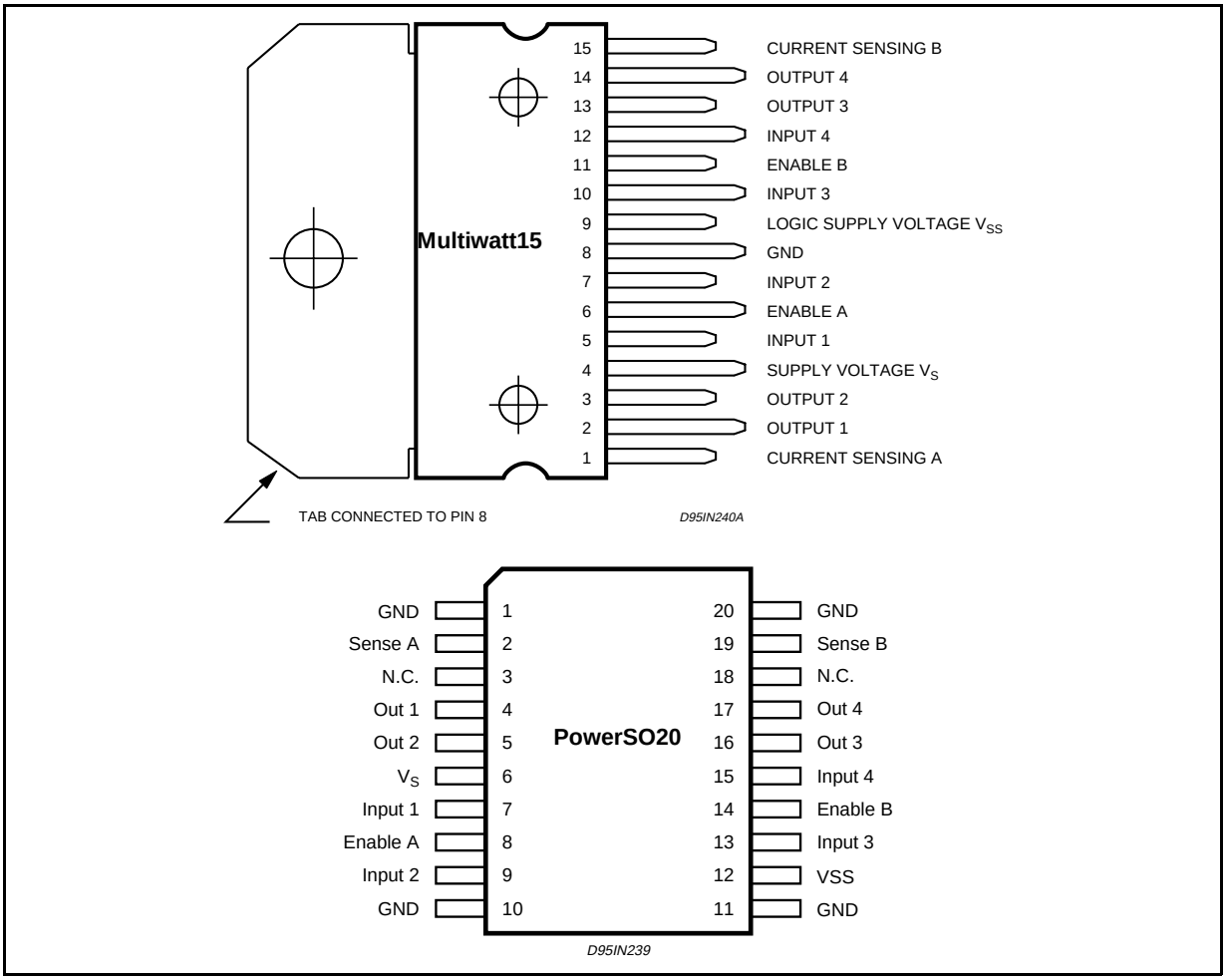
BLOCK DIAGRAM



ABSOLUTE MAXIMUM RATINGS

Symbol	Parameter	Value	Unit
V_S	Power Supply	50	V
V_{SS}	Logic Supply Voltage	7	V
V_I,V_{en}	Input and Enable Voltage	–0.3 to 7	V
I_O	Peak Output Current (each Channel) – Non Repetitive (t = 100µs) –Repetitive (80% on –20% off; t_{on} = 10ms) –DC Operation	 3 2.5 2	 A A A
V_{sens}	Sensing Voltage	–1 to 2.3	V
P_{tot}	Total Power Dissipation (T_{case} = 75°C)	25	W
T_{op}	Junction Operating Temperature	–25 to 130	°C
T_{stg}, T_j	Storage and Junction Temperature	–40 to 150	°C

PIN CONNECTIONS (top view)



THERMAL DATA

Symbol	Parameter		PowerSO20	Multiwatt15	Unit
$R_{th\ j\text{-}case}$	Thermal Resistance Junction-case	Max.	–	3	°C/W
$R_{th\ j\text{-}amb}$	Thermal Resistance Junction-ambient	Max.	13 (*)	35	°C/W

(*) Mounted on aluminum substrate

PIN FUNCTIONS (refer to the block diagram)

MW.15	PowerSO	Name	Function
1;15	2;19	Sense A; Sense B	Between this pin and ground is connected the sense resistor to control the current of the load.
2;3	4;5	Out 1; Out 2	Outputs of the Bridge A; the current that flows through the load connected between these two pins is monitored at pin 1.
4	6	V_S	Supply Voltage for the Power Output Stages. A non-inductive 100nF capacitor must be connected between this pin and ground.
5;7	7;9	Input 1; Input 2	TTL Compatible Inputs of the Bridge A.
6;11	8;14	Enable A; Enable B	TTL Compatible Enable Input: the L state disables the bridge A (enable A) and/or the bridge B (enable B).
8	1,10,11,20	GND	Ground.
9	12	VSS	Supply Voltage for the Logic Blocks. A100nF capacitor must be connected between this pin and ground.
10; 12	13;15	Input 3; Input 4	TTL Compatible Inputs of the Bridge B.
13; 14	16;17	Out 3; Out 4	Outputs of the Bridge B. The current that flows through the load connected between these two pins is monitored at pin 15.
–	3;18	N.C.	Not Connected

ELECTRICAL CHARACTERISTICS (V_S = 42V; V_{SS} = 5V, T_j = 25°C; unless otherwise specified)

Symbol	Parameter	Test Conditions	Min.	Typ.	Max.	Unit
V_S	Supply Voltage (pin 4)	Operative Condition	V_{IH} +2.5		46	V
V_{SS}	Logic Supply Voltage (pin 9)		4.5	5	7	V
I_S	Quiescent Supply Current (pin 4)	V_{en} = H; I_L = 0 V_i = L V_i = H		13 50	22 70	mA mA
		V_{en} = L V_i = X			4	mA
I_{SS}	Quiescent Current from V_{SS} (pin 9)	V_{en} = H; I_L = 0 V_i = L V_i = H		24 7	36 12	mA mA
		V_{en} = L V_i = X			6	mA
V_{iL}	Input Low Voltage (pins 5, 7, 10, 12)		–0.3		1.5	V
V_{iH}	Input High Voltage (pins 5, 7, 10, 12)		2.3		VSS	V
I_{iL}	Low Voltage Input Current (pins 5, 7, 10, 12)	V_i = L			–10	µA
I_{iH}	High Voltage Input Current (pins 5, 7, 10, 12)	Vi = H ≤ V_{SS} –0.6V		30	100	µA
V_{en} = L	Enable Low Voltage (pins 6, 11)		–0.3		1.5	V
V_{en} = H	Enable High Voltage (pins 6, 11)		2.3		V_{SS}	V
I_{en} = L	Low Voltage Enable Current (pins 6, 11)	V_{en} = L			–10	µA
I_{en} = H	High Voltage Enable Current (pins 6, 11)	V_{en} = H ≤ V_{SS} –0.6V		30	100	µA
$V_{CEsat\ (H)}$	Source Saturation Voltage	I_L = 1A I_L = 2A	0.95	1.35 2	1.7 2.7	V V
$V_{CEsat\ (L)}$	Sink Saturation Voltage	I_L = 1A (5) I_L = 2A (5)	0.85	1.2 1.7	1.6 2.3	V V
V_{CEsat}	Total Drop	I_L = 1A (5) I_L = 2A (5)	1.80		3.2 4.9	V V
V_{sens}	Sensing Voltage (pins 1, 15)		–1 (1)		2	V

ELECTRICAL CHARACTERISTICS (continued)

Symbol	Parameter	Test Conditions	Min.	Typ.	Max.	Unit
T_1 (V_i)	Source Current Turn-off Delay	0.5 V_i to 0.9 I_L (2); (4)		1.5		µs
T_2 (V_i)	Source Current Fall Time	0.9 I_L to 0.1 I_L (2); (4)		0.2		µs
T_3 (V_i)	Source Current Turn-on Delay	0.5 V_i to 0.1 I_L (2); (4)		2		µs
T_4 (V_i)	Source Current Rise Time	0.1 I_L to 0.9 I_L (2); (4)		0.7		µs
T_5 (V_i)	Sink Current Turn-off Delay	0.5 V_i to 0.9 I_L (3); (4)		0.7		µs
T_6 (V_i)	Sink Current Fall Time	0.9 I_L to 0.1 I_L (3); (4)		0.25		µs
T_7 (V_i)	Sink Current Turn-on Delay	0.5 V_i to 0.9 I_L (3); (4)		1.6		µs
T_8 (V_i)	Sink Current Rise Time	0.1 I_L to 0.9 I_L (3); (4)		0.2		µs
fc (V_i)	Commutation Frequency	I_L = 2A		25	40	KHz
T_1 (V_{en})	Source Current Turn-off Delay	0.5 V_{en} to 0.9 I_L (2); (4)		3		µs
T_2 (V_{en})	Source Current Fall Time	0.9 I_L to 0.1 I_L (2); (4)		1		µs
T_3 (V_{en})	Source Current Turn-on Delay	0.5 V_{en} to 0.1 I_L (2); (4)		0.3		µs
T_4 (V_{en})	Source Current Rise Time	0.1 I_L to 0.9 I_L (2); (4)		0.4		µs
T_5 (V_{en})	Sink Current Turn-off Delay	0.5 V_{en} to 0.9 I_L (3); (4)		2.2		µs
T_6 (V_{en})	Sink Current Fall Time	0.9 I_L to 0.1 I_L (3); (4)		0.35		µs
T_7 (V_{en})	Sink Current Turn-on Delay	0.5 V_{en} to 0.9 I_L (3); (4)		0.25		µs
T_8 (V_{en})	Sink Current Rise Time	0.1 I_L to 0.9 I_L (3); (4)		0.1		µs

1) 1)Sensing voltage can be –1 V for t ≤ 50 µsec; in steady state V_{sens} min ≥ – 0.5 V.
2) See fig. 2.
3) See fig. 4.
4) The load must be a pure resistor.

Figure 1 : Typical Saturation Voltage vs. Output Current.

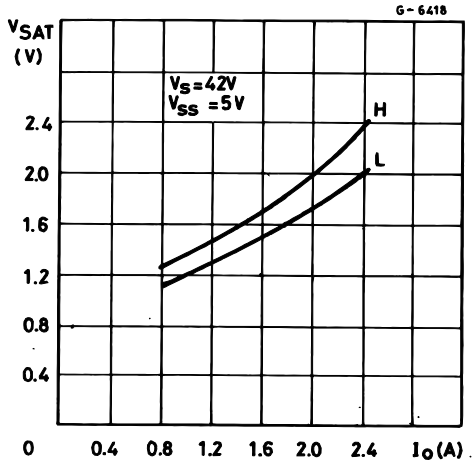
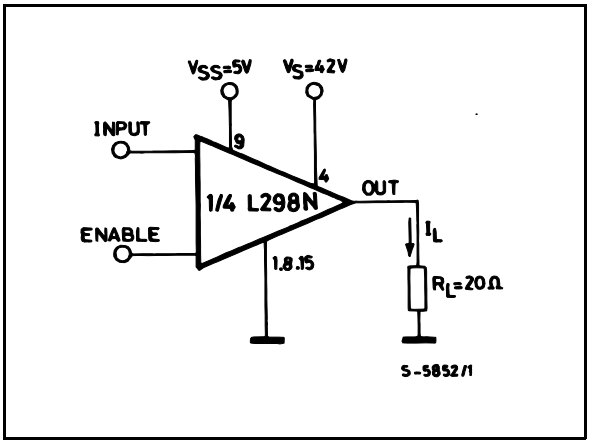


Figure 2 : Switching Times Test Circuits.



Note : For INPUT Switching, set EN = H
For ENABLE Switching, set IN = H

Figure 3 : Source Current Delay Times vs. Input or Enable Switching.

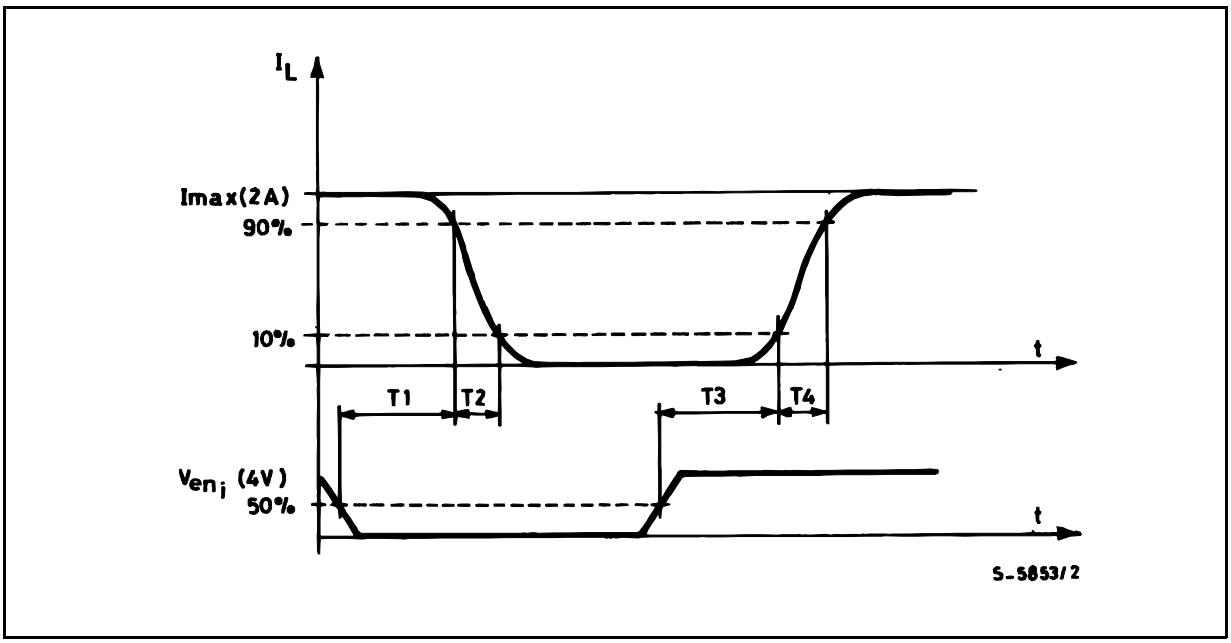
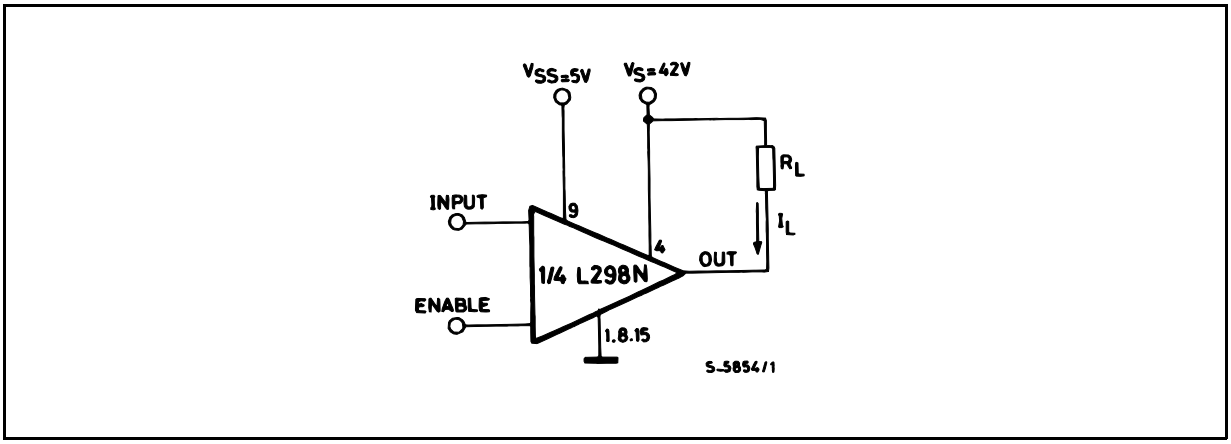


Figure 4 : Switching Times Test Circuits.



Note : For INPUT Switching, set EN = H
For ENABLE Switching, set IN = L

Figure 5 : Sink Current Delay Times vs. Input 0 V Enable Switching.

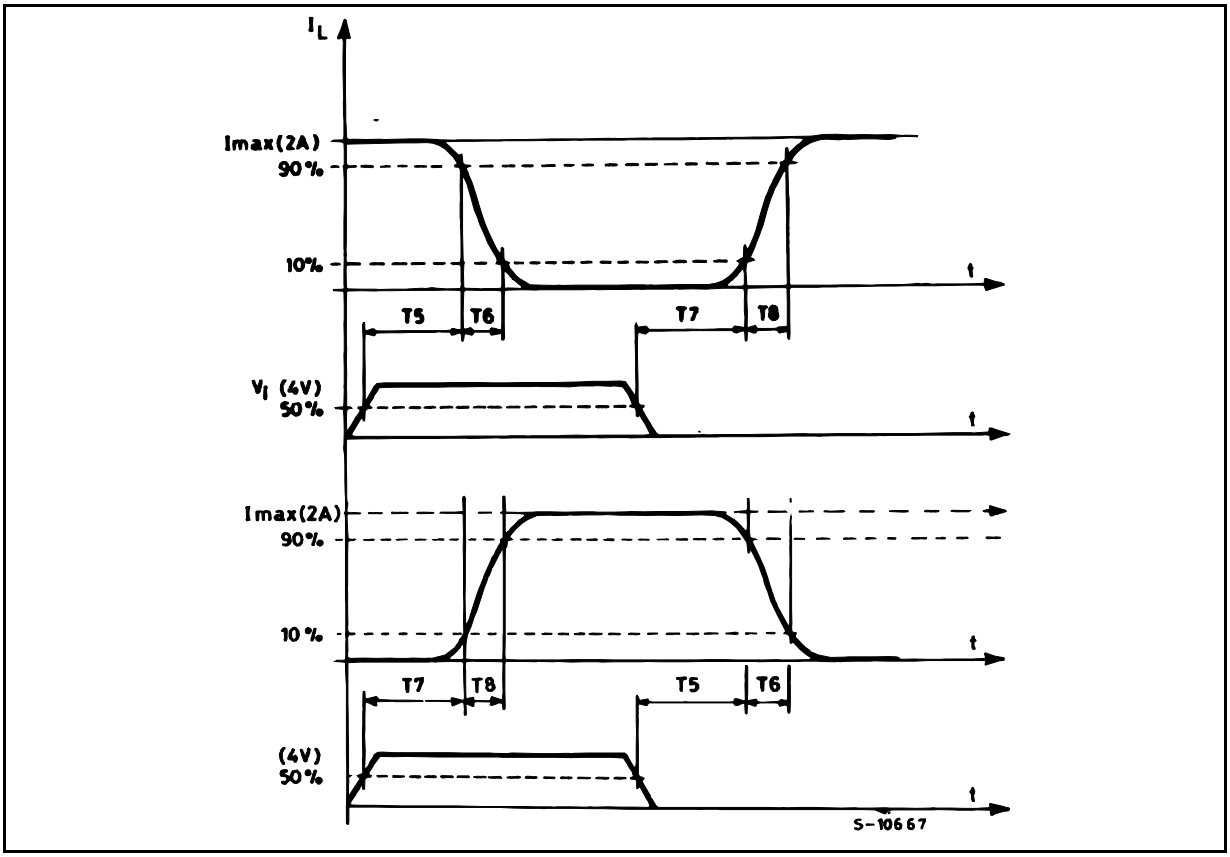
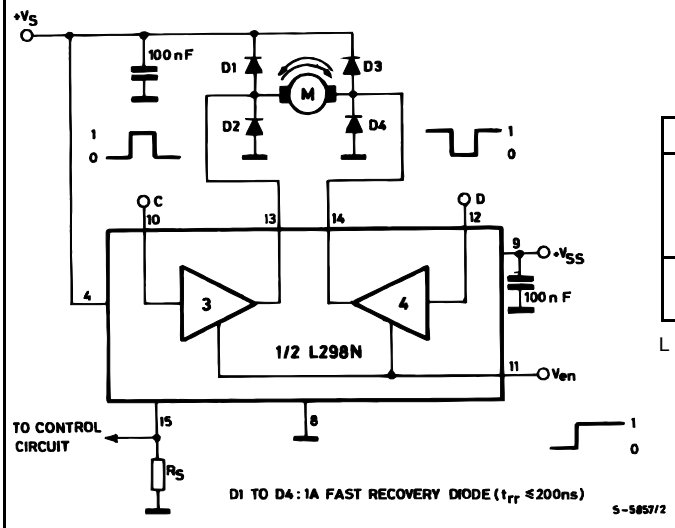


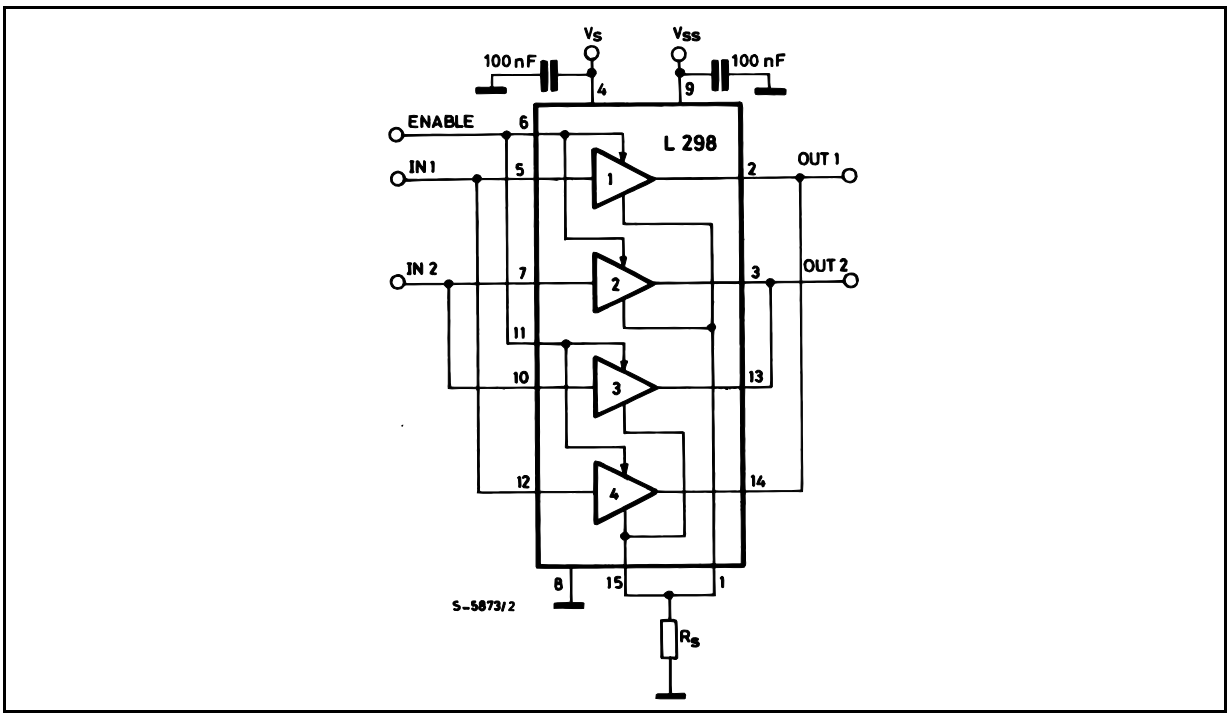
Figure 6 : Bidirectional DC Motor Control.



Inputs		Function
V_{en} = H	C = H ; D = L	Forward
	C = L ; D = H	Reverse
	C = D	Fast Motor Stop
V_{en} = L	C = X ; D = X	Free Running Motor Stop

L = Low H = High X = Don't care

Figure 7 : For higher currents, outputs can be paralleled. Take care to parallel channel 1 with channel 4 and channel 2 with channel 3.



APPLICATION INFORMATION (Refer to the block diagram)

1.1. POWER OUTPUT STAGE

The L298 integrates two power output stages (A ; B). The power output stage is a bridge configuration and its outputs can drive an inductive load in common or differenzial mode, depending on the state of the inputs. The current that flows through the load comes out from the bridge at the sense output : an external resistor (R_{SA} ; R_{SB}.) allows to detect the intensity of this current.

1.2. INPUT STAGE

Each bridge is driven by means of four gates the input of which are In1 ; In2 ; EnA and In3 ; In4 ; EnB. The In inputs set the bridge state when The En input is high ; a low state of the En input inhibits the bridge. All the inputs are TTL compatible.

2. SUGGESTIONS

A non inductive capacitor, usually of 100 nF, must be foreseen between both Vs and Vss, to ground, as near as possible to GND pin. When the large capacitor of the power supply is too far from the IC, a second smaller one must be foreseen near the L298.

The sense resistor, not of a wire wound type, must be grounded near the negative pole of Vs that must be near the GND pin of the I.C.

Each input must be connected to the source of the driving signals by means of a very short path.

Turn-On and Turn-Off : Before to Turn-ON the Supply Voltage and before to Turn it OFF, the Enable input must be driven to the Low state.

3. APPLICATIONS

Fig 6 shows a bidirectional DC motor control Schematic Diagram for which only one bridge is needed. The external bridge of diodes D1 to D4 is made by four fast recovery elements (trr $\leq$ 200 nsec) that must be chosen of a VF as low as possible at the worst case of the load current.

The sense output voltage can be used to control the current amplitude by chopping the inputs, or to provide overcurrent protection by switching low the enable input.

The brake function (Fast motor stop) requires that the Absolute Maximum Rating of 2 Amps must never be overcome.

When the repetitive peak current needed from the load is higher than 2 Amps, a paralleled configuration can be chosen (See Fig.7).

An external bridge of diodes are required when inductive loads are driven and when the inputs of the IC are chopped ; Shottky diodes would be preferred.

This solution can drive until 3 Amps In DC operation and until 3.5 Amps of a repetitive peak current.

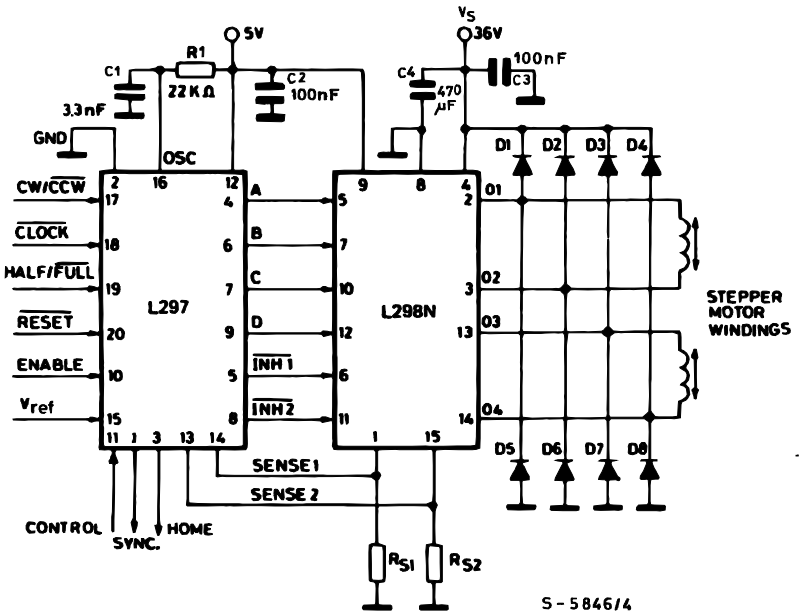
On Fig 8 it is shown the driving of a two phase bipolar stepper motor ; the needed signals to drive the inputs of the L298 are generated, in this example, from the IC L297.

Fig 9 shows an example of P.C.B. designed for the application of Fig 8.

Fig 10 shows a second two phase bipolar stepper motor control circuit where the current is controlled by the I.C. L6506.

Figure 8 : Two Phase Bipolar Stepper Motor Circuit.

This circuit drives bipolar stepper motors with winding currents up to 2 A. The diodes are fast 2 A types.



$R_{S1} = R_{S2} = 0.5\ \Omega$

D1 to D8 = 2 A Fast diodes { $V_F \leq 1.2$ V @ I = 2 A
trr ≤ 200 ns

Figure 9 : Suggested Printed Circuit Board Layout for the Circuit of fig. 8 (1:1 scale).

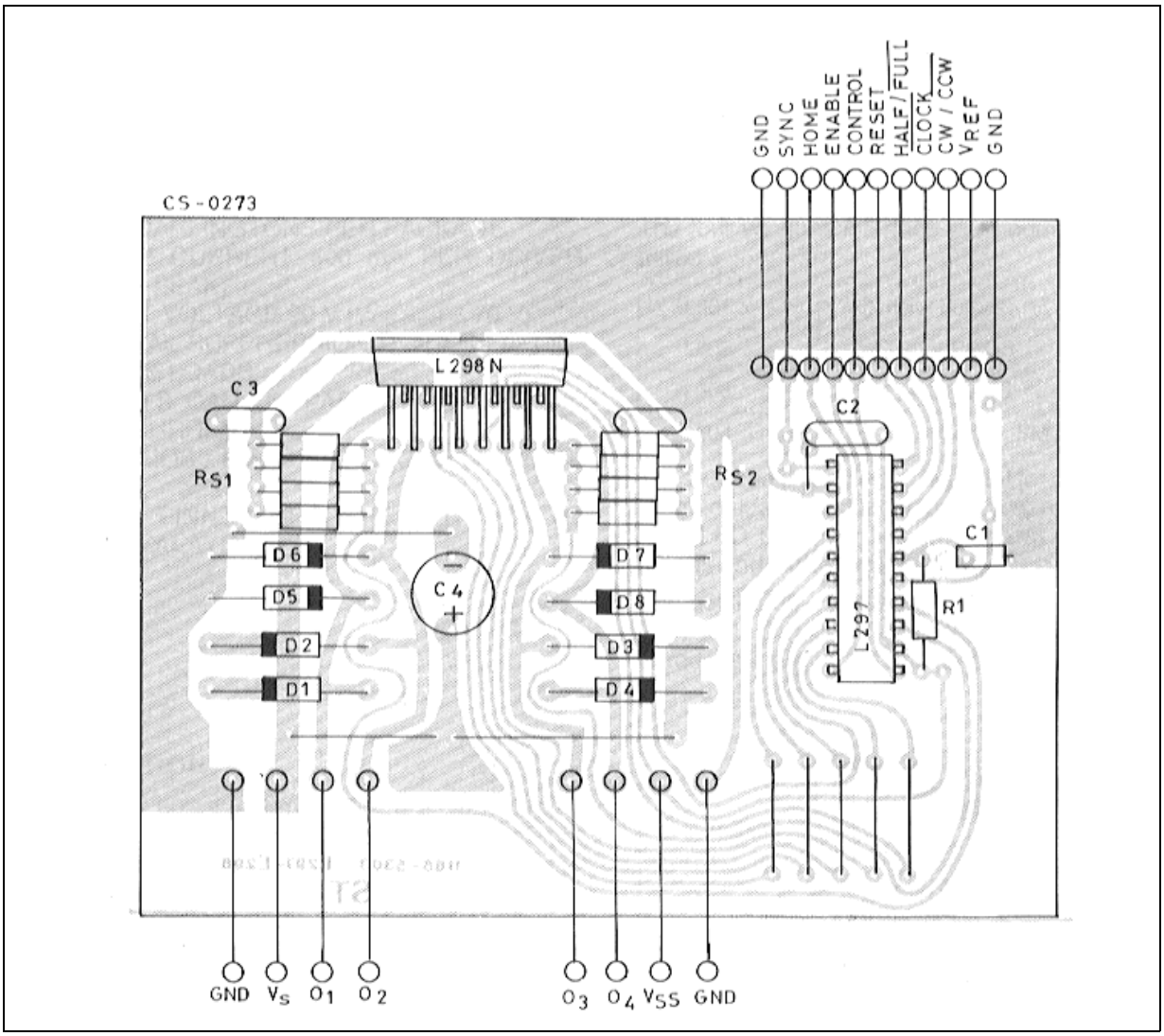
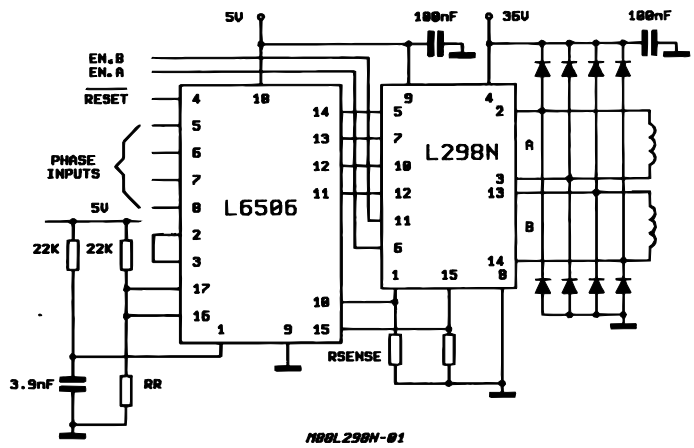


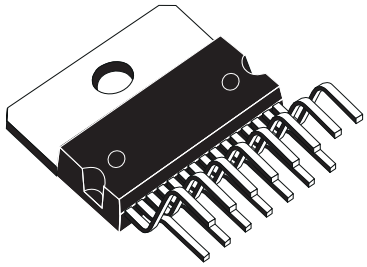
Figure 10 : Two Phase Bipolar Stepper Motor Control Circuit by Using the Current Controller L6506.



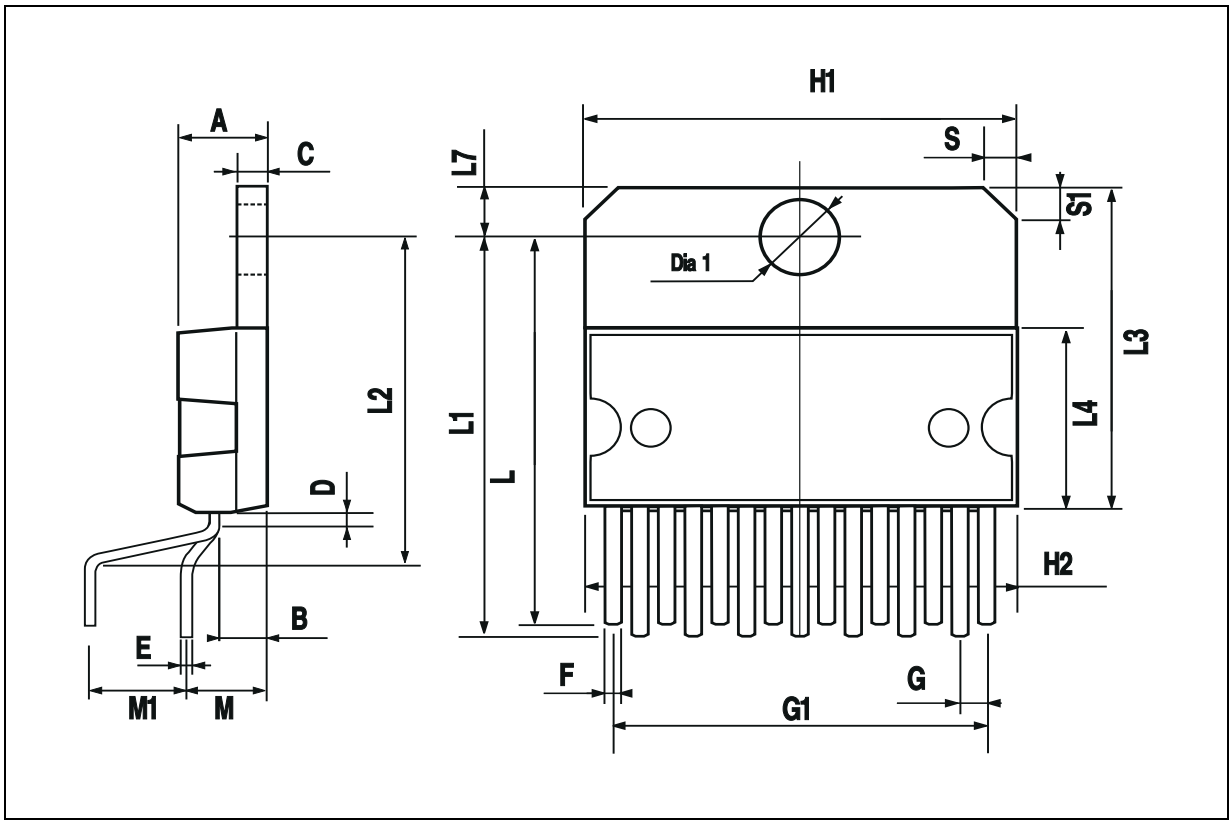
R_R and R_{sense} depend from the load current

DIM.	mm			inch		
	MIN.	TYP.	MAX.	MIN.	TYP.	MAX.
A			5			0.197
B			2.65			0.104
C			1.6			0.063
D		1			0.039	
E	0.49		0.55	0.019		0.022
F	0.66		0.75	0.026		0.030
G	1.02	1.27	1.52	0.040	0.050	0.060
G1	17.53	17.78	18.03	0.690	0.700	0.710
H1	19.6			0.772		
H2			20.2			0.795
L	21.9	22.2	22.5	0.862	0.874	0.886
L1	21.7	22.1	22.5	0.854	0.870	0.886
L2	17.65		18.1	0.695		0.713
L3	17.25	17.5	17.75	0.679	0.689	0.699
L4	10.3	10.7	10.9	0.406	0.421	0.429
L7	2.65		2.9	0.104		0.114
M	4.25	4.55	4.85	0.167	0.179	0.191
M1	4.63	5.08	5.53	0.182	0.200	0.218
S	1.9		2.6	0.075		0.102
S1	1.9		2.6	0.075		0.102
Dia1	3.65		3.85	0.144		0.152

OUTLINE AND MECHANICAL DATA

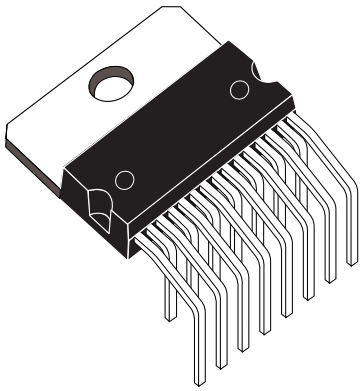


Multiwatt15 V

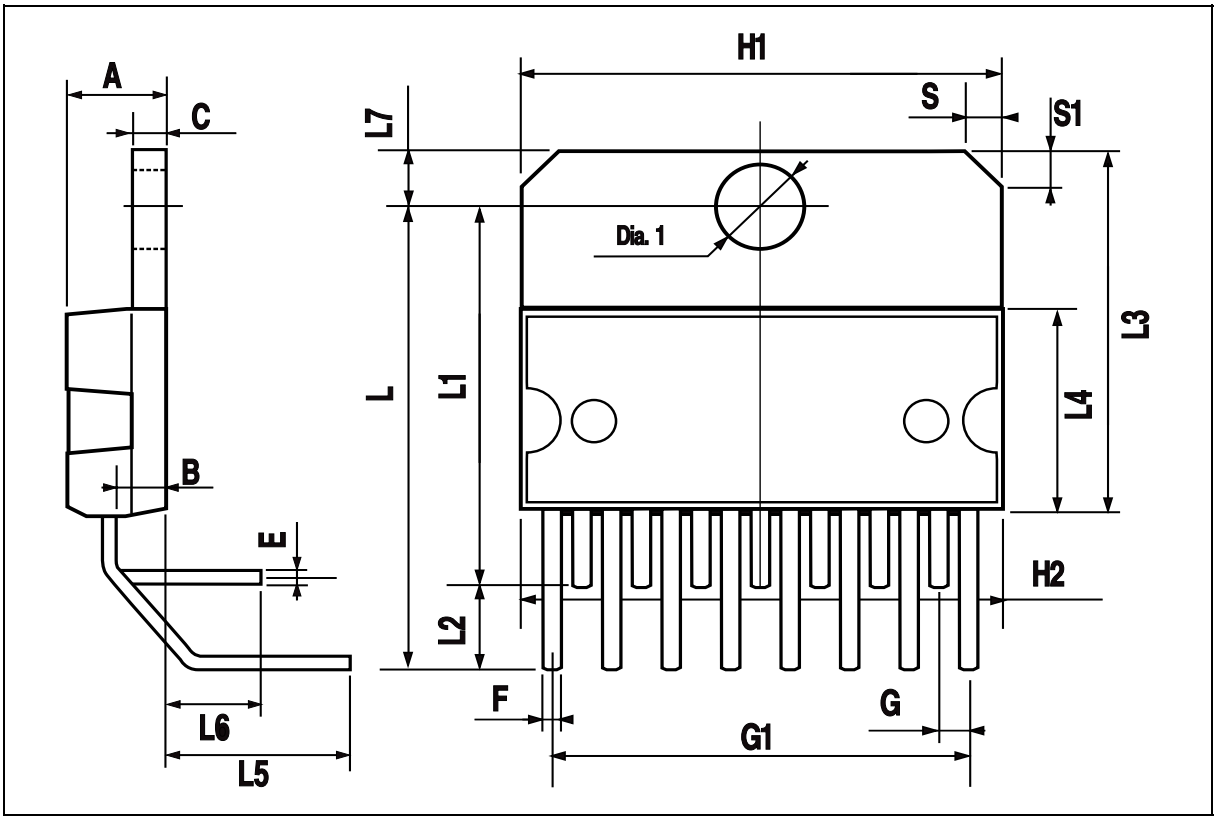


DIM.	mm			inch		
	MIN.	TYP.	MAX.	MIN.	TYP.	MAX.
A			5			0.197
B			2.65			0.104
C			1.6			0.063
E	0.49		0.55	0.019		0.022
F	0.66		0.75	0.026		0.030
G	1.14	1.27	1.4	0.045	0.050	0.055
G1	17.57	17.78	17.91	0.692	0.700	0.705
H1	19.6			0.772		
H2			20.2			0.795
L		20.57			0.810	
L1		18.03			0.710	
L2		2.54			0.100	
L3	17.25	17.5	17.75	0.679	0.689	0.699
L4	10.3	10.7	10.9	0.406	0.421	0.429
L5		5.28			0.208	
L6		2.38			0.094	
L7	2.65		2.9	0.104		0.114
S	1.9		2.6	0.075		0.102
S1	1.9		2.6	0.075		0.102
Dia1	3.65		3.85	0.144		0.152

OUTLINE AND MECHANICAL DATA



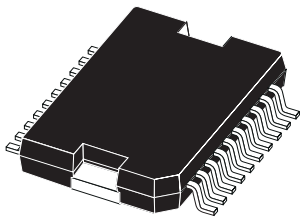
Multiwatt15 H



DIM.	mm			inch		
	MIN.	TYP.	MAX.	MIN.	TYP.	MAX.
A			3.6			0.142
a1	0.1		0.3	0.004		0.012
a2			3.3			0.130
a3	0		0.1	0.000		0.004
b	0.4		0.53	0.016		0.021
c	0.23		0.32	0.009		0.013
D (1)	15.8		16	0.622		0.630
D1	9.4		9.8	0.370		0.386
E	13.9		14.5	0.547		0.570
e		1.27			0.050	
e3		11.43			0.450	
E1 (1)	10.9		11.1	0.429		0.437
E2			2.9			0.114
E3	5.8		6.2	0.228		0.244
G	0		0.1	0.000		0.004
H	15.5		15.9	0.610		0.626
h			1.1			0.043
L	0.8		1.1	0.031		0.043
N	10˚ (max.)					
S	8˚ (max.)					
T		10			0.394	

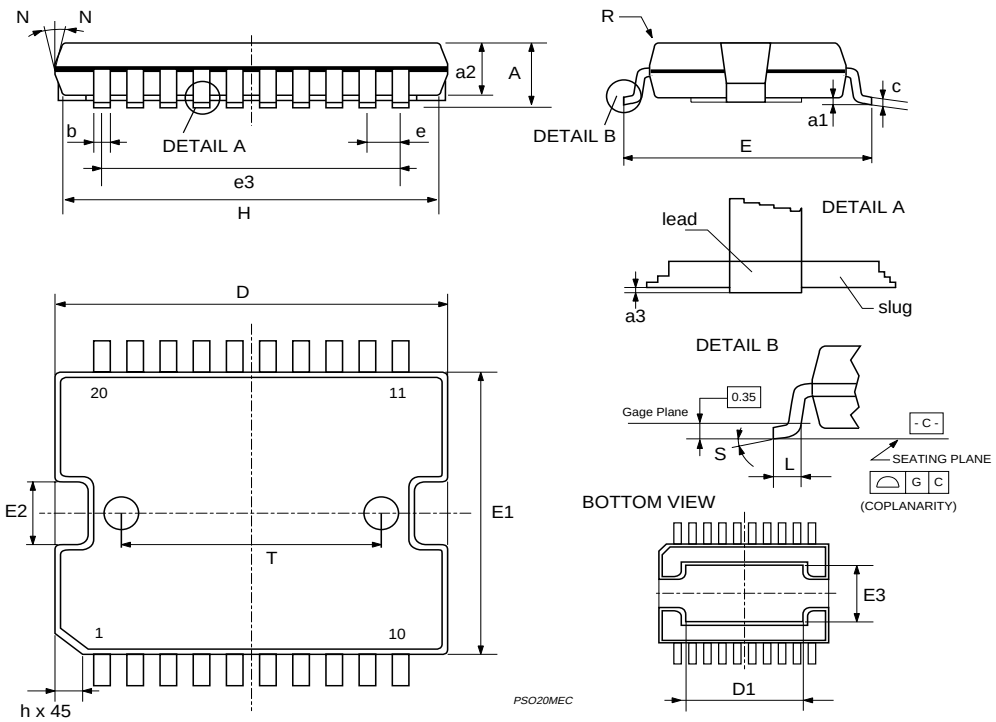
(1) "D and F" do not include mold flash or protrusions.
- Mold flash or protrusions shall not exceed 0.15 mm (0.006").
- Critical dimensions: "E", "G" and "a3"

OUTLINE AND MECHANICAL DATA



JEDEC MO-166

PowerSO20



Le moteur a CC EMG 30:

L'EMG30 est fourni avec un connecteur JST 6 voie.

Couleur du fil Connexion :

Violet (1) : Capteur B Vout.

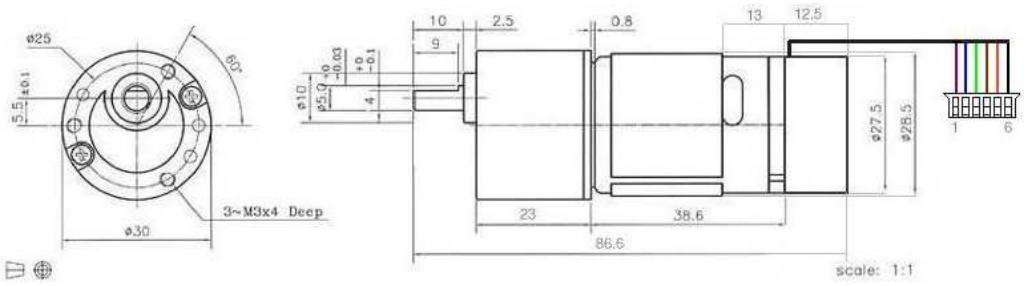
Bleu (2) : Capteur A Vout.

Vert (3): Capteur Hall terre.

Brun (4): Capteur Hall Vcc.

Rouge (5): + Moteur.

Noir (6) : - Moteur.



Leur spicification:

Tension nominale 12v.

Couple nominal 1,5 kg / cm.

Vitesse Nominale 170 rpm.

Courant 530 mA.

Vitesse à vide 216.

Puissance nominale 4.22W.

Nombre des impultions par tour 360.

Les regestres de gyroscope et l'accéléromètre

4.4 Register 27 – Gyroscope Configuration
GYRO_CONFIG

Type: Read/Write

Register (Hex)	Register (Decimal)	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
1B	27	XG_ST	YG_ST	ZG_ST	FS_SEL[1:0]		-	-	-

Description:

This register is used to trigger gyroscope self-test and configure the gyroscopes' full scale range.

Gyroscope self-test permits users to test the mechanical and electrical portions of the gyroscope. The self-test for each gyroscope axis can be activated by controlling the *XG_ST*, *YG_ST*, and *ZG_ST* bits of this register. Self-test for each axis may be performed independently or all at the same time.

When self-test is activated, the on-board electronics will actuate the appropriate sensor. This actuation will move the sensor's proof masses over a distance equivalent to a pre-defined Coriolis force. This proof mass displacement results in a change in the sensor output, which is reflected in the output signal. The output signal is used to observe the self-test response.

The self-test response is defined as follows:

Self-test response = Sensor output with self-test enabled – Sensor output without self-test enabled

The self-test limits for each gyroscope axis is provided in the electrical characteristics tables of the MPU-6000/MPU-6050 Product Specification document. When the value of the self-test response is within the min/max limits of the product specification, the part has passed self test. When the self-test response exceeds the min/max values specified in the document, the part is deemed to have failed self-test.

FS_SEL selects the full scale range of the gyroscope outputs according to the following table.

FS_SEL	Full Scale Range
0	± 250 °/s
1	± 500 °/s
2	± 1000 °/s
3	± 2000 °/s

Bits 2 through 0 are reserved.

Parameters:

XG_ST	Setting this bit causes the X axis gyroscope to perform self test.
YG_ST	Setting this bit causes the Y axis gyroscope to perform self test.
ZG_ST	Setting this bit causes the Z axis gyroscope to perform self test.
FS_SEL	2-bit unsigned value. Selects the full scale range of gyroscopes.

4.5 Register 28 – Accelerometer Configuration
ACCEL_CONFIG

Type: Read/Write

Register (Hex)	Register (Decimal)	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
1C	28	XA_ST	YA_ST	ZA_ST	AFS_SEL[1:0]		-		

Description:

This register is used to trigger accelerometer self test and configure the accelerometer full scale range. This register also configures the Digital High Pass Filter (DHPF).

Accelerometer self-test permits users to test the mechanical and electrical portions of the accelerometer. The self-test for each accelerometer axis can be activated by controlling the *XA_ST*, *YA_ST*, and *ZA_ST* bits of this register. Self-test for each axis may be performed independently or all at the same time.

When self-test is activated, the on-board electronics will actuate the appropriate sensor. This actuation simulates an external force. The actuated sensor, in turn, will produce a corresponding output signal. The output signal is used to observe the self-test response.

The self-test response is defined as follows:

Self-test response = Sensor output with self-test enabled – Sensor output without self-test enabled

The self-test limits for each accelerometer axis is provided in the electrical characteristics tables of the MPU-6000/MPU-6050 Product Specification document. When the value of the self-test response is within the min/max limits of the product specification, the part has passed self test. When the self-test response exceeds the min/max values specified in the document, the part is deemed to have failed self-test.

AFS_SEL selects the full scale range of the accelerometer outputs according to the following table.

AFS_SEL	Full Scale Range
0	± 2*g*
1	± 4*g*
2	± 8*g*
3	± 16*g*

Parameters:

XA_ST	When set to 1, the X- Axis accelerometer performs self test.
YA_ST	When set to 1, the Y- Axis accelerometer performs self test.
ZA_ST	When set to 1, the Z- Axis accelerometer performs self test.
AFS_SEL	2-bit unsigned value. Selects the full scale range of accelerometers.

4.17 Registers 59 to 64 – Accelerometer Measurements ACCEL_XOUT_H, ACCEL_XOUT_L, ACCEL_YOUT_H, ACCEL_YOUT_L, ACCEL_ZOUT_H, and ACCEL_ZOUT_L

Type: Read Only

Register (Hex)	Register (Decimal)	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
3B	59	ACCEL_XOUT[15:8]							
3C	60	ACCEL_XOUT[7:0]							
3D	61	ACCEL_YOUT[15:8]							
3E	62	ACCEL_YOUT[7:0]							
3F	63	ACCEL_ZOUT[15:8]							
40	64	ACCEL_ZOUT[7:0]							

Description:

These registers store the most recent accelerometer measurements.

Accelerometer measurements are written to these registers at the Sample Rate as defined in Register 25.

The accelerometer measurement registers, along with the temperature measurement registers, gyroscope measurement registers, and external sensor data registers, are composed of two sets of registers: an internal register set and a user-facing read register set.

The data within the accelerometer sensors' internal register set is always updated at the Sample Rate. Meanwhile, the user-facing read register set duplicates the internal register set's data values whenever the serial interface is idle. This guarantees that a burst read of sensor registers will read measurements from the same sampling instant. Note that if burst reads are not used, the user is responsible for ensuring a set of single byte reads correspond to a single sampling instant by checking the Data Ready interrupt.

Each 16-bit accelerometer measurement has a full scale defined in *ACCEL_FS* (Register 28). For each full scale setting, the accelerometers' sensitivity per LSB in *ACCEL_xOUT* is shown in the table below.

AFS_SEL	Full Scale Range	LSB Sensitivity
0	±2*g*	16384 LSB/*g*
1	±4*g*	8192 LSB/*g*
2	±8*g*	4096 LSB/*g*
3	±16*g*	2048 LSB/*g*

Parameters:

ACCEL_XOUT 16-bit 2's complement value.

Stores the most recent X axis accelerometer measurement.

ACCEL_YOUT 16-bit 2's complement value.

Stores the most recent Y axis accelerometer measurement.

ACCEL_ZOUT 16-bit 2's complement value.

Stores the most recent Z axis accelerometer measurement.

4.19 Registers 67 to 72 – Gyroscope Measurements GYRO_XOUT_H, GYRO_XOUT_L, GYRO_YOUT_H, GYRO_YOUT_L, GYRO_ZOUT_H, and GYRO_ZOUT_L

Type: Read Only

Register (Hex)	Register (Decimal)	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
43	67	GYRO_XOUT[15:8]							
44	68	GYRO_XOUT[7:0]							
45	69	GYRO_YOUT[15:8]							
46	70	GYRO_YOUT[7:0]							
47	71	GYRO_ZOUT[15:8]							
48	72	GYRO_ZOUT[7:0]							

Description:

These registers store the most recent gyroscope measurements.

Gyroscope measurements are written to these registers at the Sample Rate as defined in Register 25.

These gyroscope measurement registers, along with the accelerometer measurement registers, temperature measurement registers, and external sensor data registers, are composed of two sets of registers: an internal register set and a user-facing read register set.

The data within the gyroscope sensors' internal register set is always updated at the Sample Rate. Meanwhile, the user-facing read register set duplicates the internal register set's data values whenever the serial interface is idle. This guarantees that a burst read of sensor registers will read measurements from the same sampling instant. Note that if burst reads are not used, the user is responsible for ensuring a set of single byte reads correspond to a single sampling instant by checking the Data Ready interrupt.

Each 16-bit gyroscope measurement has a full scale defined in *FS_SEL* (Register 27). For each full scale setting, the gyroscopes' sensitivity per LSB in *GYRO_xOUT* is shown in the table below:

FS_SEL	Full Scale Range	LSB Sensitivity
0	± 250 °/s	131 LSB/°/s
1	± 500 °/s	65.5 LSB/°/s
2	± 1000 °/s	32.8 LSB/°/s
3	± 2000 °/s	16.4 LSB/°/s

Parameters:

GYRO_XOUT 16-bit 2's complement value.

Stores the most recent X axis gyroscope measurement.

GYRO_YOUT 16-bit 2's complement value.

Stores the most recent Y axis gyroscope measurement.

GYRO_ZOUT 16-bit 2's complement value.

Stores the most recent Z axis gyroscope measurement.

Bibliographie

[1] V. V. Abreu, « Balance-bot », 2009.

[2] « Robotics: Research: Research & Development: Hitachi Global ». [Online].

[3] Tshpo D Motswagae , A laboratory experiment based on the segway, thesis university of southern queensland,2011.

[4] « Anybots - Remote Yourself ». [Online]. Available: https://www.anybots.com/#front.[Accessed: 15-juin-2012].

[5] « Segway – The leader in personal, green transportation ». [Online]. Available: http://www.segway.com/. [Accessed: 15-juin-2012].

[6] « iBOTnow ». [Online]. Available: http://www.ibotnow.com/. [Accessed: 15-juin-2012].

[7] http://www.moveattitude.be/particuliers/le-segway/

[8] Khalil Sultan: 'Inverted Pendulum, Analysis, Design and Implementation', 2002.

[9] B. Deforge & Q. David «Asservissement en position d'un axe linéaire», projet d'automatique, 2008.

[10] C. le lann «Le PID utilisé en régulation de position et/ou de vitesse de moteurs électrique»,Projet de fin d'étude,2007.

[11] AZZA Abdelaziz, OUAZENE Hamza, Etude et implémentation sur FBGA de la commande PID, thèse de fin d'étude, université saad dahleb de Blida, 2012.

[12] M.A. Johnson & Mohammed H.Moradi «PID Control (New identification and design methods)»,Springer,2005.

[13] J.G. Ziegler, N.B. Nichols : « Optimum settings for automatic controllers. »Trans. ASME, 64, pp. 759-768, 1942.

[14] I.M Horowitz (1963) : Synthesis of Feedback Systems. Acaddemic Press, New-York.

[15] K.J. Aström, T. Hägglund: PID Controllers : Theory, Design and Tuning, Instrument Society of America, 2nd edition, 1995.

[16] Michel Etique : « Régulation automatique », mars 2006, Yverdon- les Bains.

[17] K.J.Astrom & T.Hagglund « Advanced PID Control», ISA, New York,2006.

Bibliographie

[18] Blog.position-livre.com/électronique/Arduino-une-carte-à- jouer.php.

[19] https://www.arduino.cc.

[20] http://www.mon-club-elec.fr/pmwiki_reference_ arduino/pmwiki.php?n= Main.Materiel Mega2560.

[21] Le Bras Robotisé Lycée Jean Perrin Réalisé par : Jules Sery - Jordan Vocisano – Florian Bekaert – Cyril Gervais – Leo Abraham

[22] LECHALUPÉ Julien, « Cours d'initiation à Arduino » ,Université Paul Sabatier Mai 2014.

[23] http ://domochris.canalblog.com/archives/2015/09/11/32611773.html

[24] http://www.generationrobots.com/fr/402159-gyroscope-et-accelerometre-3-axes-mpu-6050.html

[25] http://boutique.3sigma.fr/15-acc%C3%A9l%C3%A9rom%C3%A8tre-et-gyroscope-3-axes- mpu-6050.html

[26]https://emersion.fr/blog/2015/lire-les-données-d-un-gyro-accelerometre-mpu6050.

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

[28] http://fr.wikipedia.org/wiki/Pont_en_H.

Conclusion générale

Globalement la complexité de segway réside dans la rapidité et la précision des mesures, il est considéré comme une révolution technologique. Ce système utilisé des capteurs notamment le gyroscope et l'accéléromètre.

Ce projet a réussi à atteindre ses objectifs d'équilibrer un robot à deux roues autonome basé sur le modèle du pendule inversé. Pour résoudre le problème de contrôle de l'équilibre pour le système on a utilisée le contrôleur PID qui est conçu avec succès.

Le contrôleur PID est capable de contrôler les systèmes non linéaires d'équilibrage à deux roues, tel que les résultats de simulations montrent que ce contrôleur a de meilleures performances dans le contrôle du système d'équilibrage non linéaire du robot.