REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE

MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET DE LA RECHERCHE SCIENTIFIQUE

UNIVERSITE M'HAMED BOUGARA-BOUMERDES

Faculté des **S**ciences de l'**I**ngénieure

Département de la Maintenance Industrielle

Option Génie électrique

Mémoire de Fin d'Etudes

En vue de l'obtention du diplôme :

MASTER en Automatique

Thème

Plateforme de prototypage rapide pour la robotique mobile : Application à un robot Auto-balancé

Réalisé par :

KHEMISSAT Abdelmouneim

Promoteurs :

- Dr. AKROUM Hamza
- Mr. BELLOULA Abdelmalek

Année Universitaire : 2015/2016

ملخص

انصب اهتمامنا في هذه المذكرة الى تصميم و انجاز منصة نموذجية لروبوت ذاتي التوازن باستعمال **Mbed LPC 1768**.الروبوت ذاتي التوازن هو عبارة عن بندول معكوس عبارة عن نظام غير خطي و غير مستقر. في المحاكاة قمنا باستعمال استراتيجيتين :.الاولى استخدمنا متحكم خطي تربيعي**(LQR)** والثانية باستعمال **backstepping**.تم انجاز منصة حول**Mbed LPC 1768** تحتوي شريحة الكترونية للاتصالات عن بعد**(Xbee S2)** و شريحة التحكم في المحرك و التعامل مع الاشارات الاتية من مجس **Accéléromètre** و من مجس **gyroscope** .

تمت برمجة **Mbed LPC 1768** باستعمال لغة **c++** وتمت تصميم كذلك واجهة رسومية باستخدام لغة **python**

كلمات المفتاح :

البندول المعكوس , روبوت ذاتي التوازن ، أنظمة التحكم, LQR, Backstepping, PID, Xbee, python, c++

Abstract:

In the present work, we are interested in design and realization of a prototyping platform for a self-balancing robot using **Mbed LPC 1768**. Self-balancing robot is an inverted pendulum unstable and non-linear system. In the simulation, we used two strategies: First, we used a linear quadratic regulator (LQR); to get more benefit we offer a second control is a nonlinear control by the method of Backsteeping.

Platform was completed about **Mbed LPC 1768** contain Microchip telecommunications **Xbee S2** and Engine control chip and dealing with signals from a sensor **Accelerometer** and **gyroscope.**

Mbed LPC 1768 was programmed using **c++** and designed graphical interface using **python**.

Keywords:

The inverted pendulum, Self-balancing, Control systems, Backsteeping, linear quadratic control (LQR) ,PID, Xbee, python,c++.

Resumé

Dans ce travail, nous nous intéressons à une réalisation d'une plateforme de prototypage pour un robot auto-balancé à l'aide du **Mbed LPC 1768.** Le robot auto-balancé est un pendule inversé instable et non linéaire. Dans la simulation, on a utilisé deux commandes : La première commande est la commande linéaire quadratique (LQR) et la deuxième commande est une commande non linéaire par la méthode de Backsteeping. Une Plate-forme a été achevée vers **Mbed LPC 1768** qui contient un module pour la communication sans fil **Xbee S2** et un driver pour la commande du moteur et pour réagir avec les signaux du capteur **accéléromètre** et du capteur **gyroscope**

Finalement, on a programmé **Mbed LPC 1768** en utilisant **C++** et pour acquérir les données de robot sur le PC nous proposons une interface graphique réaliser avec le langage le python.

Mots clé :

Auto-balancé, le pendule inversé, régulation par PID, commande des systèmes, Backsteeping, commande linaire quadratique (LQR),Xbee, python, c++.

Remerciements

Je remercie d'abord ALLAH le tout puissant de mon avoir donné la force, la patience et la volonté pour achever ce travail.

Mes sincères remercîment à mon promoteur ***Dr. AKROUM Hamza*** *Et Mon encadreur* ***Mr. BELLOULA Abdelmalak*** *de m'avoir guidé et encouragé durant ce travail.*

Je souhaite aussi remercier tous mes enseignants du département de génie électrique et spécialement : Mr. HAMDAOUI, Mr. Yakhlef et Mr. KHALFI. Mr. IDIR.

Une mention toute particulière à mes deux meilleurs amis BIBI YOUSSOUF et AIT AMER MEZIAN OULHADJ pour leurs aides.

Mes vifs remerciements s'étendent à tous les membres de club scientifique ELECTRO pour leur soutien.

Que tous ceux qui ont contribué de près ou de loin à la réalisation de ce travail, trouvent ici ma sincère reconnaissance.

Dédicaces

Je tiens à dédier ce mémoire :

*A ma très chère **Mère** et à mon cher **Père**, en témoignage et en gratitude de leurs dévouements, de leur soutien permanent durant toutes mes années d'études, leurs sacrifices illimités, leurs réconforts moraux, eux qui ont consenti tant d'effort pour mon éducation, mon instruction et pour me voir atteindre ce but, pour tout cela et pour ce qui ne peut être dit, mes affectations sans limite.*

A ceux qui sont la source de mon inspiration et mon courage, à qui je dois de l'amour et de la reconnaissance :

A ma cher Grande Mère. (Que Dieu garde pour moi)

A mes Chère Sœurs, et mon Frère.

A mes Tentes et leurs enfants.

A toute ma famille.

A mes chers Amis : OULHADJ, YOUSSOUF, ISLAM, SAID, SAMY. Et tous mes amis du club ELECTRO.

A ceux qui ont cru en moi,

A ceux qui croient en moi,

Et à ceux qui croiront toujours en moi.

A vous tous un grand merci

ABDELMOUNEIM.

Listes des figures

Liste des tableaux

Liste des abréviations

MIT : Massacheutts Institute of Technology.

AUV : Autonomous Underwater Vehicle.

INS : systèmes de navigation inertielle.

IMU : unité de mesure inertielle.

MEMS: Micro-Electro-Mechanical Systems.

MLI Modulation de Largeur d'Impulsion.

PWM: Pulse Width Modulation.

SIMO: Single Input Multiple Outputs.

Sommaire

Introduction générale

Préambule

Nous avons déjà presque tous un jour essayé de maintenir en équilibre un grand bâton de bois sur notre index. Afin d'éviter la chute de celui-ci, nous devons déplacer notre doigt de façon à contrecarrer en permanence son basculement.

C'est exactement le même défi que propose le système automatisé du pendule inversé. Alors que cet exercice semble assez simple et instinctif pour l'homme, il sera nécessaire de définir des stratégies précises pour assurer le maintien automatisé du pendule inversé. Bien évidemment, les performances obtenues grâce à un système automatisé sont de loin supérieures à celles qui seraient obtenues par l'homme.

Le pendule inversé est un outil didactique puissant utilisé en automatique depuis maintenant plus de 50 ans. Nous pouvons d'ailleurs retrouver une vaste littérature à son sujet, puisque de nombreux services universitaires d'automatique ou de mécatronique créent un jour leur propre pendule inversé [1].

Ce type de dispositif était à l'origine utilisé afin d'illustrer et de comparer différentes techniques de régulation linéaire comme la stabilisation de systèmes instables. Par la suite, l'automatique se développant, la nature non linéaire des équations du pendule permit d'étudier et de mettre en œuvre les nouvelles techniques de régulation non linéaire qui voyaient le jour.

Objectifs

Ce mémoire sera l'occasion de finaliser une plateforme de prototypage d'un robot auto-balancé consistant en un pendule inversé dont la base peut se déplacer selon une dimension. Nous décrirons plus précisément le matériel mis à notre disposition, c'est à dire le pendule et ses équations, la chaîne de la transmission de puissance ainsi que la chaîne d'instrumentation.

Dans cette étude, nous tenterons de parcourir l'ensemble de la problématique du pendule inversé. Ce mémoire est organisé en quatre chapitres :

Le premier chapitre présente un aperçu sur la robotique mobile nous avons abordé le robot auto-balancé qui est le cas d'application de notre étude.

Le deuxième chapitre est consacré au développement du modèle de l'ensemble pendule chariot-moteur, nous avons utilisé pour ce faire le formalisme d'Euler-Lagrange.

Le troisième présente les deux types de commandes que nous avons étudiée par simulation sous Matlab/Simulink : la première est une commande par retour d'état (commande linéaire) par une linéarisation du système autour du point d'équilibre instable. Dans la deuxième méthode, nous avons utilisé le modèle non linéaire pour appliquer une loi de commande par la méthode backstepping, et l'appliquer sur l'angle d'inclinaison du pendule inversé.

Le quatrième chapitre, présente la plate-forme de prototypage et les étapes que nous avons suivies durant sa réalisation ainsi qu'une description softwares/hardwares pour le développement de notre carte de commande. Pour finir, une conclusion générale reprend les principaux apports de ce travail et présente les perspectives.

Chapitre I :

Introduction à la robotique mobile

I.1.Introduction

La robotique est un ensemble des disciplines (mécanique, électronique, automatique, informatique), elle se subdivise en deux types : les robots industriels et les robots mobiles. Les robots industriels sont généralement fixes, ils sont utilisés dans des nombreuses applications industrielles : l'assemblage mécanique, la soudure, la peinture, etc. Les robots mobiles sont classifiés selon la locomotion (à roues, à chenille, etc.) et le domaine d'application (déminage, surveillance, spatiale, nucléaire, etc.) [2].

Concernant la robotique de service et en particulier d'intervention, son rôle consiste à intervenir là où l'homme ne peut ou ne veut pas intervenir lui-même. Les robots utilisés sont dotés d'outils nécessaires à l'intervention (capteurs, bras de robot, canon à eau, effecteur, fusil, etc.) et doivent pouvoir se déplacer pour atteindre le lieu d'intervention et l'explorer à distance via un dispositif de contrôle fiable. Ce dernier permet de traiter plusieurs fonctions d'où l'accomplissement d'un certain nombre de tâches (parfois non connues à l'avance, et susceptibles d'évoluer dans le temps) sous contrôle d'un opérateur (artificier) humain réduit, [3] situé hors de la zone hostile ou derrière des protections.

Dans ce présent chapitre, nous présentons dans un cadre général la robotique de service et en particulier celle d'interventions de déminage dont nous citons quelques exemples de robots utilisés, puis nous étalerons la problématique en question qui est relative à la commande à distance de robot d'intervention, puis nous passons vers l'étude critique de l'existant et nous parachevons par une conclusion.

I.2.Introduction aux robots mobile1 et manipulateur

La robotique est un très bon exemple de domaine pluri-disciplinaire qui implique de nombreuses thématiques telles que la mécanique, la mécatronique, l'électronique, l'automatique, l'informatique ou l'intelligence artificielle. En fonction du domaine d'origine des auteurs, il existe donc diverses définitions du terme robot, mais elles tournent en général autour de celle-ci : Un robot est une machine équipée de capacités de perception, de décision et d'action qui lui permettent d'agir de manière autonome dans son environnement en fonction de la perception qu'il en a.

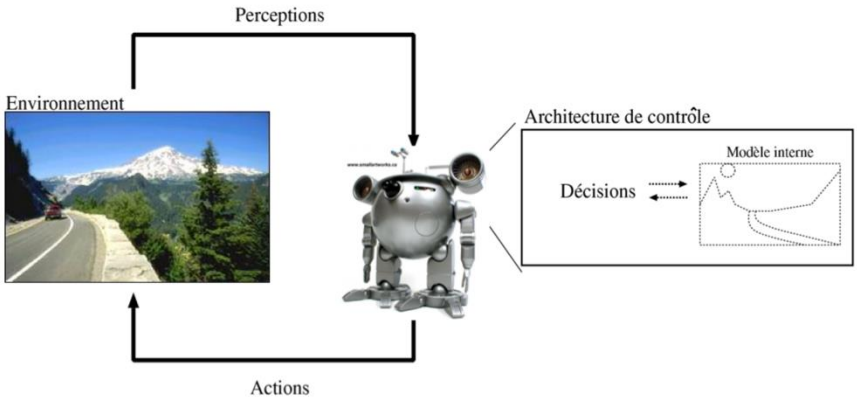


Figure I. 1 Schéma des interactions d'un robot avec son environnement. Selon les approches, un modèle interne de l'environnement peut être utilisé ou non *[4]*.

Cette définition s'illustre par un schéma classique des interactions d'un robot avec son environnement (Figure **I.**1). Les différentes notions que nous présenterons dans ce cours sont essentiellement issues de cette vision de la robotique, très orientée sur l'Intelligence Artificielle, qui place au centre des préoccupations l'enchaînement de ce cycle Perception/Décision/Action. La manière dont un robot gère ces différents éléments est définie par son architecture de contrôle, qui la plupart du temps va faire appel à un modèle interne de l'environnement qui lui permettra de planifier ses actions à long terme [4].

I.2.Historique

Le terme de robot apparaît pour la première fois dans une pièce de Karel Capek en 1920 : Rossum's Universal Robots. Il vient du tchèque 'robota' (~ servitude) et présente une vision des robots comme serviteurs dociles et efficaces pour réaliser les taches pénibles mais qui déjà vont se rebeller contre leurs créateurs [4].

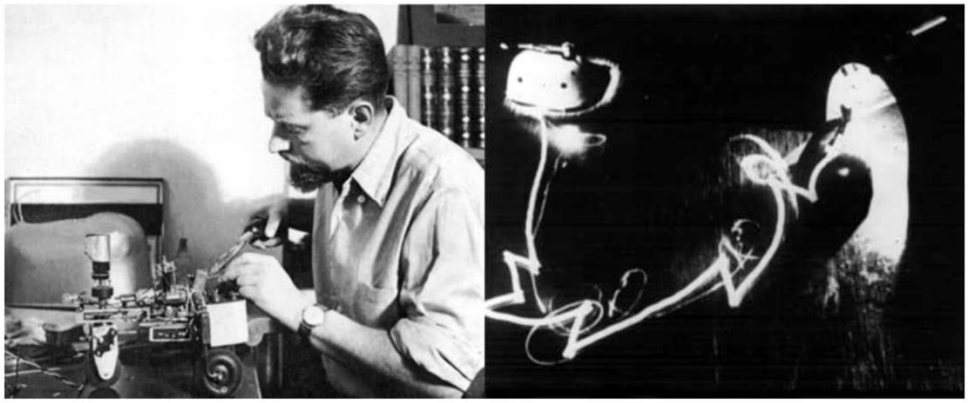


Figure I. 2 La tortue de Grey Walter (nommée "machina speculatrix" et surnommée Elsie) et une illustration de sa trajectoire pour rejoindre sa niche. *[4].*

La Tortue construite par Grey Walter dans les années 1950 (Figure I.2), est l'un des premiers robots mobiles autonomes. Grey Walter n'utilise que quelques composants analogiques, dont des tubes à vide, mais son robot est capable de se diriger vers une lumière qui marque un but, de s'arrêter face à des obstacles et de recharger ses batteries lorsqu'il arrive dans sa niche. Toutes ces fonctions sont réalisées dans un environnement entièrement préparé, mais restent des fonctions de base qui sont toujours des sujets de recherche et de développement technologiques pour les rendre de plus en plus génériques et robustes.

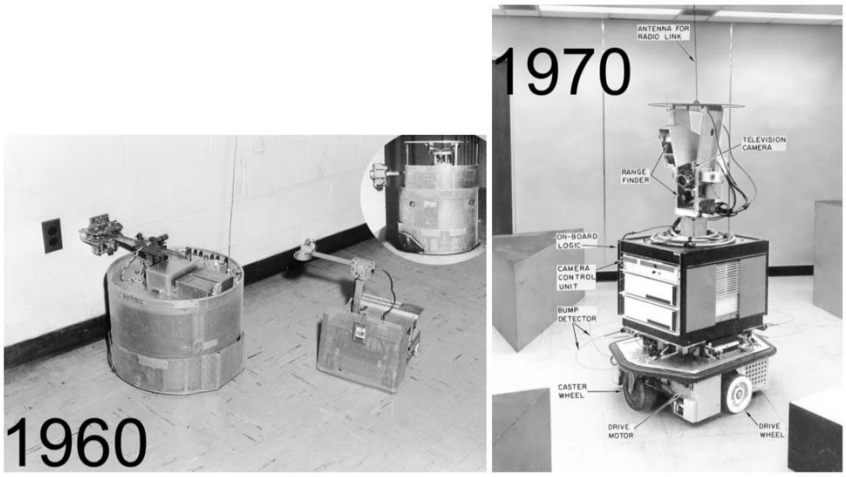


Figure I. 3 A gauche : Robot "Beast" de l'université John Hopkins dans les années 1960. A droite : Le robot Shakey de Stanford en 1969 a été une plate-forme de démonstration des recherches en intelligence artificielle *[4]*.

Dans les années 60, les recherches en électronique vont conduire, avec l'apparition du transistor, à des robots plus complexes mais qui vont réaliser des tâches similaires. Ainsi le robot "Beast" (Figure I.3) de l'université John Hopkins est capable de se déplacer au centre des cou- loirs en utilisant des capteurs ultrason, de chercher des prises électriques (noires sur des murs blancs) en utilisant des photodiodes et de s'y recharger.

Les premiers liens entre la recherche en intelligence artificielle et la robotique apparaissent à Stanford en 1969 avec Shakey (Figure I.3). Ce robot utilise des télémètres à ultrason et une caméra et sert de plate-forme pour la recherche en intelligence artificielle, qui à l'époque travaille essentiellement sur des approches symboliques de la planification. La perception de l'environnement, qui à l'époque est considérée comme un problème séparé, voire secondaire, se révèle particulièrement complexe et conduit là aussi à de fortes contraintes sur l'environnement. Ces développements de poursuivent avec le Stanford Cart dans la fin des années 1970, avec notamment les premières utilisations de la stéréovision pour la détection d'obstacles et la modélisation de l'environnement. En France, le robot Hilare est le premier robot construit au LAAS, à Toulouse (Figure I.4) [4].

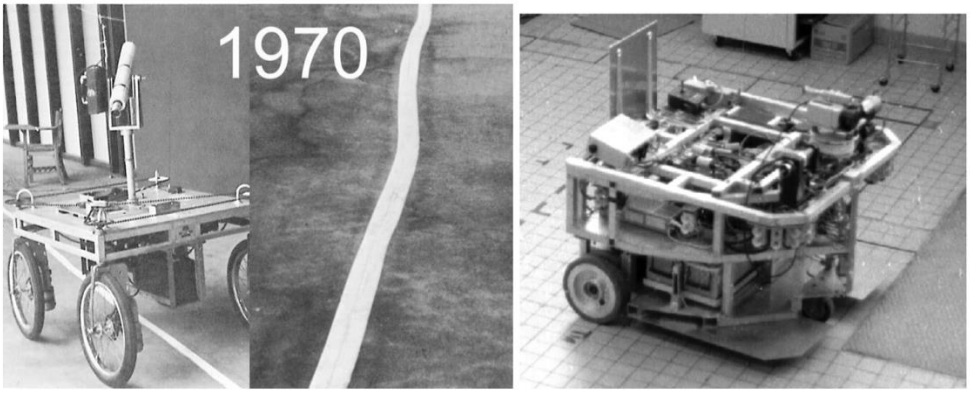


Figure I. 4 Le Stanford Cart date de la fin des années 1970. *Le robot Hilare du LAAS a été construit en 1977.* [4].

Une étape importante est à signaler au début des années 1990 avec la mise en avant de la robotique réactive, représentée notamment par Rodney Brooks. Cette nouvelle approche de la robotique, qui met la perception au centre de la problématique, a permis de passer de gros robots très lents à de petits robots (Figure I.5), beaucoup plus réactifs et adaptés à leur environnement. Ces robots n'utilisent pas ou peu de modélisation du monde, problématique qui s'est avérée être extrêmement complexe.

Figure I. 5 Genghis, développé par Rodney Brooks au MIT au début des années 1990. [4].

Ces développements ont continué et l'arrivée sur le marché depuis les années 1990 de plates- formes intégrées telles que le pioneer de la société Mobile Robots a permis à de très nombreux laboratoires de travailler sur la robotique mobile et à conduit à une explosion de la diversité des thèmes de recherche. Ainsi, même si les problèmes de déplacement dans l'espace et de modélisation de l'environnement restent difficiles et cruciaux, des laboratoires

ont pu par exemple travailler sur des approches multi-robot, la problématique de l'apprentissage ou sur les problèmes d'interactions entre les hommes et les robots.

I.3.Exemples d'applications :

Aujourd'hui, le marché commercial de la robotique mobile est toujours relativement restreint en dehors des robots aspirateurs vendus à plusieurs millions d'exemplaires. Cependant, il existe de nombreuses perspectives de développement qui en feront probablement un domaine important dans le futur. Les applications des robots peuvent se trouver dans de nombreuses activités "ennuyeuses, salissantes ou dangereuses" (3D's en anglais pour Dull, Dirty, Dangerous), mais également pour des applications ludiques ou de service, comme l'assistance aux personnes âgées ou handicapées [4].

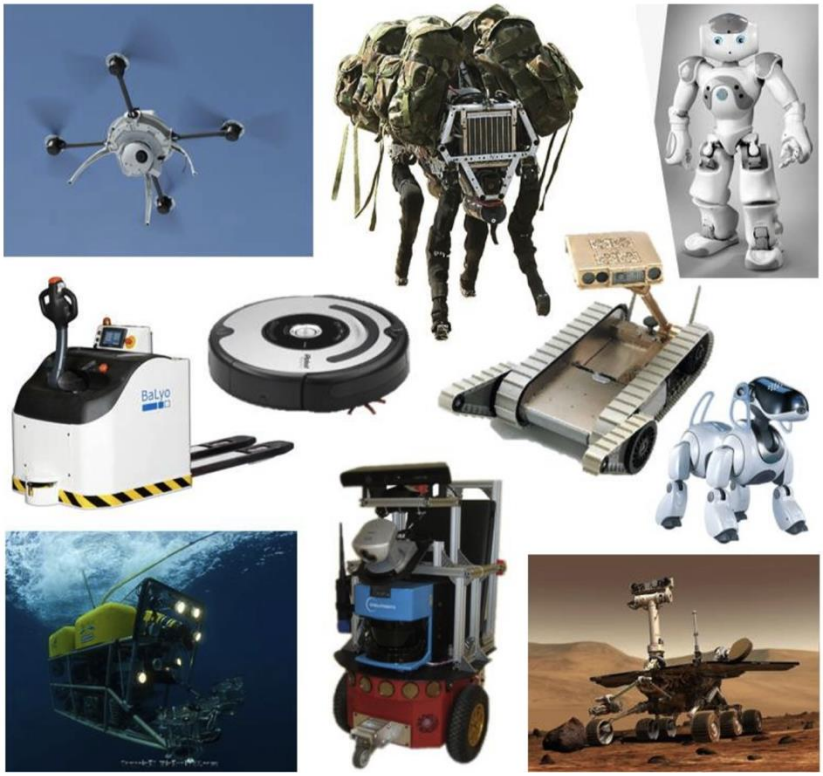


Figure I. 6 Exemples de robots commerciaux ou de recherche *[4]*.

Parmi les domaines d'applications possibles de la robotique, citons :

– La robotique de service (hôpital, bureaux, maison).

– La robotique de loisir (jouets, robot 'compagnon').

– La robotique industrielle ou agricole (entrepôts logistiques, récolte de productions agricoles, mines).

– La robotique en environnement dangereux (spatial, industriel, militaire, catastrophes naturelles).

A cela, s'ajoute à l'heure actuelle des nombreuses plates-formes conçues essentiellement pour les laboratoires de recherche. La figure I.6 montre quelques exemples de robots existants.

I.4.Les différents types des robots mobiles

Il est possible de classifier les robots mobiles de nombreuses manières et selon de nombreux critères tels : le domaine d'application, le degré d'autonomie, le degré d'intelligence, etc. Nous avons préféré les classifier, puisque leur mobilité est l'une de leurs caractéristiques essentielles, selon leur mode de locomotion [4].

On peut classer les robots mobiles en cinq catégories selon leur mode de locomotion :

- Les robots mobiles à roues (wheeled robots).
- Les robots mobiles à pattes.
- Les robots mobiles à chenilles.
- Les robots mobiles volants (drones).
- Les robots mobiles navigants (marins ou sous-marins).

I.4.1. Les robots mobiles à roues :

Ils sont les plus répandus car les plus simples à concevoir. Il existe deux grands types de robots mobiles à roues selon leur mode de direction :

- Les robots mobiles à roues différentielles (differential drive).
- Les robots mobiles à roues avec essieux classique (steer drive).
- Les robots mobiles à roues avec essieux (steer drive configuration) : Ils sont pour la plupart utilisés dans des applications éducatives car faciles à mettre en œuvre à base de châssis de modèles réduits automobiles. Le mouvement arrière du robot est assuré par deux roues motrices. La direction quant à elle est calquée sur le modèle automobile. La commande de ce type de robots est assez simple mais leur capacité en termes de mouvement est réduite. En effet, leur comportement dit « non holonomique » les contraint à des manœuvres compliquée (créneau) et limite considérablement leur angle de braquage. C'est pour cette raison qu'ils sont très peu

utilisés dans les projets de recherche actuels (mis à part quelques projets militaire visant à robotiser des jeeps de reconnaissance).

- Robots mobiles à roues différentielles (différentiel drive configuration) : Les robots de ce type possèdent deux roues motrices possédant chacune son moteur et placées en parallèle l'une de l'autre. Leurs vitesses peuvent donc être commandées indépendamment. En jouant sur les vitesses de chaque roue, on obtient une gamme pratiquement illimitée de mouvements. Malgré une commande plus complexe que celle des robots à roues classiques, leurs capacités en termes de braquage mais aussi la possibilité de tourner sur eux-mêmes fait des robots mobiles à roues différentielles le type de robots mobiles à roues le plus adapté à des applications industrielles.

I.4.2. Les robots mobiles à pattes

On distingue notamment et selon le nombre de pattes :

Les robots à une patte : Un robot doté d'une seule patte a été assemblé au Massacheutts Institute of Technology (MIT). Le robot sautille sur une patte orientable à deux degrés de liberté, dotée d'un actionneur linéaire pour la propulsion verticale. Il n'est pas autonome.

Les robots à deux patte : Les robots à deux pattes ou bipèdes sont surtout étudiés au Japon ou les progrès dans ce segment sont assez signifiants. En 1997, Honda et son humanoïde P2, font rêver le monde entier de robots domestiques.

Les robots à quatre pattes : Encore une fois, c'est au Japon qu'il faut chercher les robots quadrupèdes les plus évolués : le robot de compagnie AIBO (robot chien) de la firme SONY, notamment dans sa seconde génération est assez « vivant ». Il reproduit en effet une large palette de comportements « canins ».

Les robots à six ou huit pattes : Ce sont les robots à pattes les plus répandus à travers le monde. Avec leurs six pattes ils sont statiques mais aussi facile à programmer et mettre en œuvre, chaque patte n'ayant besoin que de deux degrés de liberté.

Les robots « mille-pattes » : Ces robots sont très utilisés dans l'entretien et la réparation des conduites et autres endroits inaccessibles. La demande professionnelle pour ce type de robots est très importante.

Robots mobiles à chenille : Les chenilles assurent à un mobile une meilleure adhérence au sol. Elles sont employées lorsque le sol est perturbé, essentiellement en environnement extérieur. La commande est de type différentiel (differential drive). On peut citer le robot ANT de la West Virginia University.

Robots volants (drones) : Les drones, petits avions autoguidés ou téléguidés ont de nombreuses applications militaires et éventuellement civiles (surveillance d'autoroutes). Des projets visent à réaliser un avion autonome de moins de 20cm d'envergure, transportant une caméra infrarouge et un transmetteur radio. Il est plus facile de faire voler un avion de 50cm d'envergure ne pesant que quelques grammes (show flyer) ; ces avions indoor sont délicats à construire et fragiles à manipuler mais ils résistent aux obstacles, à cause de leur faible inertie. Le vol stationnaire est encore plus facile à maitriser. Le concours de robots volants organisé chaque année à Atlanta en Georgie montre combien il est difficile de stabiliser en vol un hélicoptère et de commander ses déplacements avec assez de précision.

Robots navigants : La construction d'un robot bateau ou sous-marin est facilitée par le fait que le poids n'est pas un facteur limitatif à la mobilité. Les torpilles sont bien connues des militaires, mais il y a peu de recherches civiles qui ont été lancées sur les robots mobiles sous-marins appelés AUV (Autonomous Underwater Vehicle). Ces derniers sont notamment utilisés dans le secteur pétrolier (off-shore) et dans l'exploration sous-marine.

I.5.Robot auto-balancé (pendule inversé).

C'est le cas d'application de notre projet, Le robot auto-balancé est constitué d'une plate-forme solidaire de deux roues. L'axe des roues est perpendiculaire à l'axe de déplacement du robot [5].

La masse M est supposée être un homme en équilibre sur la plate-forme. Le principe du contrôle de position est basé sur le pendule inversé. L'inclinaison de la plate-forme d'un angle alpha, va provoquer le déplacement du robot, de telle sorte que l'angle redevienne nul. D'où le nom de " Robot Auto-Balancé".

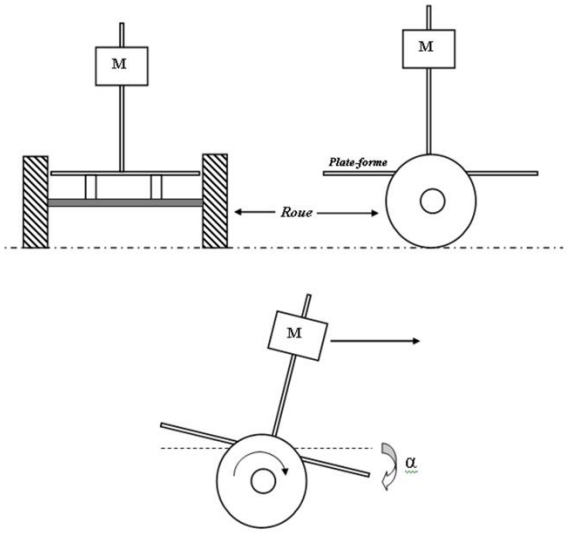


Figure I. 7 Robot Auto-Balancé [5].

Si le dispositif réagit suffisamment vite, l'angle est en permanence proche de zéro. La plate-forme est donc toujours horizontale et l'utilisateur n'est pas obligé de maintenir son équilibre. Un déplacement du centre de gravité vers l'avant obligera le robot à se déplacer dans le même sens pour maintenir l'équilibre. Ceci est valable dans les deux sens de déplacement.

I.6. Conclusion

L'objectif de notre étude consiste principalement à l'étude et réalisation d'une plateforme de prototypage rapide pour la robotique mobile.

Pour cela nous avons fait une généralité sur les différents robots mobiles, ainsi que nous avons définie et énoncée le robot auto-balancé qu'il est le cas d'application de notre étude.

Chapitre II :

Modélisation du pendule inversé

II .1. Introduction :

Le problème de commande exige la recherche du modèle mathématique du système à commander, donc une phase de modélisation est nécessaire pour permettre l'étude en simulation.

La modélisation consiste à représenter les aspects importants du système en décrivant les relations entre ses différentes grandeurs par des équations mathématiques. L'ensemble de ces relations constitue le modèle de ce système.

Le pendule inversé est constitué d'un chariot mobile sur un rail et d'un pendule suspendu sur le chariot. Son principe de fonctionnement est très simple en théorie : Après avoir ramené le pendule de sa position d'équilibre basse à la position verticale haute, il faut le maintenir dans cette position. Pour cela, quand le pendule penche vers la droite, le chariot doit le rattraper en effectuant un mouvement vers la droite, et inversement. Ce système présent des caractéristiques intéressantes permettant d'illustrer quelques problèmes types, il s'agit d'un système SIMO (Single Input Multiple Output), instable décrit par un modèle non linéaire. [6]

Dans ce chapitre, nous allons d'abord aborder la présentation du banc d'essais du pendule inversé ainsi que le dispositif de contrôle et de commande. Puis, à partir du formalisme d'Euler-Lagrange nous développons son modèle dynamique non linéaire qui pose un problème d'instabilité au point d'équilibre instable $\theta = 0$. Nous développons aussi le modèle du moteur à courant continu. Nous terminons par la linéarisation du modèle du pendule inversé associé au moteur autour du point d'équilibre instable.

II.2. Présentation du pendule inversé

II .2.1. Description du banc d'essais

Le but de la commande du pendule inversé est de maintenir en équilibre vertical une tige en aluminium à l'extrémité de laquelle est montée une masse de forme cylindrique. Cette tige est fixée par une articulation pivotante sur un chariot qui peut se déplacer en glissant le long d'un rail de guidage horizontale [6]. Le mouvement de rotation d'un moteur électrique est transformé en mouvement de translation du chariot par l'intermédiaire d'une poulie et d'une courroie crantée. Le déplacement du chariot dans un sens ou dans l'autre assure par réaction l'équilibre vertical du bras du pendule. La figure II.1 montre les éléments mécaniques principaux du dispositif.

Initialement le pendule est en position basse, le but étant de le redresser en position haute et surtout le maintenir dans cette position.

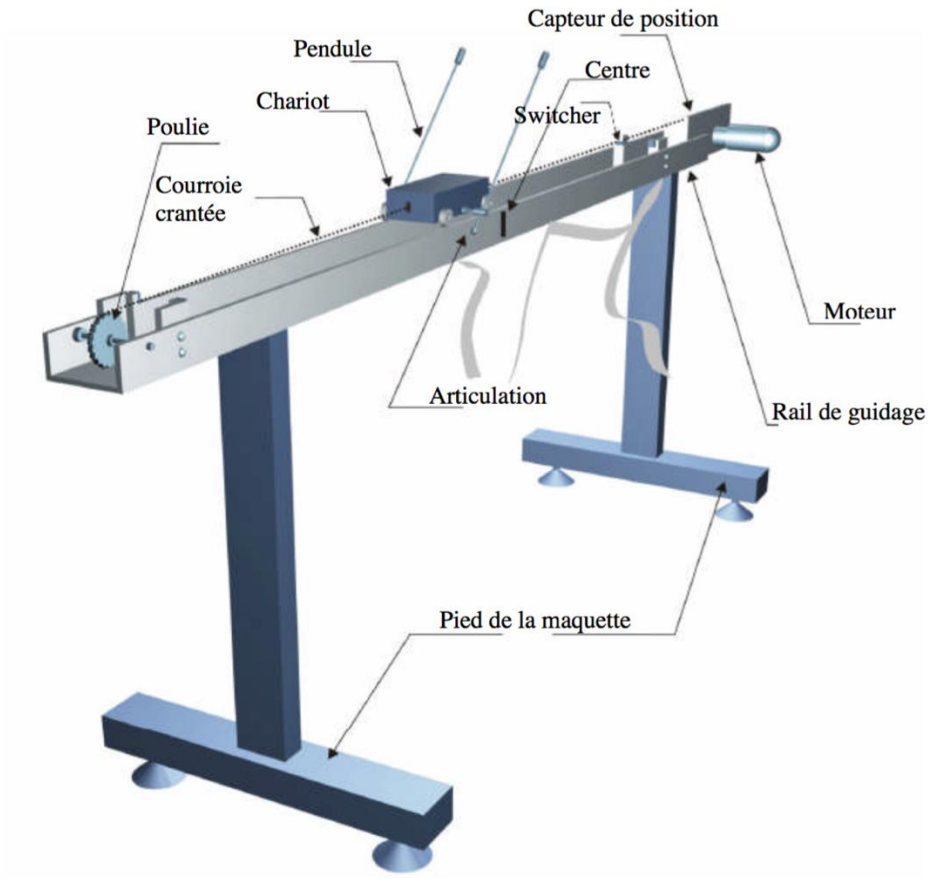


Figure II.1 synoptique de banc d'essais du pendule inversé *[6].*

II.3. Modélisation de l'ensemble chariot-pendule

L'ensemble du chariot-pendule possède deux degrés de liberté dont les coordonnées généralisées sont respectivement : x pour le déplacement horizontal du chariot et θ pour la rotation du pendule. La direction positive de x est le sens à droite en mètre et celui de l'angle est le sens des aiguilles d'une montre en radian (figure II.2).

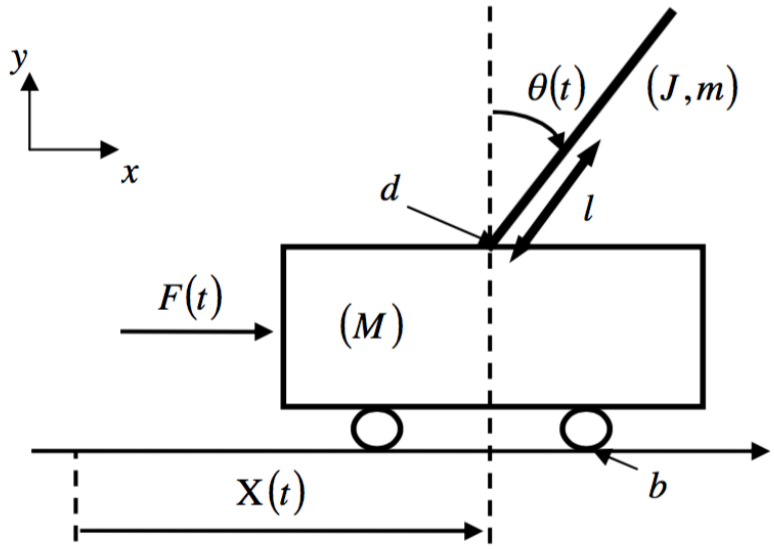


Figure II. 2 Schémas de l'ensemble chariot et pendule inversé

II .3.1. Les paramètres de l'ensemble pendule-chariot

m 0.2Kg : représente la masse du pendule, b 0.00005 Ns m : Coefficient de frottement des roues du chariot, M 2.3Kg : Masse du chariot, x: Position du chariot (m), l 0.3 m : Demi longueur du pendule, Angle de rotation du pendule(rad), F Force exercée sur le chariot (N), g 9.81 m s ^{2} : Intensité de la pesanteur, d 0.005 Nms rad :

Coefficient de frottement de rotation du pendule [6].

Les équations du mouvement du pendule sont déterminées par le formalisme d'Euler-Lagrange [18-6] qui est basé sur le principe de la conservation de l'énergie mécanique plutôt que sur le concept de force, comme dans le principe de Newton.

Le Lagrangien est défini comme étant la différence entre l'énergie cinétique E_C et l'énergie

potentielle E_p du système, l'avantage de ce formalisme réside dans l'élimination des efforts d'interaction, Il s'exprime par :

$$L = E_c - E_p \tag{II.1}$$

II .3.2. Energie cinétique du système en mouvement

Le système en mouvement comporte le chariot qui se déplace linéairement sur les rails et le pendule qui se balance sur son axe de rotation. L'énergie cinétique du chariot en mouvement est donnée par l'équation : [19]

$$E_{cM} = \frac{1}{2} M\dot{x}^2 \tag{II.2}$$

L'énergie cinétique du pendule est exprimée par l'équation

$$E_{cm} = \frac{1}{2} m{v_c}^2 + \frac{1}{2} J\dot{x}^2 \tag{II.3}$$

v_c: La vitesse de centre de gravité du pendule.

$\dot{\theta}$: La vitesse angulaire du pendule.

La position du centre de gravité du pendule, notée r_C à partir de ces coordonnées est donnée par :

$$r_c = (x + l\sin\theta)\hat{\imath} + l\cos\theta\hat{\jmath} \tag{II.4}$$

$\hat{\imath}$, $\hat{\jmath}$: étant les vecteurs unitaires du repère x, y

La vitesse du centre de gravité du pendule est donc

$$v_c = \frac{dr_c}{dt} = \left(\dot{x} + l\cos\theta\,\dot{\theta}\right)\hat{\imath} - l\sin\theta\,\dot{\theta}\hat{\jmath} \tag{II.5}$$

En substituant les équations (II.4) et (II.5) dans l'équation (II.3), l'expression de l'énergie cinétique du pendule devient :

$$E_{cm} = \frac{1}{2}m\left(\dot{x}^2 + 2\dot{x}l\cos\theta\,\dot{\theta} + l^2\dot{\theta}^2(\cos^2\theta + \sin^2\theta)\right) + \frac{1}{2}J\dot{\theta}^2 \quad \text{(II.6)}$$

Qui s'écrit après simplification, du terme $l^2\dot{\theta}^2(\cos^2\theta + \sin^2\theta) = l^2\dot{\theta}^2$:

$$E_{cm} = \frac{1}{2}m\left(\dot{x}^2 + 2\dot{x}l\cos\theta\,\dot{\theta} + l^2\dot{\theta}^2\right) + \frac{1}{2}J\dot{\theta}^2 \quad \text{(II.7)}$$

Finalement, l'énergie cinétique totale de l'ensemble chariot et pendule est exprimée par :

$$E_c = E_{cM} + E_{cm} = \frac{1}{2}M\dot{x}^2 + \frac{1}{2}m\left(\dot{x}^2 + 2\dot{x}l\cos\theta\,\dot{\theta} + l^2\dot{\theta}^2\right) + \frac{1}{2}J\dot{\theta}^2 \quad \text{(II.8)}$$

Energie potentielle du système

Le chariot étant en mouvement sur un rail horizontal, seul le pendule en mouvement possède une énergie potentielle. L'énergie potentielle du centre de gravité du pendule est donnée par :

$$E_p = mgl\cos\theta \quad \text{(II.9)}$$

Maintenant que les expressions de l'énergie cinétique E_C de l'ensemble chariot-pendule

(Equation II.8) et l'énergie potentiel du système E_p (Equation II.9) sont établies, on utilise

L'équation générale d'Euler-Lagrange pour déterminer les équations de mouvement de l'ensemble chariot-pendule.

En substituant les équations (II.8) et (II.9) dans l'équation (II.1) on trouve :

$$L = \frac{1}{2}M\dot{x}^2 + \frac{1}{2}m(\dot{x}^2 + 2\dot{x}l\cos\theta\ \dot{\theta} + l^2\dot{\theta}^2) + \frac{1}{2}J\dot{\theta}^2 - mgl\cos\theta \quad \text{(II.10)}$$

L'équation générale d'Euler-Lagrange est donnée par :

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{\xi}_j}\right) - \frac{\partial L}{\partial \xi_j} + \frac{\partial D_f}{\partial \dot{\xi}_j} = F_j \quad \text{(II.11)}$$

Où ξ : désigne les degrés de liberté. Dans le cas du pendule, ces deux degrés de liberté sont la position du chariot x(t) et l'angle de rotation du pendule du pendule $\theta(t)$: D_f désigne l'énergie dissipée par frottement, F : la force généralisée, L : représente le Lagrangien, il est donné par l'équation (II.1).

On définit donc l'équation de Lagrange pour le pendule inversé comme suit :

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{\xi}_j}\right) - \frac{\partial L}{\partial \xi_j} = F_j \qquad \text{(II.12)}$$

Pour le degré de liberté $\xi(t) = x(t)$, on a :

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{x}}\right) - \frac{\partial L}{\partial x} = F - b\dot{x} \qquad \text{(II.13)}$$

Où F : représente la force exercée sur le chariot.

La dérivée partielle du lagrangien suivant $\dot{x}$ et t s'écrit :

$$\frac{d}{dt}\left(M\dot{x} + m\dot{x} + mlcos\theta\ \dot{\theta}\right) - 0 = F = b\dot{x} \qquad \text{(II.14)}$$

Donc, la première équation de Lagrange est :

$$(M+m)\ddot{x} + ml\sin\theta\ \ddot{\theta} - ml\sin\theta\ddot{\theta}^2 \qquad \text{(II.15)}$$

Pour le degré de liberté $\xi(t) = \theta(t)$, on a :

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{\theta}}\right) - \frac{\partial L}{\partial \theta} = -d\dot{\theta} \qquad \text{(II.16)}$$

La dérivée partielle du lagrangien suivant $\dot{\theta}$ et t s'écrit :

$$\frac{d}{dt}\left(+ml\dot{x}\cos\theta + ml^2\dot{\theta} + j\dot{\theta}\right) - \left(-ml\dot{x}\sin\theta\ \dot{\theta} + mgl\sin\theta\right) = -d\dot{\theta} \qquad \text{(II.17)}$$

Donc la deuxième équation de Lagrange est :

$$\left(ml^2 + j\right)\ddot{\theta}\right) + ml\ddot{x}\cos\theta - ml\dot{x}\sin\theta\ \dot{\theta} + ml\dot{x}\sin\theta\ \dot{\theta} = mgl\sin\theta = -d\dot{\theta} \qquad \text{(II.18)}$$

Le modèle de connaissance de l'ensemble chariot-pendule (Equation II.17 et II.18) est donné par le système d'équations :

$$\begin{cases} h\ddot{x} + b\dot{x} + ml\cos\theta\ddot{\theta} - ml\sin\theta\dot{\theta}^2 = F \\ ml\ddot{x}\cos\theta + N\ddot{\theta} + d\dot{\theta} - mgl\sin\theta = 0 \end{cases} \qquad \text{(II.19)}$$

Où $h = M + m$, $N = ml^2 + j$

Le système d'équations (II.19) montre la dépendance qui existe entre l'accélération du chariot $\ddot{x}$ et l'accélération angulaire du pendule $\ddot{\theta}$. Ainsi, lorsque F = 0 (pas de force extérieur), si on déplace le pendule de sa position d'équilibre, il ne sera soumis qu'à sa propre inertie, il se mettrait alors à osciller et, puisque le pendule est fixé sur le chariot, ce dernier commencerai également à se mouvoir.

II.4. Modélisation du moteur électrique à courant continu à aimant permanent commandé par l'induit

Comme le moteur utilisé pour entrainer le chariot est un moteur à courant continu à aimant permanent, le flux inducteur est constant. Le schéma électrique et mécanique équivalent de l'induit est donné par la figure II.3 [6].

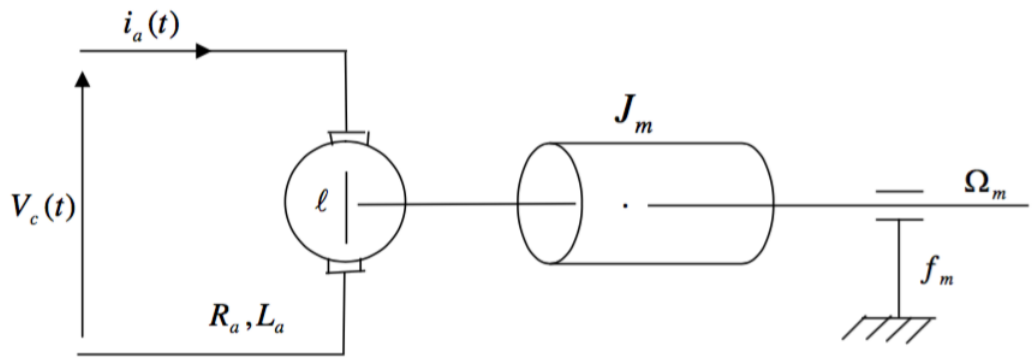


Figure II. 2 Schéma électrique et mécanique de l'induit *[6].*

II .4.1. Les paramètres de la machine :

$R_a = 2.5\Omega$: Résistance de l'induit, L_a =0.0025 H : Inductance de l'induit, $-2.5 \leq V_c \leq 2.5$: La tension d'alimentation de l'induit du moteur (V) , i_a (t) : Intensité du courant (A), K_b=0.05 N/A : Constante électrique du moteur, K_m =0.05 Nm : Constante mécanique, $\ell(t)$: fcem, J_m =1.4×10^{-5} Kg.m^2 : moment d'inertie , C_r (t): Couple résistant, C_m (t) : Couple moteur, f_m = 10^{-6} Kg.m^2/s : Coefficient de frottement visqueux, Ω_m : vitesse angulaire de l'arbre de moteur [18].

Les équations régissant le fonctionnement du moteur à courant continu à aimant permanent sont :

II .4.2. Les équations électriques :

Equation de l'induit :

$$v(t) = R_a\, i_a(t) + L_a\, \frac{di_a(t)}{dt} + \ell(t) \qquad \text{(II.20)}$$

Equation de la fcem :

$$\ell(t) = K_b \Omega_m(t) \qquad \text{(II.21)}$$

Equation mécanique :

$$C_m(t) = J_m \frac{d\Omega_m(t)}{dt} + C_r(t) + f_m \Omega_m(t) \qquad \text{(II.22)}$$

Equation de couple :

$$C_m(t) = K_m i_a(t) \qquad \text{(II.23)}$$

II .4.3. Modèle d'état du moteur

En l'absence du couple résistant (C_r(t) = 0) , et en considérant la vitesse angulaire de l'arbre de moteur comme sortie, puis en remplaçant l'équation (II.23) dans l'équation (II.22) et l'équation (II.21) dans l'équation (II.20) on obtient .

$$\frac{d\Omega_m(t)}{dt} = \frac{-f_m}{J_m}\Omega_m(t) + \frac{K_m}{J_m} i_a(t)$$

$$\frac{di_a(t)}{dt} = \frac{-K_b}{L_a}\Omega_m(t) - \frac{R_a}{L_a} i_a(t) + \frac{1}{L_a} V_c(t)$$

$$y_m = \Omega_m \qquad \text{(II.24)}$$

Ces équations, écrites sous la forme matricielle, permettent d'obtenir le modèle d'état de la machine, il est donné par :

$$\begin{cases} \dot{z} = Az + BV_c \\ y = Cz \end{cases} \qquad \text{(II.25)}$$

avec $z=[\Omega_m, i_a]^T$

$$A=\begin{bmatrix} \frac{-f_m}{J_m} & \frac{K_m}{J_m} \\ \frac{-K_b}{L_a} & \frac{-R_a}{L_a} \end{bmatrix}, B=\begin{bmatrix} 0 \\ \frac{1}{L_a} \end{bmatrix}, C=[1 \quad 0]$$

(II.26)

Si on considère le courant de l'induit comme étant la sortie, l'équation dynamique du modèle reste la même, par contre la matrice de sortie devient :

$$C=(0 \quad 1) \tag{II.27}$$

II.5. Modélisation du système global : moteur-chariot-pendule

Le modèle (II.19) décrit le régime transitoire de l'ensemble pendule-chariot lorsque l'entrée est une force extérieure F. Dans le banc d'essai que nous utilisons, la force F est développée par un moteur à courant continu. Le schéma de la figure II.4 illustre la relation existante entre le moteur commandé par la tension V_C (t) et la force F permettant l'entraînement du chariot donc le balancement du pendule. [6-18].

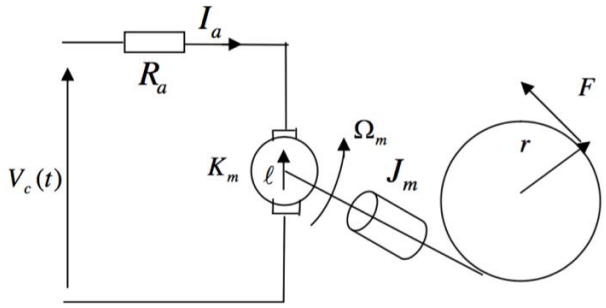


Figure II. 3 Relation entre la force mécanique F et la tension Vc *[6]*

II .5.1.Expression de la force F en fonction de la tension d'alimentation du moteur V_c.

Pour déterminer cette expression et afin de ne pas rendre plus complexe le modèle global de l'ensemble pendule-chariot-moteur, on néglige la dynamique du moteur, par conséquent, l'équation électrique (II.20) devient en tenant compte de l'équation (II.21) [19-6] :

$$V_c = R_a i_a + K_b \Omega_m \tag{1.28}$$

Et comme $x = r\Phi$.

Φ Étant la position angulaire d'un point quelconque située sur le périmètre de la poulie, et r=0.0027 m : le rayon de la poulie, on obtient :

$$\frac{d}{dt}\, x(t) = r\frac{d}{dt}\Phi(t) = r\Omega_m(t) \tag{II.29}$$

La vitesse angulaire du moteur Ω_m s'exprime donc par rapport à la vitesse du chariot $\dot{x}$ par la relation.

$$\Omega_m = \dot{x}/r \tag{II.30}$$

En substituant l'équation (II.30) dans l'équation (II. 28), l'expression du courant de l'induit I_a s'écrit

$$I_a = \frac{V_c}{R_a} - \frac{K_b}{rR_a}\dot{x} \tag{II.31}$$

Le couple produit à l'arbre du moteur (équation (II.23)) crée une force qui est donnée par

$$F = \frac{C_m}{r} = \frac{K_m I_a}{r} \tag{II.32}$$

En remplaçant l'équation (II.31) dans l'équation (II.32), on obtient finalement :

$$F = \frac{K_m}{rR_a}V_c - \frac{K_m K_b}{r^2 R_a}\,\dot{x} \tag{II.33}$$

1.1.1 Modèle d'état de l'ensemble moteur-chariot -pendule

Afin d'obtenir le modèle d'état de l'ensemble moteur-chariot-pendule, on utilise une nouvelle fois le modèle (II.19). Le vecteur d'état est :

$$z = [z_1 \quad z_2 \quad z_3 \quad z_4]^T = \left[x \quad \dot{x} \quad \theta \quad \dot{\theta}\right]^T \tag{II.34}$$

Après quelques manipulations mathématiques, on obtient le modèle non linéaire (II.35) ou la grandeur de commande est la force F.

$$
\begin{cases}
\dot{z}_1 = z_2 \\
\dot{z}_2 = \dfrac{-bN}{hN - m^2 l^2 \cos^2 z_3} z_2 - \dfrac{m^2 l^2 g}{hN - m^2 l^2 \cos^2 z_3} \cos z_3 \sin z_3 \\
\quad + \dfrac{mld \cos z_3}{hN - m^2 l^2 \cos^2 z_3} z_4 + \dfrac{mlN \sin z_3}{hN - m^2 l^2 \cos^2 z_3} z_4^2 + \dfrac{FN}{hN - m^2 l^2 \cos^2 z_3} \\
\dot{z}_3 = z_4 \\
\dot{z}_4 = \dfrac{mgl}{N} \sin z_3 - \dfrac{d}{N} z_4 + \dfrac{mlb \cos z_3}{hN - m^2 l^2 \cos^2 z_3} z_2 + \dfrac{m^3 l^3 g \cos^2 z_3 \sin z_3}{N\left(hN - m^2 l^2 \cos^2 z_3\right)} \\
\quad - \dfrac{m^2 l^2 d \cos^2 z_3}{N\left(hN - m^2 l^2 \cos^2 z_3\right)} z_4 - \dfrac{m^2 l^2 \cos z_3 \sin z_3}{hN - m^2 l^2 \cos^2 z_3} z_4^2 - \dfrac{mlF}{hN - m^2 l^2 \cos^2 z_3} \cos z_3 \\
y_1 = z_1 \\
y_2 = z_3
\end{cases}
\tag{II.35}
$$

Lorsque la grandeur de commande du modèle est la tension de commande V_c du moteur, il suffit de substituer dans le modèle (II.35) la force F par son expression (Equation (II.33)) on obtient

$$
\begin{cases}
\dot{z}_1 = z_2 \\
\dot{z}_2 = \dfrac{-bN}{hN - m^2 l^2 \cos^2 z_3} z_2 - \dfrac{m^2 l^2 g}{hN - m^2 l^2 \cos^2 z_3} \cos z_3 \sin z_3 \\
\quad + \dfrac{mld \cos z_3}{hN - m^2 l^2 \cos^2 z_3} z_4 + \dfrac{mlN \sin z_3}{hN - m^2 l^2 \cos^2 z_3} z_4^2 \\
\quad + \dfrac{N}{hN - m^2 l^2 \cos^2 z_3} \dfrac{K_m}{R_a r} V_c - \dfrac{N}{hN - m^2 l^2 \cos^2 z_3} \dfrac{K_m K_b}{R_a r^2} z_2 \\
\dot{z}_3 = z_4 \\
\dot{z}_4 = \dfrac{mgl}{N} \sin z_3 - \dfrac{d}{N} z_4 + \dfrac{mlb \cos z_3}{hN - m^2 l^2 \cos^2 z_3} z_2 + \dfrac{m^3 l^3 g \cos^2 z_3 \sin z_3}{N\left(hN - m^2 l^2 \cos^2 z_3\right)} \\
\quad - \dfrac{m^2 l^2 d \cos^2 z_3}{N\left(hN - m^2 l^2 \cos^2 z_3\right)} z_4 - \dfrac{m^2 l^2 \cos z_3 \sin z_3}{hN - m^2 l^2 \cos^2 z_3} z_4^2 - \dfrac{ml \cos z_3}{hN - m^2 l^2 \cos^2 z_3} \dfrac{K_m}{R_a r} V_c \\
\quad + \dfrac{ml \cos z_3}{hN - m^2 l^2 \cos^2 z_3} \dfrac{K_m K_b}{R_a r^2} z_2 \\
y_1 = z_1 \\
y_2 = z_3
\end{cases}
\tag{II.36}
$$

II .6. Conclusion

Ce chapitre a été consacré au développement du modèle de l'ensemble pendule chariot-moteur, on a utilisé pour ce faire le formalisme d'Euler-Lagrange.

Le modèle obtenu a permis de mettre en évidence les forces non linéarités existantes dans le pendule.

Chapitre III :

Commande du pendule inversé

III.1. Introduction

Un des aspects qualitatifs les plus importants des systèmes dynamiques est leur comportement asymptotique, c'est-à-dire le comportement des solutions en régime permanent, ce concept étant directement lié à la stabilité [6]. Dans cette mémoire, on s'intéresse aussi à la commande du pendule inversé qui consiste en deux phases : linéaire et non linéaire.

On présente dans ce chapitre deux méthodes de commande linéaire en utilisant la commande par retour d'état cette méthode nous exige une linéarisation de système au tour de point de fonctionnement. La deuxième est une commande non linéaire par la méthode de BACKSTEEPING.

III.2. Commande linéaire

III.2.1 Linéarisation du modèle autour du point d'équilibre instable ($\boldsymbol{\theta = 0}$)

Le modèle du pendule inversé est trop complexe et non linéaire, et comme l'objectif de la commande dans le système pendule inversé est d'asservir la position du chariot x et l'angle $\boldsymbol{\theta}$ à zéro (position d'équilibre instable), alors une linéarisation autour de cette dernière a été établie [20]. Pour des petites variations de $\boldsymbol{\theta}$ autour du point d'équilibre $\boldsymbol{\theta_0}$ on a :

$$\begin{cases} \theta = \theta_0 + \varepsilon \\ \dot{\theta} = \dot{\varepsilon} \end{cases} \tag{III.1}$$

On considère que tous les termes d'ordre supérieurs sont nuls

$$\dot{\varepsilon}^2 \approx 0$$

Le développement en série de Taylor du premier ordre d'une fonction de θ est donné par :

$$f(\theta) \approx f(\theta_0) + \varepsilon \left.\frac{df}{d\theta}\right|_{\theta_0} \tag{III.2}$$

Si on se limite aux petites variations de θ autour du point de fonctionnement $\theta_0 = 0$ correspondant à la position haute du pendule et on utilisant le développement limité du premier ordre (équations (III.1) et (III.2)) on obtient les approximations suivantes:

$$\begin{cases} \cos\theta \approx \cos 0 + \theta[-\sin 0] = 1 \\ \sin\theta \approx \sin 0 + \theta[\cos 0] = \theta \\ \dot{\theta}^2 = 0 \end{cases} \quad \text{(III.3)}$$

On substituant les linéarisations (III.3) dans le système d'équation ((II.36), on trouve le système d'équation linéarisé du système pendule-chariot-moteur suivant :

$$\begin{cases} \dot{z}_1 = z_2 \\ \dot{z}_2 = \left(\dfrac{-bN}{hN - m^2l^2} - \dfrac{N}{hN - m^2l^2}\dfrac{K_m K_b}{r^2}\dfrac{1}{R_a}\right) z_2 - \dfrac{m^2 l^2 g}{hN - m^2 l^2} z_3 + \dfrac{mld}{hN - m^2 l^2} z_4 \\ \qquad + \dfrac{N}{hN - m^2 l^2}\dfrac{K_m}{R_a r} V_c . \\ \dot{z}_3 = z_4 . \\ \dot{z}_4 = \left(\dfrac{mlb}{hN - m^2 l^2} + \dfrac{ml}{hN - m^2 l^2}\dfrac{K_m K_b}{r^2}\dfrac{1}{R_a}\right) z_2 + \left(\dfrac{mgl}{N} + \dfrac{m^3 l^3 g}{N\left(hN - m^2 l^2\right)}\right) z_3 - \dfrac{d}{N} z_4 \\ \qquad - \dfrac{m^2 l^2 d}{N\left(hN - m^2 l^2\right)} z_4 - \dfrac{ml}{hN - m^2 l^2}\dfrac{K_m}{R_a r} V_c . \\ y_1 = z_1 \\ y_2 = x_3 \end{cases} \quad \text{(III.4)}$$

Le modèle d'état (III.4) peut être mis sous la forme : $\begin{cases} \dot{z} = Az + Bu \\ Y = Cz \end{cases}$

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & \dfrac{-bN}{hN - m^2 l^2} & \dfrac{-gm^2 l^2}{hN - m^2 l^2} & \dfrac{mld}{hN - m^2 l^2} \\ 0 & 0 & 0 & 1 \\ 0 & \dfrac{mlb}{hN - m^2 l^2} + \dfrac{ml}{hN - m^2 l^2}\dfrac{K_m K_b}{R_a r^2} & \dfrac{mgl}{N} + \dfrac{m^3 l^3 g}{N\left(hN - m^2 l^2\right)} & \dfrac{-d}{N} + \dfrac{-m^2 l^2 d}{N\left(hN - m^2 l^2\right)} \end{bmatrix} \quad \text{(III.5)}$$

$$B = \begin{bmatrix} 0 & \dfrac{k_m N}{\left(hN - m^2 l^2\right) rR_a} & 0 & \dfrac{-ml}{hN - m^2 l^2}\dfrac{K_m}{R_a r} \end{bmatrix}^T , \quad C = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

III.2.2. Introduction à la régulation par retour d'état

Penchons-nous maintenant sur la régulation linéaire du pendule in- versé. Nous utiliserons pour ce faire une régulation optimale par retour d'état. Nous introduirons les caractéristiques générales d'une telle régulation et nous verrons par la suite différentes façons de l'appliquer [7].

Contrairement à la réalité, nous supposerons tout d'abord que l'entièreté de l'état du système est mesuré, car la connaissance de celui-ci

Est nécessaire pour ce type de régulation. Nous laissons donc pour l'instant de côté la problématique de l'élaboration d'un estimateur d'état et l'étude des conséquences de son insertion.

III.2.3. Mise en équations :

Considérons le système linéaire et permanent

$$\dot{x}(t) = A\, x(t) + B\, u(t) \qquad \text{(III.6)}$$

Où $x(t)$ est le vecteur des grandeurs d'état et $u(t)$ celui des grandeurs régnantes. Nous souhaitons par ailleurs maintenir un état d'équilibre qui est celui pour lequel le vecteur x est identiquement nul.

Par retour d'état, nous entendons que nous rechercherons une loi de réglage s'exprimant par la relation

$$u(t) = -K\, x(t) \qquad \text{(III.7)}$$

Le système (III.6) réglé par une telle loi est donc régi par les équations différentielles homogènes

$$\dot{x}(t) = (A - BK)\, x(t) \qquad \text{(III.8)}$$

Dont on peut démontrer que la stabilité est assurée si et seulement si les valeurs propres de la matrice $(A-BK)$ sont à partie réelle négative.

En effet, la résolution de l'équation vectorielle (III.8) nous donne, dans le cas où les valeurs propres de $(A-BK)$ sont distinctes, l'expression

$$\mathbf{x}(t) = \sum_{i=1}^{n} e^{\lambda_i t} m_i l_i \mathbf{x}_0 \qquad \text{(III.9)}$$

Où les mi et les li sont respectivement les vecteurs propres à droite et à gauche de la matrice (A − BK) associés à la valeur propre λi. En considérant L une matrice modale de (A − BK), les mi sont donc les colonnes de L et les li les lignes de L−1. On constate bien que x(t) ne reste borné que pour des λi à partie réelle négative.

III.2.4 Critères d'optimisation pour le régulateur

La synthèse du régulateur se fait en minimisant les écarts quadratiques de l'angle et de la position longitudinale ainsi que l'énergie mise en jeu pour déplacer l'ensemble chariot-balancier. Ceci nous conduit à définir des coefficients de pondération pour chacune des positions et vitesses et pour le signal de commande. La fonction à minimiser est alors la suivante.

$$J(Q,R) = \int_0^\infty (X^t(t)QX(t) + R.u^2(t))\,dt \qquad \text{(III.10)}$$

Avec :

$$Q = \begin{pmatrix} Q_\varphi & 0 & 0 & 0 \\ 0 & Q_{\dot\varphi} & 0 & 0 \\ 0 & 0 & Q_x & 0 \\ 0 & 0 & 0 & Q_{\dot x} \end{pmatrix} \qquad \text{(III.11)}$$

Et selon l'équitation différentiel de RECCATI :

$$(\dot p(t) + PA - PBR^{-1}B^TP + Q + A^TP) = C \qquad \text{(III.12)}$$

On fixer les coefficients de pondération, on a adopté les critères suivants :

1. l'amplitude du débattement angulaire n'est pas très importante,

2. le déplacement du chariot sur le rail doit être limité,

3. on ne se préoccupe pas de limiter les vitesses atteintes,

4. le signal de commande $u(t)$ ne doit pas être trop grand.

Fort de ces considérations, on a choisi les pondérations suivantes :

$$Q = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 5 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \qquad R = 1$$

et $$U = -K.x \quad \text{(III.13)}$$

Tell que :

$$K = R^{-1}B^{T}P \quad \text{(III.14)}$$

III.2.5 Gains du régulateur

L'utilisation de l'algorithme LQR (Linear-Quadratic-Regulator) pour la recherche des gains optimums est d'une grande efficacité. Une fois les coefficients de pondération *Q* et *R* fixés, on obtient les gains du régulateur d'état :

$K_{\varphi} = -4.98\ [V/rad]$

$K_{\dot{\varphi}} = -0.78\ [V/(rad/sec)]$

$K_{x} = +2.24\ [V/m]$

$K_{\dot{x}} = +1.41\ [m/sec]$

III.2.6 Simulations et programmations de la commande linaire

III.2.6.1 Le code Matlab pour obtenir les gains de K par placement des pôles.

```
clear all
clc
A=[0 1 0 0 ; 40.4 -0.217 0 -1.54 ; 0 0 0 1 ; 0.959 -0.005 0 -0.411]
B=[0 ; 50 ; 0 ; 13.3]
C=[1 0 0 0 ; 0 0 1 0]
D=[0]
step(A,B,C,D)
pA=poly(A)
pAH=poly([-1 -2 -3 -4])
KH=[pAH(4)-pA(4) pAH(3)-pA(3) pAH(2)-pA(2) pAH(1)-pA(1)]
p4=B
p3=[A+pA(3)*eye(4)]*B
p2=[A^2+pA(3)*A+pA(2)*eye(4)]*B
p1=[A^3+pA(3)*A^2+pA(2)*A+pA(1)*eye(4)]*B
p=[p1 p2 p3 p4]
pinvr=inv(p)
Kc=KH*pinvr
N=-inv(C*inv(A-B*Kc)*B)
```

III.2.6.2 Le code Matlab pour le système linaire totale :

```
function [xp1,xp2,xp3,xp4] = pendule_lin(x1,x2,x3,x4,u)
m=0.1;
g=9,81;
l=0.23;
M=0.9;
ph2=g*tan(x3)+(m*l*(sin(x3))*x4^2/M+m*sin(x3)*sin(x3));
phi4=((m*x4^2*sin(x3)*cos(x3))/(m*(sin(x3))^2+M));
```

xp1= x2;
xp2= 40.4*x1 -0.217*x2 -1.54*x4 +50*u ;
xp3= x4;
xp4= 0.959*x1 -0.005*x2 -0.411*x4 +13.3*u ;

III.2.6.3 Le schéma de système sous Simulink :

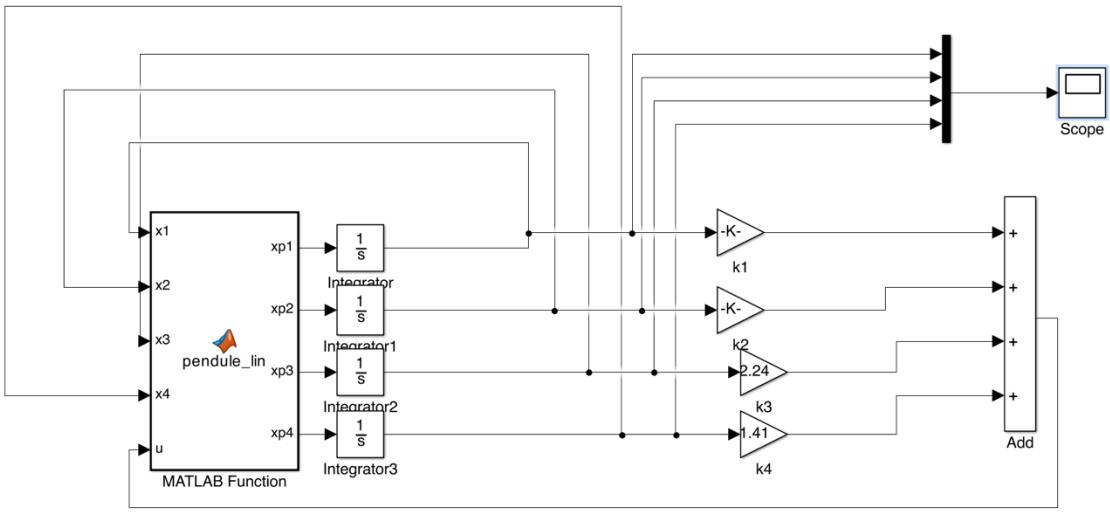


Figure III. 1 schéma bloc sous Simulink

III.2.6.4 Le résultat de simulation de cette commande :

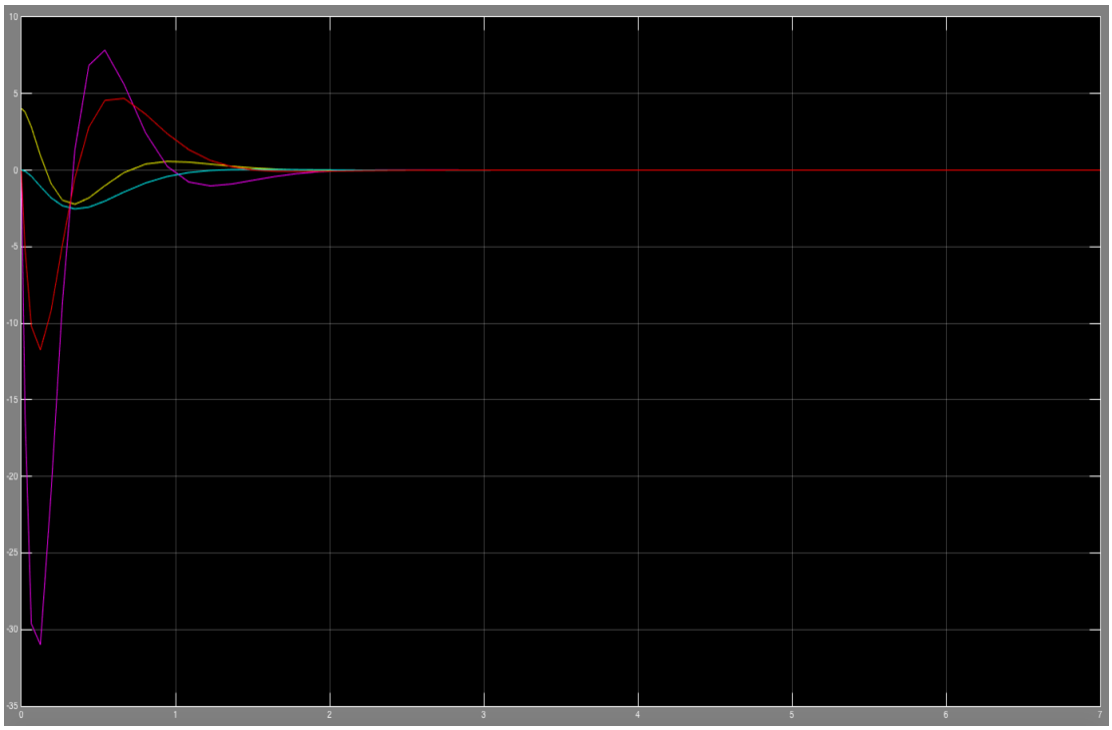


Figure III. 2 Résultat et simulation

III.3. Commande non linaire (*Backstepping*)

La théorie de commande non linéaire a été le sujet de développements intenses pendant les deux dernières décennies. Les outils développés dans ce secteur sont appliqués à l'étape de la conception et à l'exécution des unités de commande dans les systèmes non linéaires. Comparativement aux anciens outils, actuellement ces derniers sont devenus plus structurés et mieux développés. L'un des outils les plus connus est la théorie de *Backstepping*. Cette méthode donne un outil pour la conception récursive de la loi de commande basée sur la théorie de Lyapunov [8].

La méthodologie du *Backstepping* peut se définir comme une façon d'organiser un système en plusieurs sous-systèmes en cascade. L'exploitation de la méthodologie de conception sur un plan général aboutit à la mise en place d'une loi de commande par rétroaction associée systématiquement à une fonction de Lyapunov ou l'équivalent [8].

La théorie de Lyapunov a été pendant longtemps un outil important dans les systèmes linéaires aussi bien que dans la commande non linéaire. Cependant, son utilisation dans la commande non linéaire a été entravée par les difficultés pour trouver une fonction de Lyapunov pour un système donné. L'invention des outils constructifs pour la conception de commande non linéaire basée sur la théorie de Lyapunov, comme le *Backstepping*, a été donc reçue à bras ouverts par la communauté de commande. Ici, une loi de commande stabilisant le système est dérivée avec une fonction de Lyapunov pour prouver la stabilité [8].

Les propriétés importantes de la stabilité globale ou de chaque sous-système s'obtiennent en définissant une fonction stabilisatrice à chaque étape de représentation du modèle en sous- systèmes en cascade. Un des avantages que procure la méthode du *Backstepping* est celui de garder les propriétés du système initial dans la loi de commande obtenue. Ceci constitue en quelque sorte la particularité du *Backstepping* par rapport à d'autres méthodes. La passivité est liée à d'autres notions très importantes comme la stabilité, la détectabilité et l'optimalité. Toutes ces notions sont nécessaires pour la synthèse des lois de commande. Pour cette raison, l'analyse débute par l'étude de la méthode de *Backstepping* pour mener à son application à des systèmes de type *strict-feedback* [8].

III.3.1. Théorie de Lyapunov

La condition de base sur un système commandé est qu'il doit converger vers un état que nous déterminons. Formalisons cette condition en termes de propriétés de l'équilibre désiré [8].

III.3.2. Backstepping

La simplicité de la conception des lois de commande scalaires de la section précédente se fonde sur la connaissance d'une fonction de Lyapunov. Mais comment trouvons-nous une telle fonction ? Pour une certaine classe des systèmes non linéaires qui montrent une structure triangulaire inférieure, le *Backstepping* répond à cette question d'une façon récursive. Donc, nous énoncerons d'abord le résultat principal et ensuite nous traiterons les problématiques associées par exemple, en fonction des systèmes qui peuvent être manipulés en utilisant le *Backstepping* [8].

III.3.3 Principe de la technique du *Backstepping*

La simplicité de la conception des lois de commande scalaires nous motive à les utiliser en tant que point de départ de conception récursive pour les systèmes évolués. La première construction de (clf) pour les systèmes non linéaires a été développée par Sepulchre.

III.3.4. Le modèle non linaire de pendule inversé :

Le système considéré est représenté sur figure III. 3 Son modèle peut être exprimé par l'utilisation des variables d'état suivantes $x, \dot{x}, \theta, \dot{\theta}$; où x est la position du chariot, θ est l'angle de la tige, F est la force agissant sur le chariot, m et M sont respectivement les masses [9].

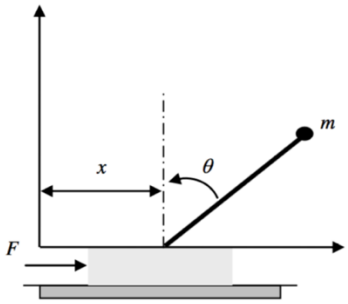


Figure III. 4 Plat de pendule inversé

Le modèle de pendule inversé en présence de perturbations, le modèle à l'étude prend en compte les variables d'état suivantes [9] :

$$x_1 = x, \quad x_2 = \dot{x}, \quad x_3 = q, \quad x_4 = \dot{q}$$

$$\begin{cases} \dot{x}_1 = x_2 + \eta_1(x, \omega, t) \\ \dot{x}_2 = \varphi_2 - \dfrac{1}{\cos x_3} u + \eta_2(x, \omega, t) \\ \dot{x}_3 = x_4 + \eta_3(x, \omega, t) \\ \dot{x}_4 = \varphi_4 + \dfrac{1}{\ell} u + \eta_4(x, \omega, t) \end{cases} \quad \text{(III.15)}$$

$$\varphi_2 = g\ tg\ x_3 + \frac{ml(\sin x_3)x_4^2}{M + m.\sin 2\, x_3} \quad \text{(III.16)}$$

$$\varphi_4 = -\frac{mx_4^2 \sin x_3\ \cos x_3}{m.\sin 2\, x_3 + M} \quad \text{(III.17)}$$

g : c'est l'accélération gravitationnelle et ℓ : c'est le vecteur de paramètre inconnus

η_i : Sont toutes les perturbations (internes et externes) ω : est le vecteur de paramètres inconnus.

III.3.5. Commande Backstepping

Dans ce qui suit, les différentes étapes sont conçues pour déduire la loi de commande. Dans ce cas, les paramètres du système sont supposés connus.

Le but est de contrôler la position angulaire souhaitée. Ainsi, nous avons opté pour la première variable d'erreur :

Etape 1

$z_3 = x_3 - y_r$

Et pour déterminer la stabilité du système, la première fonction de Lyapunov est définie par :

$$\vartheta_3 = \frac{1}{2} z_3^2 \quad \text{(III.18)}$$

Son dérivé conduit à :

$$\dot{\vartheta}_3 = z_3 \dot{z}_3 = z_3(x_4 + \eta_3 - \dot{y}_r) \quad \text{(III.19)}$$

Cela vous donnera la fonction de stabilisation suivante :

$$\alpha = x_{4d} - \dot{y}_r = -c_3 z_3 - h_3 \quad \text{(III.20)}$$

Où η_3 est délimitée par une valeur positiveh_3, $|\eta_3| \leq h_3$

$$\dot{\vartheta}_3 = -c_3 z_3^2 + z_3 . z_4 + z_3(\eta_3 - h_3) \quad \text{(III.21)}$$

Etape 2

La deuxième variable d'erreur est choisie de la manière suivante :

$$z_4 = x_4 - \alpha - \dot{y}_4 \quad \text{(III.22)}$$

Et la deuxième fonction de Lyapunov peut être représentée par l'expression suivante :

$$\vartheta_4 = \vartheta_3 + \frac{1}{2} z_4^2 \quad \text{(III.23)}$$

En utilisant les équations (III.15, III.20, III.21 et III.23)) la dérivée de la fonction précédente conduit à :

$$\dot{\vartheta}_4 = -c_3 z_3^2 - c_4 z_4^2 + z_4 \left(z_3 + c_4 z_4 + \varphi_4 + \eta_4 + \frac{1}{\ell} . u + c_3 . (x_4 + \eta_3) - c_3 \dot{y}_r - \ddot{y}_r \right) + z_3(\eta_3 - h_3) \quad \text{(III.24)}$$

Enfin, la loi de commande est obtenue comme suit :

$$u = -\ell [z_3 + c_4 z_4 + \varphi_4 + h_4 + c_3(x_4 + h_3) - c_3 \dot{y}_r - \ddot{y}_r] \quad \text{(III.25)}$$

III.3.5.1. Partie de code Matlab

.
.
. (En cours…)
.
.
.
.
.
.
.
.
.

III.3.5.2. Le schéma de système sous Simulink

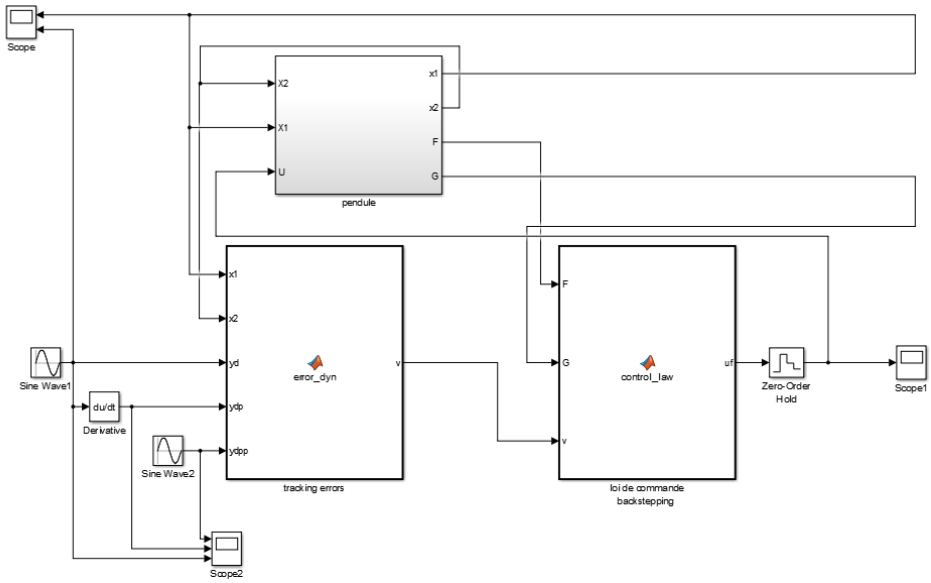


Figure III. 5 Schéma bloc sous Simulink

III.3.5.3. Le résultat de simulation de cette commande :

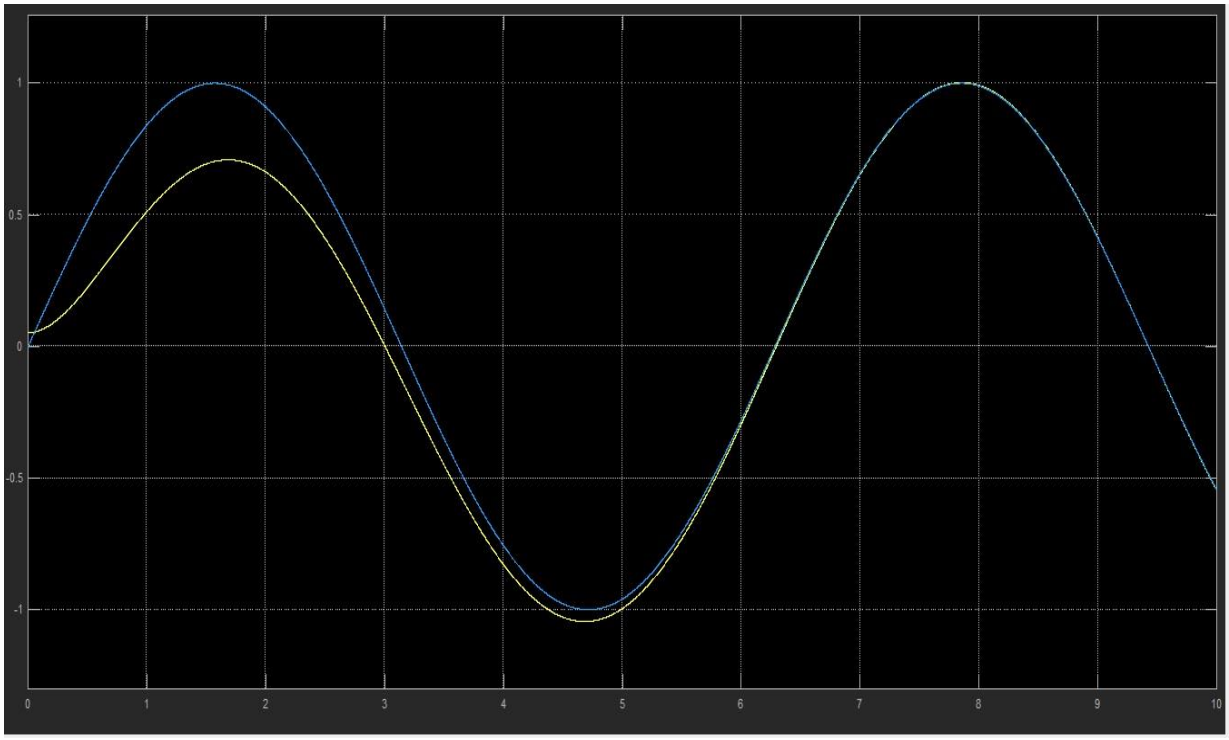


Figure III. 6 Résultat de simulation

III.4. Conclusion

Dans la première partie de ce chapitre nous avons présenté une commande linéaire et une linéarisation du système autour du point d'équilibre instable ainsi que des simulations ont également été établies. La simulation avec une impulsion d'entrée a permis de constater la non linéarité et l'instabilité du système autour de point d'équilibre, $(x, \dot{x}, \theta, \dot{\theta}) = (0,0,0,0)$. La complexité de contrôler des systèmes sous-actionnés tel que le pendule inversé, est en grande partie due au fait que l'entrée de commande u apparaît dans la partie actionnée du système ainsi que dans la partie non actionnée d'où l'intérêt d'utiliser une commande par retour d'état.

En suite dans la deuxième partie nous avons utilisé le modèle non linéaire pour appliquer une loi de commande par la méthode **backstepping**, et l'appliquer sur l'angle d'inclinaison de pendule inversé. Ainsi que simuler cette commande dans l'enivrement de Matlab Simulink.

Chapitre IV :

Conception et réalisations de la plateforme de prototypage

IV.1. Introduction

Les plateformes robotisées sont généralement bâties autour d'un électronique divisé en deux grandes parties distincte, assurant chacune des fonctionnalités bien précise, une première partie destinée à la commande de la partie en charge de la motricité du système et une partie dédier à l'intelligence du système, représenté son informatique, ces deux systèmes sont fortement communiquant et interagit en fonction de l'évolution du système.

La robotique mobile regroupe l'ensemble des techniques qui permettent de créer des systèmes dotés de moyens d'acquisition et de traitement d'information, ainsi que de capacités décisionnelles, pouvant accomplir, sous contrôle humain réduit, un certain nombre de tâches, parfois non connues à l'avance, et susceptibles d'évoluer dans le temps. Les robots mobiles fonctionnent en mode télé opéré, commandés à distance par un opérateur, ou en mode autonome, où le robot prend seul ses décisions face à un environnement donné. [2]

La robotique est un ensemble des disciplines (mécanique, électronique, automatique, informatique), lorsqu'en parle sur le côté informatique en parle sauvant sur l'architecture des logiciels qui est nécessaire soit dans la partie de simulation ou de programmation.

Dans ce chapitre, nous allons présenter la plate-forme de prototypage et les étapes qu'on a suivies durant notre réalisation ainsi qu'une description des différents softwares et hardwares utilisés pour le développement de notre carte de commande ; voire le langage de programmation, le logiciel de simulation et le logiciel d'implémentation.

IV.2. Architecture générale

La figure suivante présente l'architecture générale de notre plateforme de prototypage et ces différents organes (Figure *IV. 1*), la plateforme mobile est équiper par :

- Deux moteurs à courant contenue.
- Unité de traitement (Mbed).
- Centrale inertielle (IMU).
-Organe de puissance (L293D).
- Modem Radio (XBee).

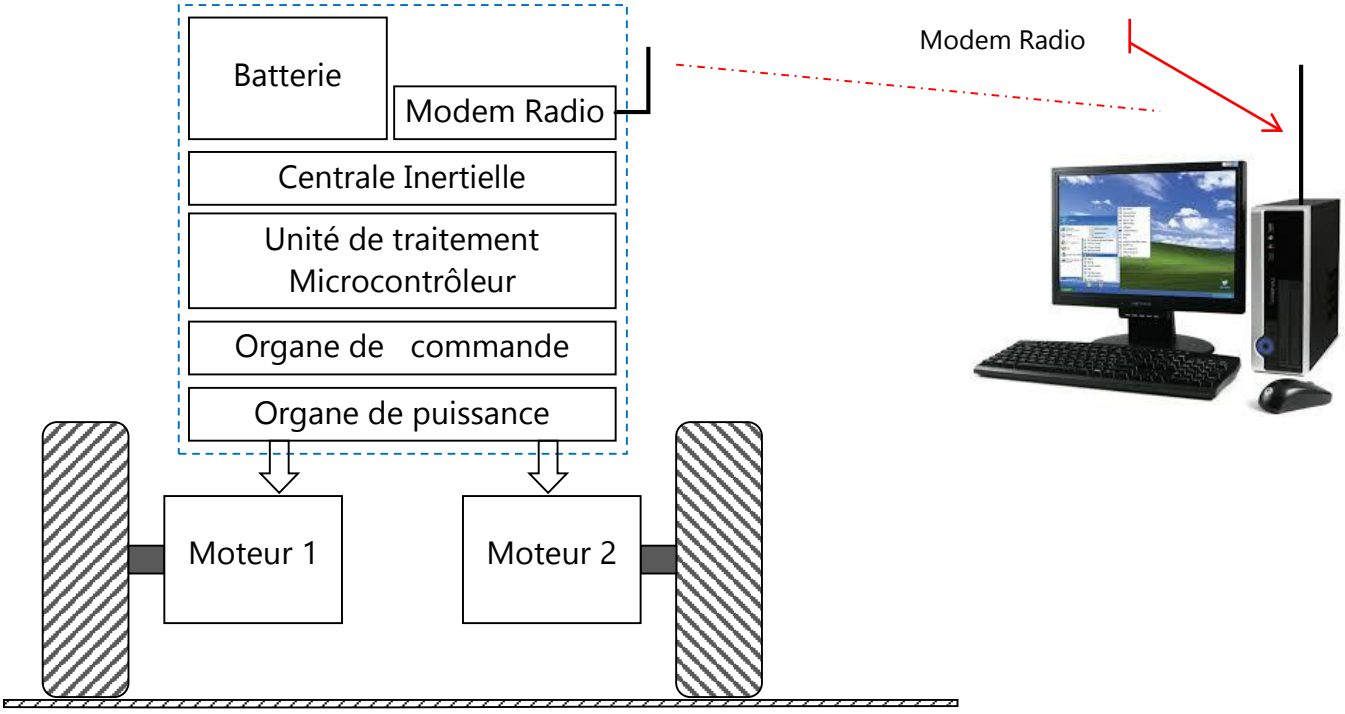


Figure IV. 2 Architecture générale de notre plateforme.

IV.3. Les microcontrôleurs

IV.3.1.Généralité sur les microcontrôleurs

Un microcontrôleur est un système informatique contenu dans un seul circuit intégré, Le microcontrôleur est composé de quatre parties : Un **microprocesseur** qui va prendre en charge la partie traitement des informations et envoyer des ordres. Il est lui-même composé d'une unité arithmétique et logique (UAL) et d'un bus de données. C'est donc lui qui va exécuter le programme embarqué dans le microcontrôleur. [10]

Une mémoire de données (RAM ou EEPROM) dans laquelle seront entreposées les données temporaires nécessaires aux calculs. C'est en fait la mémoire de travail qui est donc volatile.

Une mémoire programmable (ROM), qui va contenir les instructions du programme pilotant l'application à laquelle le microcontrôleur est dédié. Il s'agit ici d'une mémoire non volatile puisque le programme à exécuter est à priori toujours le même. Il existe différents types de mémoires programmables que l'on utilisera selon l'application. Notamment :

- OTPROM : programmable une seule fois mais ne coute pas très cher.
- UVPROM : on peut la réefacéer plusieurs fois grâce aux ultraviolets.
- EEPROM : on peut la réefacéer plusieurs fois de façon électrique comme les mémoires flash. [10]

Il ne consomme généralement qu'une fraction de Watt. Son jeu d'instructions est plus simple.

- la mémoire vivre est généralement très limitée : de quelques centaines de Bytes, à quelques dizaines de kB, selon les modèles. - les circuits d'entrée-sortie sont simplement des entrées logiques (pour lire une valeur binaire, par exemple un interrupteur) et des sorties logiques (capables de fournir quelques mA, par exemple pour commander une LED). Certains microcontrôleurs ont aussi des entrées analogiques : des convertisseurs analogiques-numériques (ADC= Analog to Digital Converter) et parfois des sorties analogiques : des convertisseurs numériques analogiques (DAC=Digital to Analog Converter). [10]

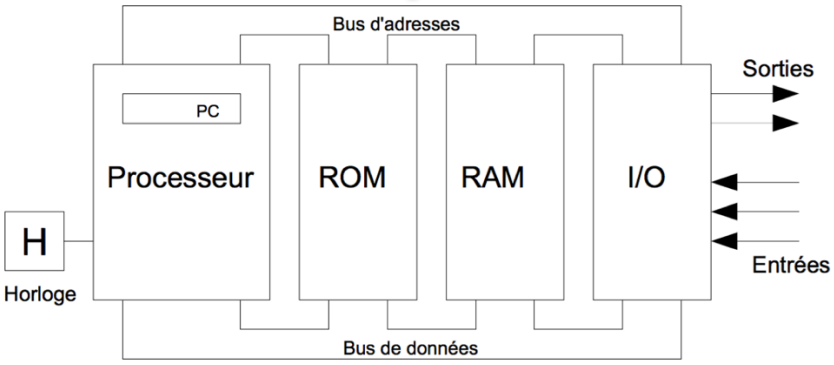


Figure IV. 3 Architecture simplifiée d'un système informatique. *[10]*

L'intérêt des microcontrôleurs est leur coût très faible, leur faible consommation de courant (quelques dizaines de mA) et leur taille très réduite (un seul circuit intégré, ayant de 6 à quelques centaines de pattes). Ils sont donc utilisés dans de très nombreuses applications. Bien que des microcontrôleurs existent depuis les années 1970, ils se sont développés de manière spectaculaire depuis quelques années. Alors qu'il était encore complexe et coûteux à mettre en œuvre un microcontrôleur au début des années 2000, cette tâche est maintenant beaucoup plus simple et ne nécessite que du matériel très peu coûteux, en plus d'un simple PC utilisé comme système de développement. Les microcontrôleurs sont donc devenus des composants électroniques incontournables, modifiant profondément la manière de concevoir les circuits électroniques. [10]

Plusieurs fabricants proposent des microcontrôleurs (Microchip, Atmel, Texas Instrument, Free Scale, NXP, ST-micro, Cypress, etc). Chaque fabricant propose souvent plusieurs familles de microcontrôleurs (PIC et dsPIC chez Microchip; AVR, AVR32 et ARM chez Atmel, etc). Chaque famille comporte souvent des dizaines de modèles, qui diffèrent principalement par les tailles mémoires (RAM et ROM) et le nombre de pattes d'entrée/sortie. A titre d'exemple, le processeur MSP430G2253, de Texas Instrument, contient :

• un processeur 16 bits, avec une centaine d'instructions, cadencé entre 1 et 16MHz, avec 16 registre de 16 bits.

• une mémoire morte de type flash (technologie similaire à celle des clés USB) de 16 kB. Cette mémoire peut être effacés et écrite à nouveau un très grand nombre de fois.

• une mémoire vive de 512 octets.

• 16 pattes d'entrée-sorties. Huit d'entre elles peuvent être connectées à un convertisseur analogique-numérique de 10 bits de résolution. Certaines de ces pattes ont également d'autres fonctions spécifiques (capture d'événements, utilisation d'un quartz, etc).

IV.3. 2.Critères de choix des microcontrôleurs

Il existe un très grand nombre de modèles de microcontrôleurs, et de nombreux fabricants proposent chacun plusieurs familles de microcontrôleurs, comptant chacune parfois des centaines de modèles.

On trouve des microcontrôleurs allant du petit circuit à 6 pattes coûtant une fraction d'€ jusqu'à des circuits comptant des centaines de pattes et coûtant des dizaines d'€.

Figure IV. 4 Les différents microcontrôleurs

- Le nombre de pattes d'entrées-sorties.
- Taille de la mémoire de programme (ROM)
- Taille de la mémoire vive (RAM)
- Consommation électrique (tension de fonctionnement, courant consommer)
- « La Puissance » du processeur, la fréquence de l'horloge n'est pas le seul paramètre pour la puissance de processeur mais il existe autre paramètres aussi :

1. La taille du bus d'adresses dépend du nombre de cellules mémoires
2. La taille du bus de données influence la capacité de traitement du système

- Le prix
- L'environnement de développement (matériel et logiciel, Le coût et la disponibilité)
- L'expérience.

IV.3. 3.Vue d'ensemble de la carte de développement mbed

La carte de développement mbed NXP LPC1768 est basée sur le processeur 32 bits ARM Cortex-M3, qui tourne à 96 MHz et comprends 512 Ko de mémoire flash et 64 Ko de RAM. [11]

La carte de développement mbed NXP LPC1768 possède des interfaces série (UART), SPI, I2C et un bus CAN. Elle supporte également d'autres périphériques, comme l'ethernet, un USB OTG, un convertisseur analogique-numérique 12 bits et un convertisseur numérique-analogique 10 bit, qui ne sont pas accessibles à des cartes moins puissantes.

La carte de développement mbed NXP LPC1768 possède des connecteurs mâles 0.1" qui lui permettent d'être installée facilement sur des plaques de prototypage rapide sans soudure.

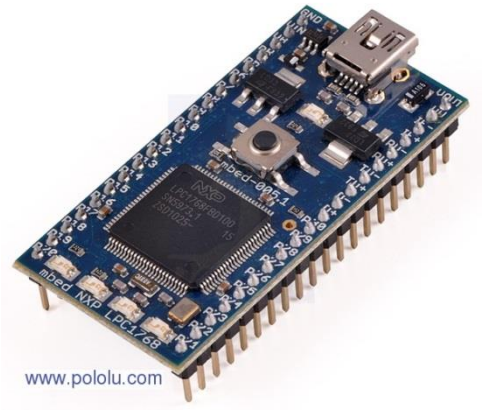


Figure IV. 5 mbed NXP LPC1768

L'un des avantages de la carte de développement Mbed NXP LPC1768 est son éditeur de code et son compilateur qui sont gratuits et en ligne, ces nombreuses librairies et ces exemples de codes. Le téléchargement des programmes dans la carte se fait par USB.

IV.3.4. Programmation sur la carte de développement mbed

Les outils compatibles avec la carte de développement mbed NXP LPC1768 permettent de s'affranchir de la plus grande partie du travail de bas niveau, nous développons notre code en utilisant des API intuitives qui nous permettent d'abstraire notre code et de concevoir nos applications sans nous préoccuper de l'implémentation exacte sur le microcontrôleur ou ses périphériques.

Le compilateur C/C++ en ligne de mbed est l'une des particularités les plus intéressantes de la carte de développement mbed NXP LPC1768. Il ne nécessite aucun

téléchargement ni installation et permet de créer et compiler les programmes de façon très simple et rapide. Il est utilisable avec Internet Explorer, Firefox, Safari et Chrome sur des ordinateurs sous Windows, Mac ou Linux, le fait que tout soit en ligne facilite également le partage du code. [11]

Le compilateur utilise le moteur de compilation ARM RealView qui produit un code optimisé, Mbed peut être utilisés avec d'autres outils professionnels comme Keil MDK.

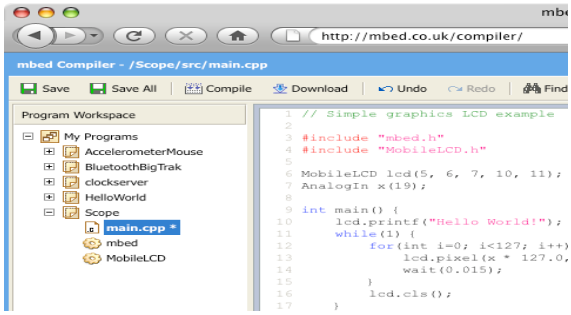


Figure IV. 6 KEIL MDK [11]

IV.3. 5 Caractéristiques techniques de mbed

- Peut être alimentée par USB ou par une alimentation externe entre 4.5 et 9 V (une tension jusqu'à 12 V ne devrait pas endommager la carte de développement mbed NXP LPC1768 mais peut causer un échauffement important du régulateur s'il doit alimenter des composants externes). [11]
- Forme de la carte : DIP de 40 broches avec un pas de 0.1"
- Programmation par glisser-déposé avec la carte de développement mbed NXP LPC176 représentée comme un disque USB
- Hardware Cortex-M3 :
- ARM 96 MHz avec 64 Ko de SRAM, 512 Ko de mémoire Flash
- Ethernet, USB OTG
- SPI, I2C, UART, CAN
- GPIO, PWM, ADC, DAC, DMA
- Environnement de programmation C/C++ basé sur le Web
- Utilisation du moteur de compilation ARM RealView
- Développement par API utilisant des librairies avec des interfaces intuitives
- Aide en ligne et communauté

- LEDs intégrées à la carte carte de développement mbed NXP LPC1768 pour du feedback visuel, port série via la connexion USB pour du debug en utilisant les logs.

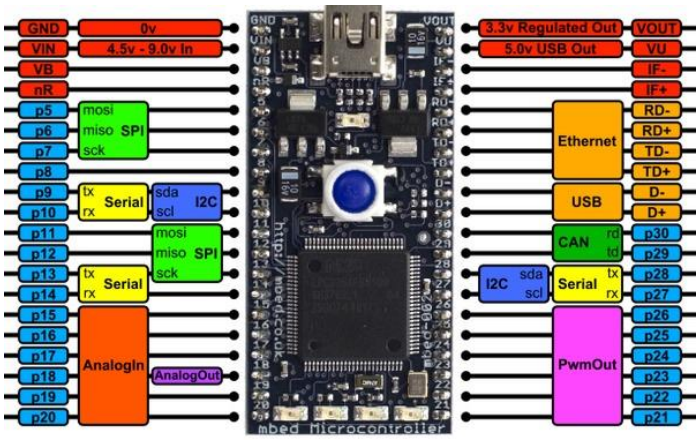


Figure IV. 7 Les Pines de Mebed [11]

IV.3.6. Architecture ARM

Les Cortex-M3 sont des processeurs conçus par ARM. Le processeur Cortex- M3 a été le premier de la génération de processeurs Cortex, publié par ARM en 2005 (produits de silicium libérés en 2006). Les processeurs Cortex-M3 utilisent une architecture 32 bits.[12]

IV.4. Capteurs et instrumentation de la plateforme

IV.4.1 Les systèmes de navigation inertielle (INS)

Une unité de mesure inertielle (IMU) est un instrument qui fournit des mesures de vitesses angulaires et d'accélérations. C'est donc un capteur proprioceptif, c'est à dire, mesure le comportement du véhicule par rapport à un repère relatif au véhicule. L'unité se compose d'un certain nombre de gyromètres et d'accéléromètres avec des blocs électroniques pour le traitement des signaux [13]. Une IMU à six axes comporte trois accéléromètres dans un arrangement orthogonal avec trois gyromètres (gyros) dans un arrangement orthogonal aussi. Certaines IMU disposent de 3 magnétomètres.

L'IMU avec l'unité de calcul additionnelle et les algorithmes d'estimation des angles d'orientations du mobile forment l'INS.

IV.4.2 MEMS (microsystème électromécanique)

MEMS est l'acronyme du terme anglo-saxon (Micro-Electro-Mechanical Systems), qui désigne la conception et la fabrication d'appareils électromécaniques miniatures en

exploitant la technologie des semi-conducteurs [14].

La technique des MEMS, a permis de se passer des dispositifs mécaniques volumineux et difficiles à mettre en œuvre (toupies et porteurs par exemple). Il est possible, désormais, d'intégrer ces appareils dans la majorité des plateformes (téléphone portable, manettes de jeux...). Les capteurs inertiels MEMS ont été introduits par l'industrie automobile dans sa recherche à améliorer les performances, réduire les coûts et accroître la fiabilité des véhicules.

La plupart des gyromètres MEMS sont basés sur le principe du transfert d'énergie entre deux modes de vibration dus à l'effet Coriolis expliqué dans le paragraphe suivant.

IV.4.2.1 Les gyromètres

Pour décrire le mouvement d'un corps dans l'espace 3D, le mouvement de rotation aussi bien que le mouvement de translation doit être mesuré. Les capteurs qui mesurent des angles ou des vitesses angulaires autour d'un axe dans un repère inertiel sont les gyromètres.

Les gyromètres fournissent le changement d'angle par rapport à une orientation initiale. Presque tous les gyromètres MEMS emploient des éléments mécaniques vibrants (masses) qui se servent de l'effet de Coriolis pour mesurer la vitesse angulaire [15]. Ils n'ont aucune pièce en rotation qui exige des roulements et par conséquent ils peuvent être miniaturisés sur silicium.

Les gyromètres produisent des mesures d'orientation relatives. Ils ne peuvent pas mesurer directement les angles mais ils rendent leur calcul possible par intégration :

$$\theta(t) = \int \omega\,(t).dt$$

Où θ est l'angle à déduire et ω la vitesse angulaire mesurée par le gyromètre.

IV.4.2.2. Les accéléromètres

Le capteur inertiel qui mesure l'accélération linéaire d'un corps le long de l'un de ses axes s'appelle un accéléromètre. Un accéléromètre mesure la force spécifique dans le repère inertiel, qui peut être employée pour estimer l'accélération du corps mobile.

Les accéléromètres dépendent de la deuxième loi de Newton dont la forme est :

$$F = ma$$

Ils mesurent l'accélération a d'une masse connue m (appelé la masse d'épreuve) par le biais d'une mesure de la force F que cette masse subit. Ce principe a permis l'apparition d'une large gamme d'accéléromètres utilisant plusieurs principes physiques pour effectuer la mesure. Les accéléromètres peuvent être classifiés comme étant mécanique ou à semi-conducteur.

IV.4.2.3. Les magnétomètres MEMS

Les INS n'utilisent que des gyromètres pour mesurer l'attitude et leurs erreurs sont compensées par l'estimateur de Kalman. Cependant, la dérive dans le gyroscope MEMS low-cost est importante et on a besoin d'un capteur complémentaire pour améliorer la fiabilité à long terme de la solution. Ce capteur doit être capable de mesurer les mêmes phénomènes angulaires mais dans une modalité physique différente. Le magnétomètre, mesurant le champ magnétique terrestre, est un capteur complémentaire pertinent pour améliorer la précision du calcul d'attitude.

Les unités de mesure inertielles contiennent souvent trois magnétomètres orthogonaux, en plus des gyromètres et des accéléromètres orthogonaux. Les magnétomètres mesurent l'intensité du champ magnétique (dans notre cas le champ terrestre), permettant de déduire une direction (donc un angle)

Les magnétomètres ne sont pas assez précis pour remplacer les gyromètres dans les INS. En particulier ils sont affectés par les perturbations locales du champ magnétique terrestre, provoqués par les objets ferreux ou magnétiques du voisinage. Les mesures peuvent cependant être fusionnées avec des données de gyromètres pour améliorer la précision de l'orientation calculée.

Les magnétomètres MEMS exploitent le changement des propriétés physiques (perméabilité, inductance, résistance électrique...etc.) d'un cœur métallique à cause du changement des conditions magnétiques extérieures. La conception de ces capteurs permet de transformer le changement des propriétés en une tension mesurable et proportionnelle au champ magnétique appliqué. Les capteurs magnétiques terrestres principaux sont :

- Magnétomètre à entrefer (sonde magnétométrique)
- Capteur magnéto-inductif
- Capteur magnéto-résistif

IV.4.3. Avantages des capteurs MEMS

Actuellement les capteurs MEMS sont dépassés en termes de précisions par les capteurs optiques. Toutefois ils sont les plus utilisés dans les applications courantes (car bas coût). Les principaux avantages des capteurs MEMS sont [16] :

- Dimensions réduites ;
- Faible poids ;
- Faible consommation de puissance ;
- Court temps de démarrage ;
- Production peu coûteuse (à grande échelle) ;
- Fiabilité élevée ;
- Entretien réduit ;
- Compatibles avec des opérations dans les environnements hostiles.

IV.4.4. Centrale inertielle UM6 (CHR-6DM AHRS)

Le CHR-6DM AHRS est un capteur d'orientation fournissant les sorties angulaires **yaw**, **pitch**, et **roll (IMU)** jusqu'à 300 Hz. Un filtre de Kalman étendu (EKF) combine les données d'accéléromètres embarqués, gyromètres et capteurs magnétiques pour produire des estimations angulaires **Yaw, Pitch,** et **Roll** [17]**.**

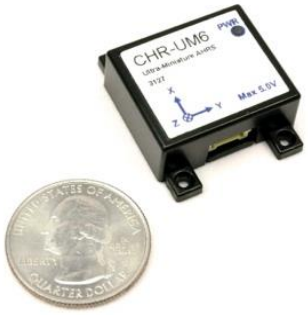


Figure IV. 8 CHR-6DM. [17]

La communication avec le CHR-6DM est réalisée sur une UART à 115200 Bauds. Les AHRS peuvent être configurés pour transmettre les mesures capteur brut en sus des estimations d'angle, et le taux de transmission peut être configuré par incréments de 1 Hz de 20 Hz à 300 Hz. Le CHR-6DM peut fonctionner en « mode silencieux », où les données sont transmises uniquement lorsque des demandes spécifiques sont reçues sur l'UART. Quel que soit le mode de transmission et le débit, les estimations des angles internes sont maintenues à plus de 500

Hz pour assurer une précision à long terme. La Figure IV. 9 présente l'IMU CHR-6DM de CH Robotics. Elle est considérée comme une IMU low-cost (moins de 2000€) et c'est celle-ci qui est utilisé dans notre projet.

Le CHR-6DM simplifie l'intégration en fournissant un certain nombre de routines de calibrage automatique, et d'étalonnage, Toutes les routines de calibrage sont déclenchées par l'envoi de commandes via l'interface série.

Un régulateur de 3.3V à bord simplifie également l'intégration. Avec la possible de tensions allant de + 3,3V à + 12V. Les broches IO ont une tolérante de 5V. Le CHR-6DM fournit également une tension de sortie de 3.3V qui peuvent fournir jusqu'à 400mA de courant pour alimenter d'autres périphériques.

Pour des applications personnalisées, le CHR-6DM dispose de six broches GPIO supplémentaires, qui peuvent être configurées comme entrées et sorties numériques dispose de deux UART et un bus SPI. Ces broches peuvent être utilisées pour le contrôle de moteur, l'intégration d'un GPS, l'intégration d'un capteur de pression, et la communication avec d'autres appareils numériques. Des outils de développement gratuits sont disponibles.

IV.4.4.1 Les paramètres maximaux absolus

Ce tableau nous montre les paramètres maximaux absolus du capteur CHR-6DM :

Symbol	Ratings	Maximum Value	Unit
Vdd	Supply voltage	-0.3 to +12V	V
Vin	Input voltage on any digital IO pin	-0.3 to 5.1 V	V
A	Acceleration	3000 *g* for .5 ms 10000 *g* for .1 ms	
Top	Operating temperature range	-40 to +85	°C
Tstg	Storage temperature range	-40 to +125	°C

Tableau IV.1 paramètres maximaux absolus [17]

IV.4.4.2 Caractéristiques électriques

Ce tableau nous montre les caractéristiques électriques du capteur CHR-6DM

Symbole	Paramètre	Condition de test	Min	Typ	Max	Unit
Vdd	**Tension d'alimentation**	**/**	**3.3**	**3.3-9**	**12**	**V**
Idd	**courant d'alimentation**	**/**	**50**	**52**	**58**	**mA**

Tableau IV. 2 caractéristique électrique [17]

IV.4.5. Les angles d'Euler

Les capteurs UM6 CH Robotics peuvent fournir des informations d'orientation à l'aide à la fois des angles d'Euler et des Quaternions. Par rapport au quaternions, Les angles d'Euler sont simples et intuitifs et ils se prêtent bien à l'analyse de contrôle. D'autre part, les angles d'Euler sont limités par un phénomène appelé « Lock Gimbal», Dans les applications où le capteur ne sera jamais fonctionner angles près de pas de +/- 90 degrés.[21]

Les angles d'Euler fournis un moyen pour représenter l'orientation 3D d'un objet en utilisant une combinaison de trois rotations autour d'axes différents.

Pour plus de commodité, nous utilisons plusieurs coordonnées repère pour décrire l'orientation du capteur, y compris le repère d'inertie, le repère véhicule 1," le repère véhicule 2," et le " repère de corps." Les axes de repère inertiels sont fixes terrestres et les axes du repère de corps sont alignés avec le capteur. Le véhicule 1 et un véhicule 2 sont des repères intermédiaires utilisés pour plus de commodité lors illustrant la séquence des opérations qui nous conduisent du cadre inertiel au cadre du corps du capteur. [21]

Il peut sembler inutilement compliqué à utiliser quatre coordonnées cadres différents pour décrire l'orientation du capteur, mais la motivation pour le faire deviendra clair que nous avançons.

IV.4.6 Le repère inertiel

Le " repère inertiel" est un ensemble de la Terre-fixe d'axes qui est utilisé comme une référence immobile. Les capteurs de CH Robotics utilisent un cadre inertiel aéronautique commun où les points de l'axe des x au nord, les points d'axe y est, et l'axe z pointe vers le bas, comme illustré ci-dessous. Nous appellerons ce cadre de référence d'un Nord-Est-Bas (NED). Notez que parce que l'axe z pointe vers le bas, l'altitude au-dessus du sol est en fait une quantité négative. [21]

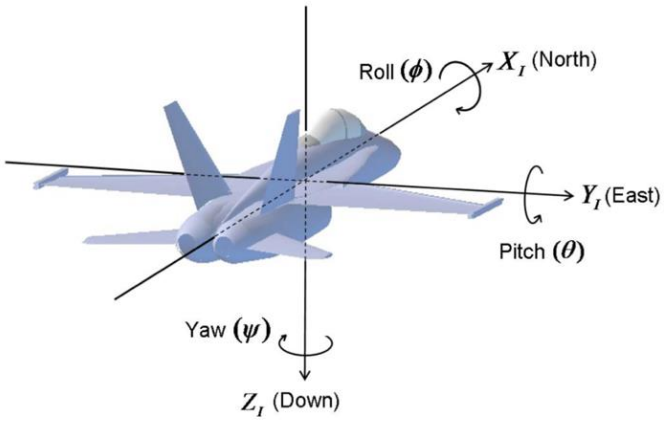


Figure IV. 10 Le repère inertiel (Roll , Pitch et Yaw). *[21]*

IV.4.7 La communication avec CHR-6DM

Le CHR-6DM communique sur un UART de niveau logique TTL à 115200 bauds, 8 bits de données, 1 bit d'arrêt et aucune parité.

IV.4.8 Les paquets RX de CHR-6DM (RX Packets)

(RX Packets) sont des paquets qui peuvent être reçus par le CHR-6DM. Chaque paquet reçu par les AHRS doit commencer par les trois octets séquence "**snp**" pour signaler le début d'un nouveau paquet. Le quatrième octet est indicateur de type de paquet (**PT**), qui identifie le paquet reçu. Le cinquième octet, **N** est le nombre d'octets contenus dans la partie de données du paquet. La section de données de **N** octets suivant immédiatement l'octet de longueur de données, et le paquet est finalement terminée par une somme de contrôle sur deux octets, qui contient la somme de tous les octets précédents du paquet. [17]

Si l'indicateur de type de paquet est reconnu, les AHRS vont transmettre un paquet de UNRECOGNIZED_COMMAND.

IV.4.8.1 La structure de paquets RX

Function	's'	'n'	'p'	PT	N	d_1	…	d_N	CHK	
Byte	1	2	3	4	5	6	…	N+5	N+6	N+7

Tableau IV.3 structure de paquets RX *[17]*

IV.4.8.2 Discrétion de paquet RX

1 - 3	Each received packet must begin with the three-byte (character) sequence "snp" to signal the beginning of a new packet.
4	**PT** specifies the packet type.
5	**N** specifies the number of data bytes to expect.
6 - (N+5)	$\mathbf{d_1}$ through $\mathbf{d_N}$ contain the **N** data bytes in the packet.
(N+6) - (N+7)	**CHK** is a two-byte checksum.

Tableau IV.4 discrétion de paquet RX *[17]*

Note : **Les paquets TX** sont des paquets qui peuvent être transmis par les AHRS. La structure des paquets transmis est exactement la même que la structure des paquets RX.

IV.5. Moteurs à courant continu

Le moteur à courant continu posse une caractéristique couple/vitesse de pente importante, ce qui permet de vaincre un couple résistant élevé, et d'absorber facilement les coups de charge, la vitesse du moteur s'adapte à sa charge. D'autre part, la miniaturisation recherchée par les concepteurs trouve dans le moteur à courant continu une solution idéale, puisqu'il présente un rendement élevé [22].

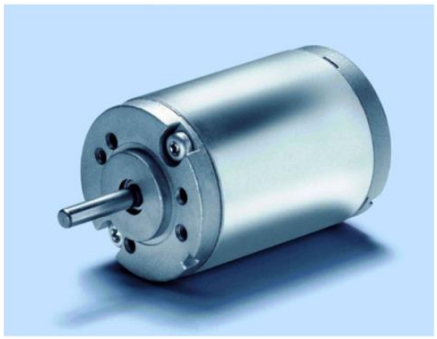


Figure IV. 11 Moteur à courant continu. *[22].*

IV.4.1 Généralités sur le moteur à courant continu

En comparaison aux autres technologies, **le moteur à courant continu** est une machine qui transforme l'énergie électrique en énergie mécanique, il est constitué de : l'inducteur (stator), l'induit (rotor), un aiment, un axe, le balai et le collecteur.

L'inducteur (parfois appelé champ) produit le flux magnétique dans la machine. Il est constitué d'un électro-aimant qui engendre la force magnétomotrice nécessaire à la production du flux. Dans les machines bipolaires (à deux pôles), deux bobines excitatrices sont portées par deux pièces polaires montées à l'intérieur d'une culasse. Les bobines

excitatrices sont alimentées en courant continu, et le courant qui les traverse porte le nom de courant d'excitation. Elles sont composées de plusieurs centaines de spires et portent un courant relativement faible. Les bobines sont bien isolées des pièces polaires afin de réduire les risques de court-circuit à la terre. Dans certains moteurs à courant continu, les bobines et pièces polaires sont remplacées par des aimants permanents.

L'induit est composé d'un ensemble de bobines identiques réparties uniformément autour d'un noyau cylindrique. Il est monté sur un arbre et tourne entre les pôles de l'inducteur. L'induit constitue donc un ensemble des conducteurs que coupe le flux magnétique. Les bobines sont disposées de telle façon que leurs deux côtes coupent respectivement le flux provenant d'un pôle nord et d'un pôle sud de l'inducteur.

Le collecteur est un ensemble cylindrique de lames de cuivre isolées les unes des autres par des feuilles de mica. Le collecteur est monté sur l'arbre de la machine, mais isolé. Les deux fils sortant de chaque bobine de l'induit sont successivement et symétriquement soudés aux lames du collecteur [22].

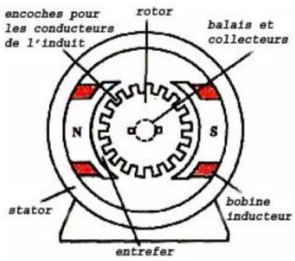


Figure IV. 12 Constitution du moteur à courant continu. *[22]*

Les balais sont fait en carbone, car ce matériau possède une bonne conductivité électrique et est assez doux pour ne pas user indûment le collecteur. Pour améliorer leur conductivité, on ajoute parfois au carbone une petite quantité de cuivre. La pression des balais sur le collecteur peut être réglée à une valeur appropriée grâce à des ressorts ajustables.

IV.4.2 Pilotage d'un moteur à courant continu par MLI

Dans cette partie on va expliquer comment effectue la commande d'un moteur à courant continu afin de contrôler son sens de rotation et sa vitesse par **MLI** (Modulation de Largeur d'Impulsion) ce qu'on appelle en anglais **PWM** (Pulse Width Modulation).

IV.4.2.1. La variation du sens de rotation d'un moteur à courant continu

Pour modifier le sens de rotation d'un moteur á courant continu, il suffit d'inverser l'alimentation à ses bornes (induit). La structure permettant de réaliser cette inversion est appelée« pont en H ».

IV.4.2.2. Le Principe de fonctionnement du pont en H

Le pont en H peut être symbolisé par une structure à quatre interrupteurs ouverts au repos (Figure *IV.13*). Dans ce cas, la tension aux bornes du moteur est nulle. Pour mettre en rotation le moteur il suffit de fermer un couple d'interrupteurs (voir Figure *IV. 14*) et de laisser les deux autres au repos. Le moteur est donc aliment´e avec la tension UM=VCC.

Pour modifier le sens de rotation, il suffit de permuter le couple d'interrupteurs fermés et ouverts (voir figure *IV. 15*). On aura donc UM= - VCC.

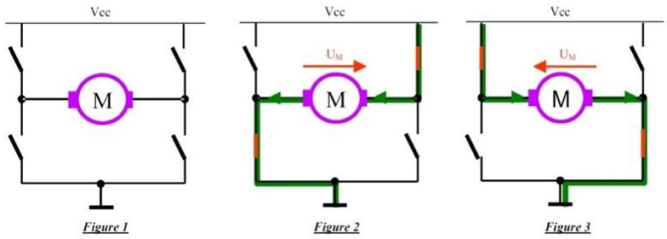


Figure IV. 16 Schéma de principe de fonctionnement du pont en H.

IV.4.2.3. La structure autour de transistors

Les transistors vont assurer la fonction des interrupteurs et fonctionneront en saturer/bloquer.

Ainsi en bloquant un couple de transistor et en saturant l'autre, on établit une tension UM aux bornes du moteur, dont on change le signe en modifiant les couples saturés/bloqués.

Les diodes placées entre émetteurs et collecteurs des transistors jouent le rôle de diode de roue libre (DRL). Elles permettent de protéger les transistors en devenant passantes lorsque ces derniers sont tous bloqués et qu'il faut évacuer le courant IM de l'inductance du moteur.

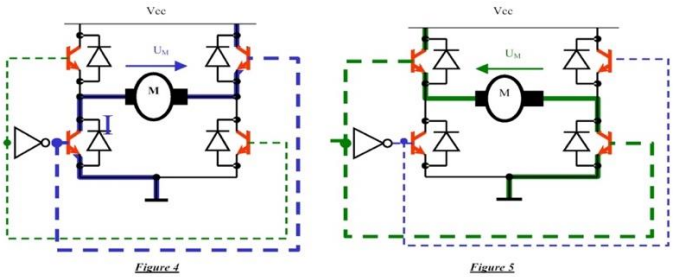


Figure IV. 17 Pont en H de transistors.

Il est possible d'activer la rotation d'un moteur à l'aide d'un relais ou d'un transistor. L'inconvénient de l'option transistor (ou relais) est qu'il n'est possible de facilement contrôler le sens de rotation du moteur. La solution réside dans l'utilisation d'un pont H. composant constitué de plusieurs transistors mais vendu préassemblé sous forme de circuit intégré. Pour cela on a utilisé dans notre carte de commande le circuit intégré **L293D**.

IV.5. Présentation du L293D

Le L293D est un double pont-H, ce qui signifie qu'il est possible de l'utiliser pour commander quatre moteurs distincts (dans un seul sens) grâce à ses 4 canaux. En raccordant les sorties de façon appropriées, il est possible de constituer deux pont- H. Il est ainsi possible de commander deux moteurs distincts, dans les deux sens et indépendamment l'un de l'autre.

"Le L293D un circuit intégré monolithique, a haut voltage, peut être utilisé pour des moteurs DC alimentés jusqu'à 36 Volts. Le circuit peut fournir un maximum 600mA par canal. Avec deux signaux de commande Input 1 et Input 2, il est possible d'inverser la direction du courant dans le pont-H et donc renverser le sens de rotation du moteur qui y est raccordé. En utilisant différentes combinaisons d'Input 1 et Input 2 il devient possible de démarrer, stopper ou inverser le courant. [22]

IV.5.1. Brochage de L293D

Ci-dessous la configuration des broches du L293D.

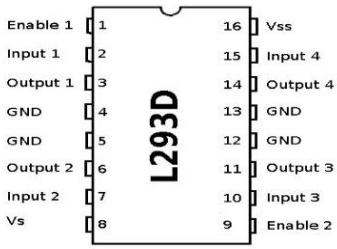


Figure IV. 18 Le Brochage de L293D Caractéristiques du L293D *[22]*

On peut résumer les Caractéristiques du L293 dans le tableau suivant

Caractéristique	Valeur	Caractéristique	Valeur
Nbr de pont-H	2	Courant Max	600mA (x2)
Courant de pointe Max	1200mA	VS Max	36V
VSS Max	7v	Perte de tension	1.3 à 1.4v (typical)

Tableau IV.5 : caractéristique du L293D. *[22]*

IV.5.2. La logique de commande de L293D

Broche	Nom	Description
1	ENABLE 1	L'activation/désactivation du premier Pont-H.
2	Input1	La commande du Pont-H Out1/Out2
3	Output 1	Raccorder à la charge (le moteur).
4	GND	La masse (GND) de l'alimentation de puissance VS
5	GND	Raccorder `a la masse (GND).
6	Output2	Raccorder `a la charge (le moteur).
7	Input2	La commande du Pont-H Out1/Out2
8	VS	Alimentation de puissance des moteurs.
9	Enable 2	Commande l'activation du second pont-H Out3/Out4.
10	Input 3	Avec Input 4 pour commander le pont-H Out3/Out4.
11	Output 3	La sortie 1 du second pont-H (Out3/Out4).
12	GND	Doit être raccorder `a la masse (GND).
13	GND	Doit être raccorder `a la masse (GND).
14	Output 3	La sortie 2 du second pont-H (Out3/Out4).
15	Input 4	Avec Input 3 pour commander le pont-H Out3/Out4.
16	VSS	Alimentation de la logique de commande (5V).

Tableau IV. 6. Logique de la commande de L293D. *[22]*

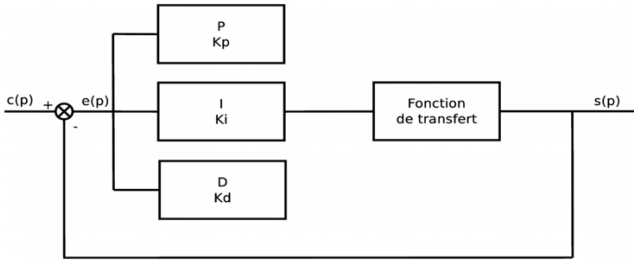
IV.6. Variation de vitesse du moteur à courant continu

Pour faire varier la vitesse d'un moteur on peut faire varier la tension d'alimentation à ses bornes mais dans ce cas une partie importante de l'énergie est consommée par le dispositif d'alimentation, on préfère l'alimenter de façon discontinue avec un hacheur et faire ainsi varier la tension moyenne à ses bornes. On parle alors de Modulation par Largeur d'Impulsions **(MLI)**, ou Pulse Wide Modulation **(PWM)**. La sortie PWM du microcontrôleur est utilisée pour commander **le L293D**. [22]

IV.7 Régulateur PID :

Régulateur PID (Proportionnel, Intégral, Dérivé) C'est un système d'auto régulation (boucle fermée), qui cherche à réduire l'erreur entre la consigne et la mesure [23].

E = Consigne – Mesure



Le régulateur PID sert à atteindre la valeur souhaitée pour une des variables du système (vitesse, position,...)

• Régulation : minimiser rapidement les perturbations

• Poursuite : s'adapter rapidement aux nouvelles consignes.

Ceci s'appelle l'asservissement.

IV.7.1 L'action Proportionnel :

L'erreur est multipliée par une constante Kp

$$u(t) = Kp \times e(t)$$
$$u(p) = Kp \times e(p)$$

Plus Kp est grand, plus la réponse est rapide mais on obtient une Erreur statique.

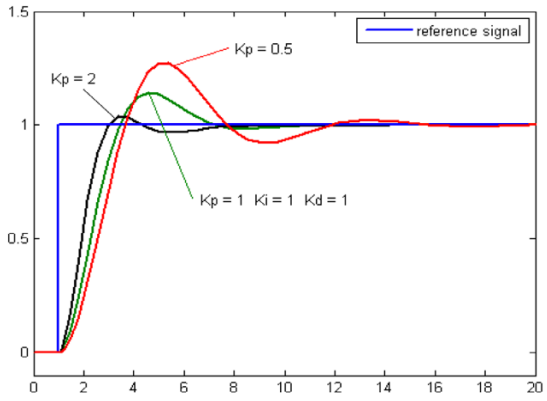


Figure IV. 19 :la réponse de l'action proportionnel [23]

IV.7.2 L'action Intégral

L'erreur est intégrée sur un intervalle de temps, puis multipliée par une constante Ki

$$u(t) = Ki \times \int_0^t e(t)\,dt$$
$$u(p) = Ki \times e(p)/p$$

Le corrige l'erreur statique, Plus Ki est élevé, plus l'erreur statique est corrigée.

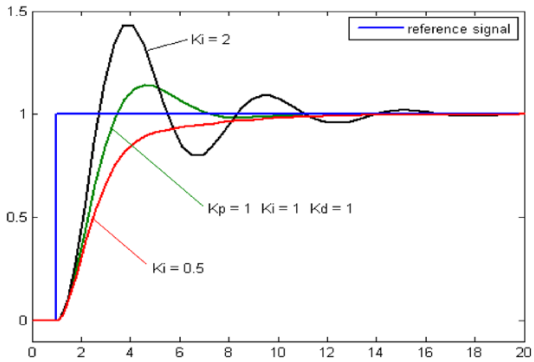


Figure IV. 20 : la réponse de l'action intégral [23]

IV.7.3 L'action Dérivé :

L'erreur est dérivée par rapport au temps, puis multipliée par une constante Kd

$$u(t) = Kd \times \frac{\partial e(t)}{\partial t}$$
$$u(p) = Kd \times e(p) \times p$$

Réduit le dépassement et le temps de stabilisation mais aussi notre système sera Sensible au bruit.

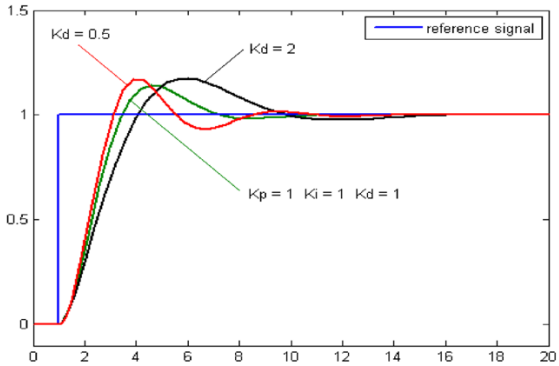


Figure IV. 21 : la réponse de l'action dérivé *[23].*

On résume alors :

$$u(t) = Kp \times e(t) + Ki \times \int_0^t e(t)dt + Kd \times \frac{\partial e(t)}{\partial t}$$
$$u(p) = e(p) \times (Kp + \frac{Ki}{p} + Kd \times p)$$

Coefficient	Temps de montée	Temps de stabilisation	Dépassement	Erreur Statique
Kp	Diminue	Augmente	Augmente	Diminue
Ki	Diminue	Augmente	Augmente	Annule
Kd	–	Diminue	Diminue	-

Tableau IV. 7 : *Tableau récapitulant l'influence d'un PID série sur le système qu'il corrige si l'on augmente séparément l'action proportionnelle (P), intégrale (I) ou dérivée (D). [23].*

IV.8 La Commination par Xbee :

Quand il s'agit de concevoir des systèmes embarqués, interactifs ou bien quand des objets doivent communiquer entre eux, plusieurs techniques de communication sont envisageables. Nous explorerons ici le protocole **Zigbee** qui permet de communiquer par ondes radio, c'est-à-dire sans fil. [24]

IV.8.1 Présentation du module XBee :

Les produits MaxStream XBee™ sont des modules de communication sans fil. La certification Zigbee se base sur le standard IEEE 802.15.4 qui définit les fonctionnalités et spécifications des réseaux sans fil (Wireless Personal Area Networks : WPANs). [24]

Les principales caractéristiques du XBee :

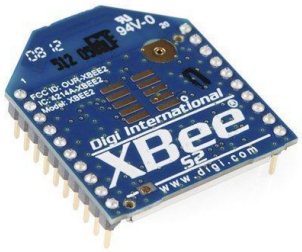
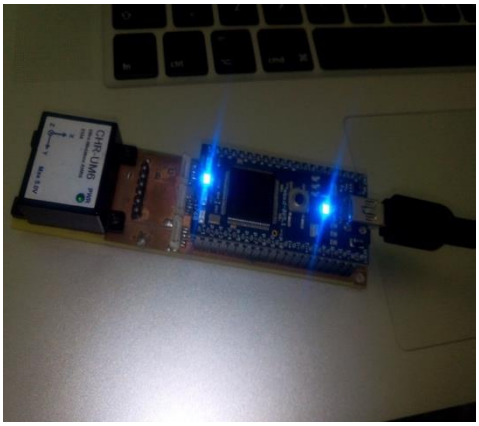


Figure IV. 22 Module XBee S2.[24]

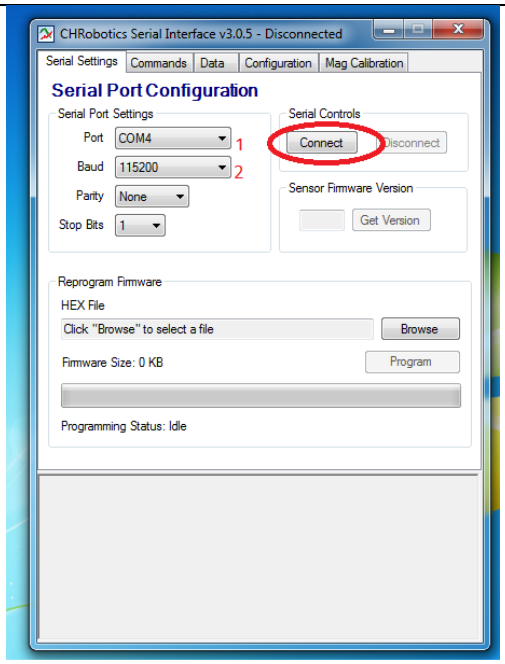
- Fréquence porteuse : 2.4Ghz
- Portées variées : assez faible pour les XBee 1 et 2 (10 - 100m), grande pour le XBee Pro (1000m)
- Faible débit : 250kbps
- Faible consommation : 3.3V @ 50mA
- Entrées/sorties : 6 10-bit ADC input pins, 8 digital IO pins
- Sécurité : communication fiable avec une clé de chiffrement de 128-bits
- Simplicité d'utilisation : communication via le port série
- Ensemble de commandes AT et API
- Flexibilité du réseau : sa capacité à faire face à un nœud hors service ou à intégrer de nouveaux nœuds rapidement
- Grand nombre de nœuds dans le réseau : 65000
- Topologies de réseaux variées : maillé, point à point, point à multipoint.

IV.9. Interface SERIAL CHAR configurations :

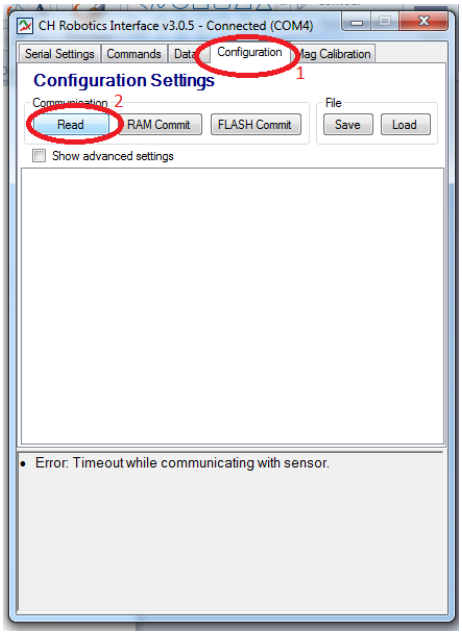
Le capteur UM6 besoin une configuration spécial qui nous permet d'utiliser leurs données avec l'IDE, pour cela Interface SERIAL CHAR est un logiciel qu'on peut faire cette configuration.



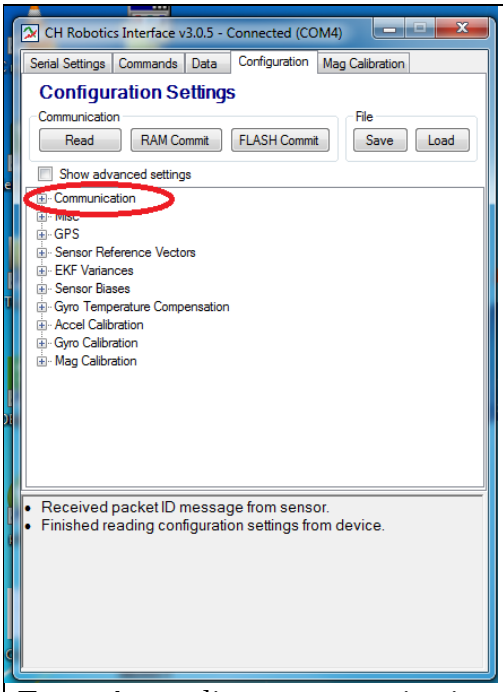
Etape 1 : Après qu'on connecte la carte mbed avec le capteur UM6 par la commination serial UART on connecte aussi la carte Mbed avec le PC via USB.



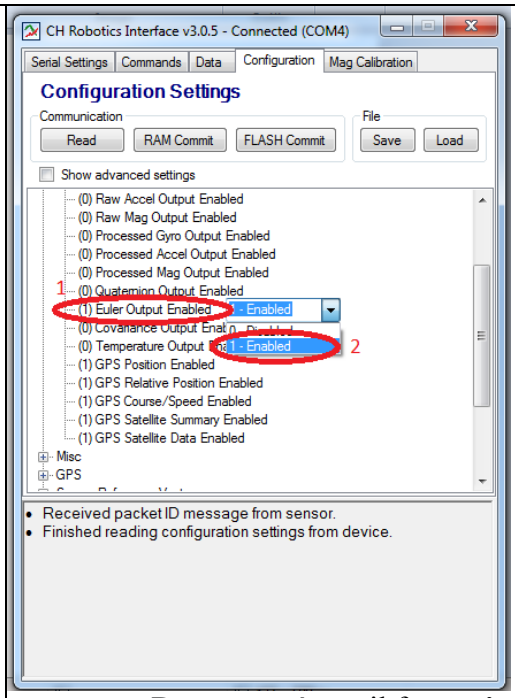
Etape 2 : On selecte le port qu'il est connecter avec le PC, avec une vitesse de transmission de 115200.



Etape 3 : On clique sur configuration, puis Read .

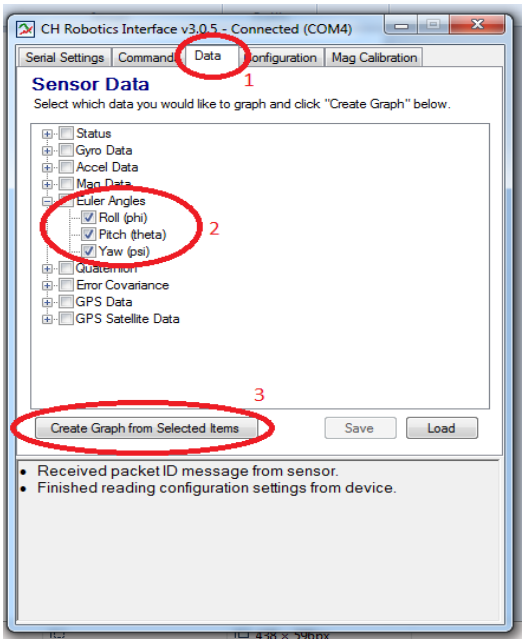


Etape 4 : on clique sur commination.



Etape 5 : Dans cette étape il faut qu'on active seulement les donner qu'on besoin dans notre projet et on désactive les autres. Alors on active les sorties des angles d'Euler en cliquant sur (Enabled)

Et (Disabled) sur les paramètres comme (Raw acc, Raw Gryo, Processed Acc, Processed Gyro).



Etape 6 : Ensuite on clique sur Data et Euler Angles, et on sélectionne (Roll, Pich et Yaw).
Puis on clique sur (Create Graph from Selected Itms).
On observe alors les variations en temps réel des angles d'Euler dans chaque axe.

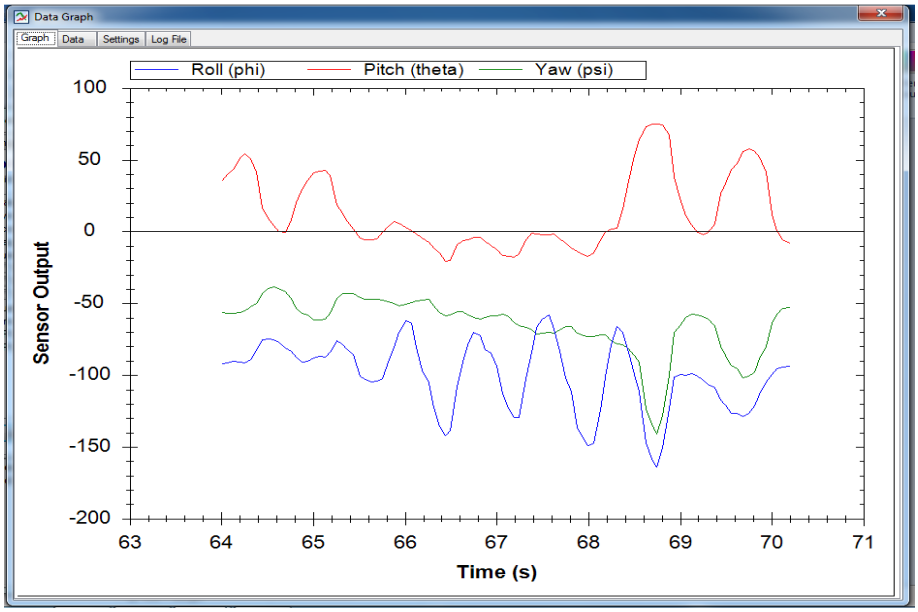


Figure IV. 23 : variation en temps réel des angles d'Euler.

Dans la *Figure IV. 23,* On observe les variations des angles d'Euler en temps réel lorsque on bouge le capteur UM6 dans les différentes régions sur les trois axes (X, Y, et Z).

IV.10. Schéma de circuit d'application

Ci-dessous le schématique de circuit de la plateforme avec les capteurs (Xbee , UMI) et la circuit de puissance :

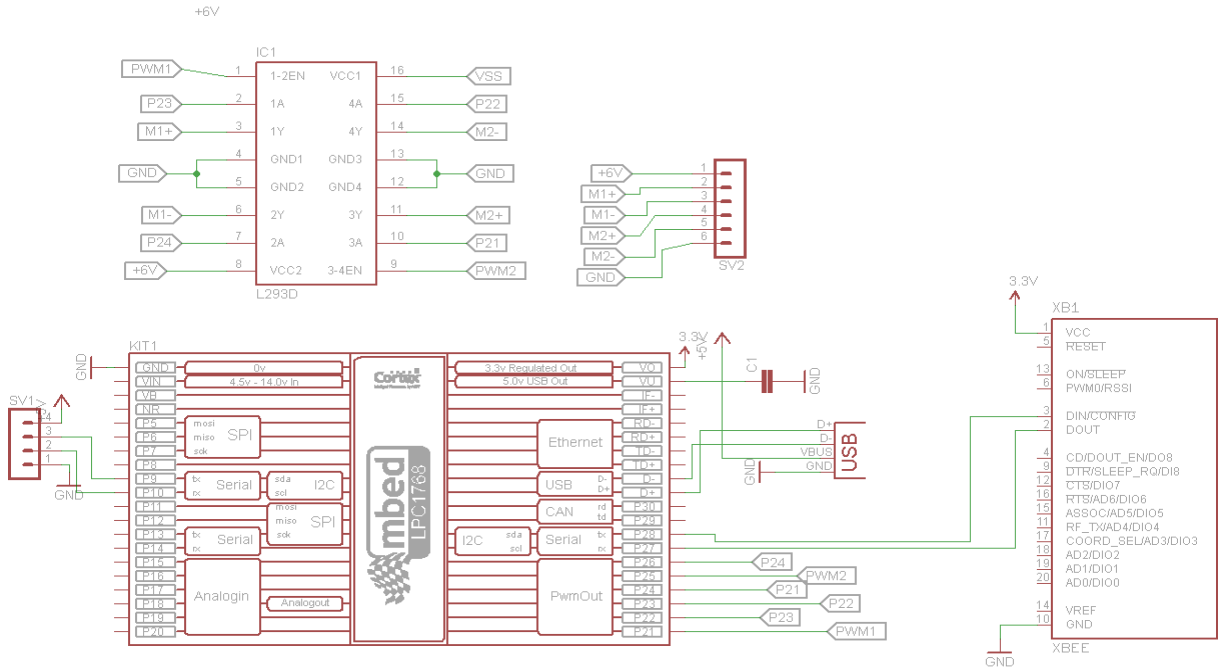


Figure IV. 24 Schématique de circuit.

Ci-dessous le bord de circuit (PCB) de la plateforme avec les capteurs (Xbee , UMI) et la circuit de puissance :

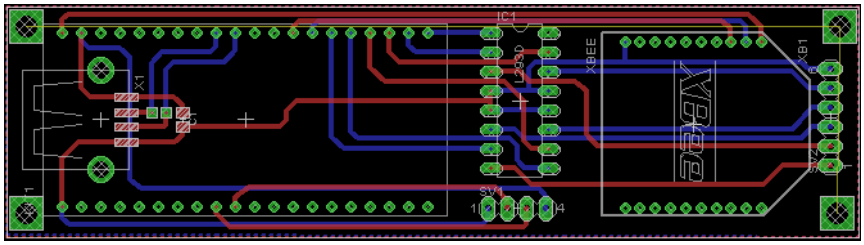


Figure IV. 23 Bord

IV.11. Plateforme mobile

Ci-dessous image de la plateforme mobile utilisé dans la réalisation :

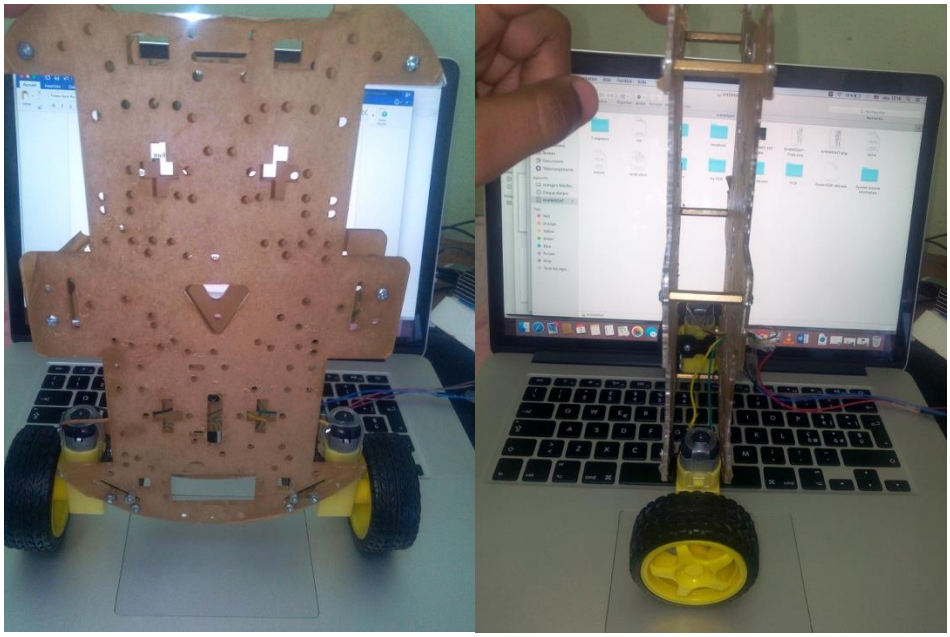


Figure IV. 26 La plateforme mobile.

IV .12. Organigrammes

IV .12.1. Organigramme par régulation TOR (ON/OFF)

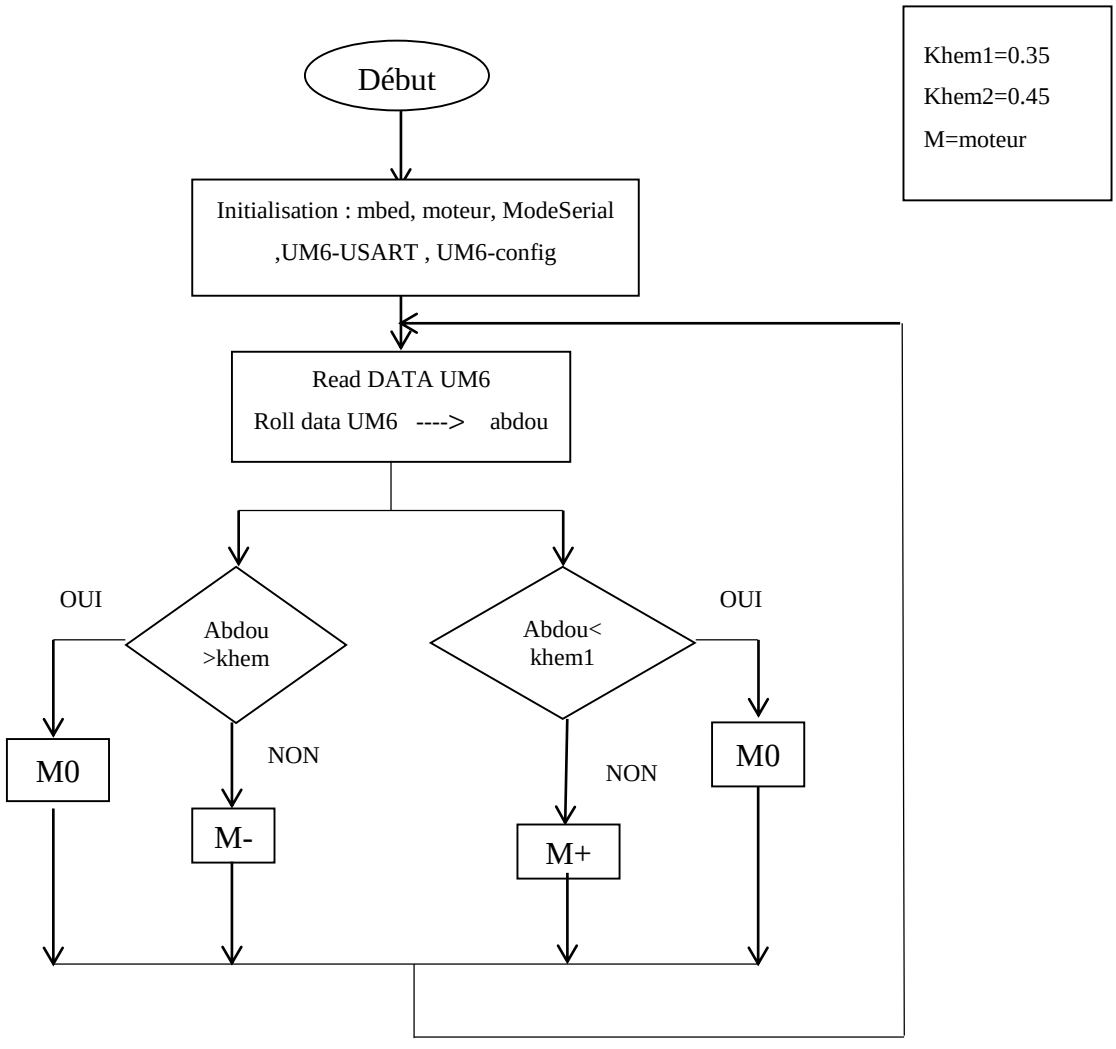
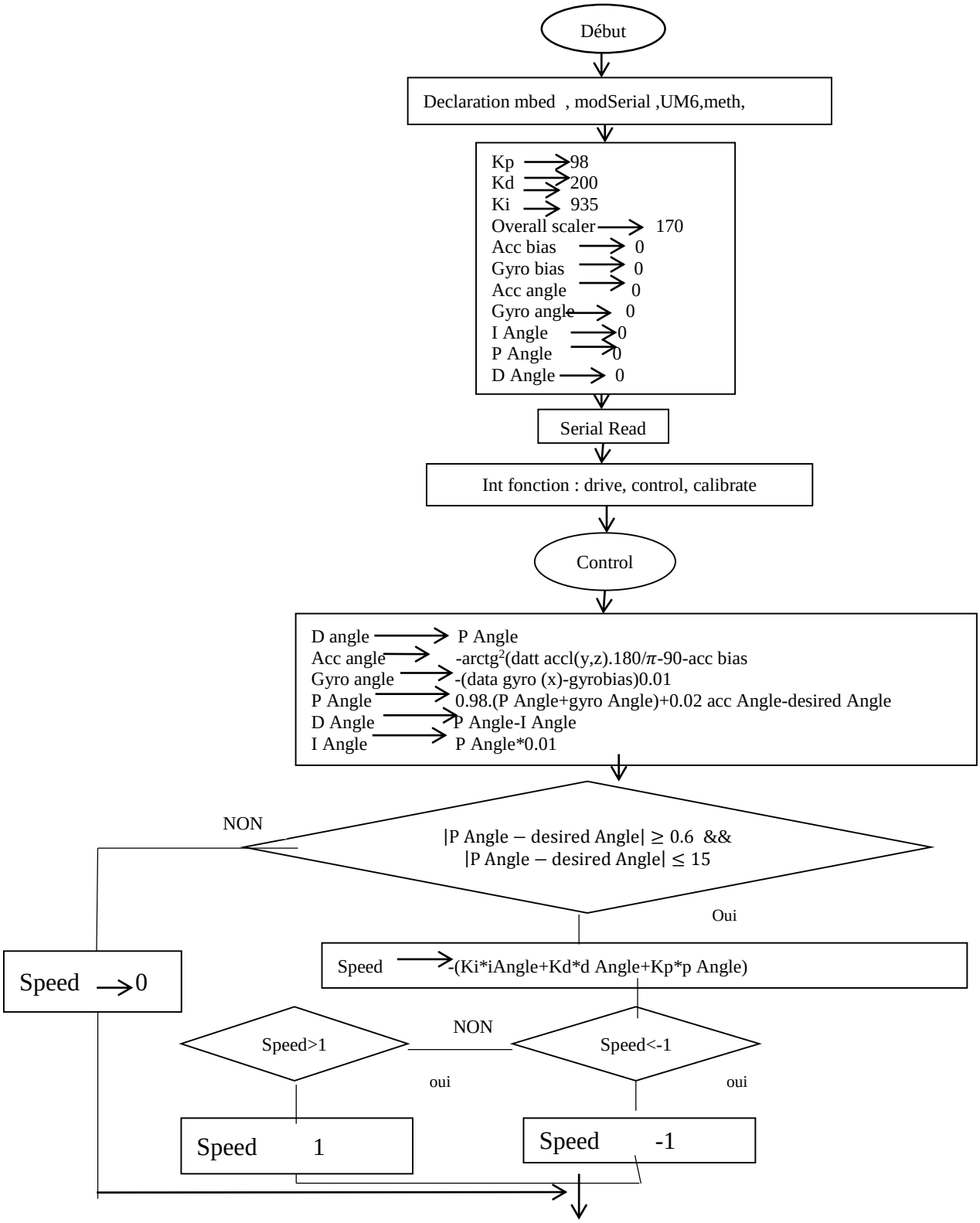


Figure IV. 27 :organigramme par régulation TOR

IV.12.2. Organigramme d'implémentation du contrôleur PID



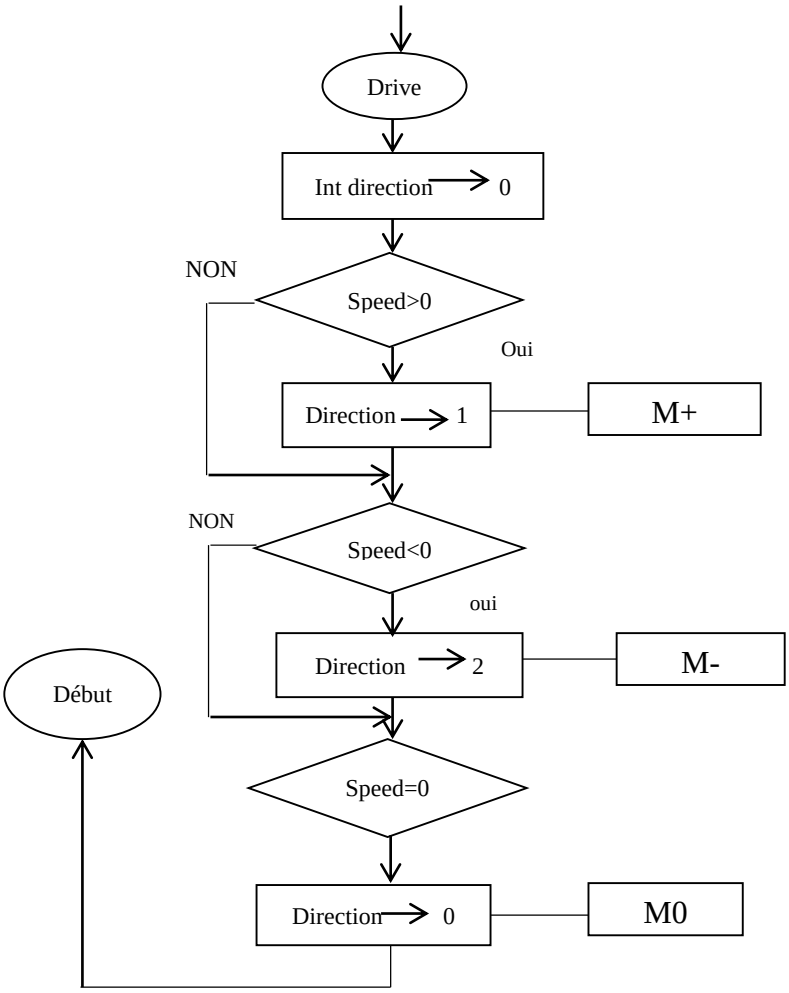


Figure IV.28 : organigramme par régulation PID

IV.13. L'interface Graphique pour le contrôleur PID

Dans le but d'obtenir les paramètres de PID pour une bonne régulation de notre robot nous avons développé une interface graphique à base de langage **Python**. (*Figure IV. 24)* de l'interface.

Dans cette interface nous essayions de varier les trois paramètres (Kp, Kd et Ki) par tâtonnement et les envoyer par le port Série de PC (USB ou Xbee) vers le robot en temps réel.

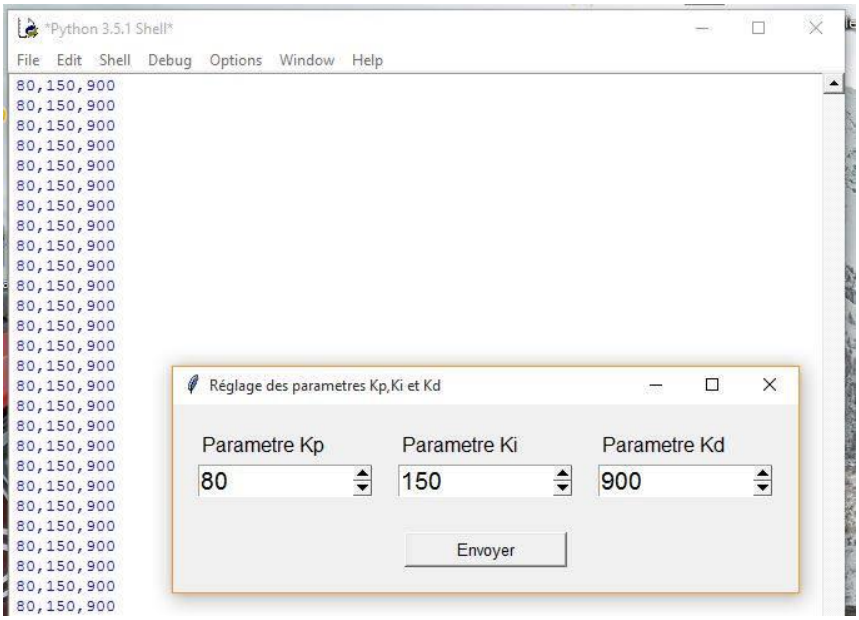


Figure IV. 30 L'interface de tâtonnement des paramètres du régulateur PID

Cette interface est associée avec une visualisation des variations des angles d'Euler (Roll, pitch et Yaw) en temps réel.

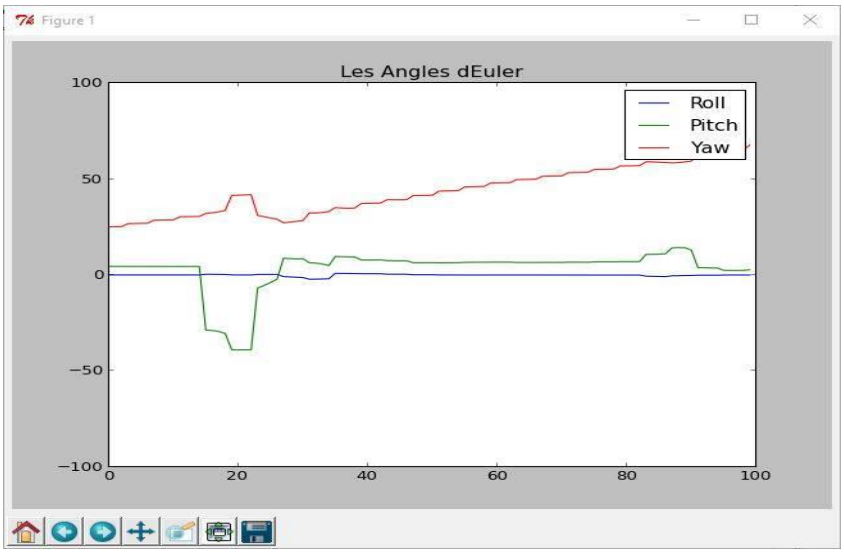


Figure IV. 31 Interface de visualisation des angles d'Euler

IV .16. Conclusion

Dans la première partie de ce chapitre nous avons présenté une conception générale de la plateforme de prototypage rapide avec la carte commande et de puissance de la plateforme mobile et les différents éléments constituant.

La conception matérielle est une étape déterminante dans le développement du projet, en fait, le choix des composants, leurs positionnements dépendent de l'application qu'on veut réaliser.

Dans la deuxième partie nous fait une étude dimensionnelle sur les capteurs et instrumentation utilisé dans la plateforme et aussi une généralité sur les moteurs à courant contenue et pour bien définir le régulateur PID et leurs paramètres. Ainsi que la communication par le module Xbee.

Finalement dans la troisième partie nous avons développé une Architecture logicielle. Cette dernière nous a permis de développer notre carte de commande et puissance en utilisant d'une manière logique un côté d'obtenir les bons paramètres de PID et d'autre coté est de visualiser les donner de robot sur le PC.

Conclusion générale

Dans ce présent travail qui s'inscrit dans le cadre de projet de fin d'études nous avons mis en valeur l'une des principales lignes de recherche actuelle pour la robotique mobile en environnement naturel et qui consiste en : « Une Plateforme de prototypage rapide pour la robotique mobile : Application à un robot auto-balancé ».

Conformément aux objectifs ciblés d'une part, et vus, les problèmes rencontrés, d'autre part, les résultats obtenus se résument en la réalisation de la commande électronique et son interfaçage à un PC, ainsi qu'une première série de tests a été effectuée et montrée le fonctionnement correct du système après une analyse critique et résolution des anomalies localisées.

Par ailleurs, nous signalons que le travail abordé englobe plusieurs domaines d'étude (la programmation, l'électronique, la gestion de l'énergie, la mécanique). Il nous a permis d'approfondir nos connaissances théoriques et d'améliorer d'autre niveau et aussi de toucher le coté pratique grâce a la réalisation et la gestion de projet.

On note que le dispositif de commande électronique est opérationnel dans sa première version. Il pourra servir d'un banc de tests au profit des futurs candidats au master. En perspective, nous recommandons la prise en compte des points suivants :

- Effectuer des améliorations en intégrant d'autres fonctionnalités de détection et le traitement et vision.
- Améliorer la technique de commande sans fil (pour rendre le contrôleur d'une commande par (Pc), et commande autonome).

Bibliographie

[1] K. J. Astrom et K. Furuta, “Swinging Up a Pendulum by Energy Control,” in 13th IFAC World Congress, San Antonio 1996.

[2] C. Boukazzoula, “Etude, conception et réalisation d’une partie de commande à distance d’une plateforme mobile à roues,” mémoire de master, UMBB, Boumerdes, 2015.

[3] M. Ziat, “Etude et réalisation d’un robot mobile,” projet de fin d’etude, ENP, Alger, 2004.

[4] D. Filliat, “Introduction,” in *Robotique Mobile*, D. Filliat, Fondation. Paris Tech, ENSTA Eds. UNIT : 2004, pp. 9–14.

[5] Iutenligne, “Principe de robot auto-balancé,”18-Jan-2013. [Online]. Available : http://public.iutenligne.net/etudes-et-realisations/nardi/Segway/Principe/index.html. [Accessed : 06-Jan-2016].

[6] C. Boussalem, “Implémentation de régulateurs fonctionnaire pour la stabilisation d’un pendule inversé, mémoire de magister, UMMTO, Tizi-ouzou, 2012.

[7] M. Kinnaert, *Théorie des régulations optimales*. Université Libre de Bruxelles (U.L.B.) : Service d’automatique et d’analyse des systèmes, Notes de cours, 2005.

[8] F. Eshbair, “ Modélisation et commande d’un système multi-moteur par la technique de commande BACKSTEPPING,” mémoire de la maitrise, UQTR, Québec, 2005.

[9] M. Mokhtari, “Adaptive Neural Network Control for Inverted Pendulum Using Backstepping with Uncertainties,” in *Advanced Science and Technology* ijast, SERSC. Vol.71.2014, pp. 1–14.

[10] Pierre-Yves Rochat, “Qu'est-ce qu'un microcontrôleur ?,” EPFL, École polytechnique fédérale de Lausanne , Nov. 2013.

[11] Génération robot, “La carte de développement mbed NXP LPC1768,”18-Jan-2013. [Online]. Available : http://www.generationrobots.com/fr/400992-carte-de-d%C3%A9veloppement-mbed-nxp-lpc1768.html. [Accessed : 12-Apr-2016].

[12] Joseph Jiu, *The definitive guide to ARM CORTEX-M3 AND CORTEX-M4 Processors*. Cambridge : Third, 2014.

[13] K. Jonghyuk "Autonomous navigation for airborne applications," PhD, Department of Aerospace, Mechanical and Mechatronic Engineering, The University of Sydney, Australia. 2004.

[14] M.S. GREWAL, et al., Global Positioning Systems, Inertial Navigation, and Integration. USA : WILEY-INTERSCIENCE, 2007.

[15] O. Woodman, "An introduction to inertial navigation," Computer Laboratory, University of Cambridge, United Kingdom 2007.

[16] D.H.Titterton and J. L. Weston, Strapdown inertial navigation technology. USA : The Institution of Electrical Engineers, 2004.

[17] CHR Robotic, *UM6 Ultra-Miniature Orientation Sensor Datasheet*. CHR Robotic : USA, Oct. 2013.

[18] M. Bugeja "Non linear swing-up and stabilizing control of an inverted pendulum system," Eurocon Ljubljana, Slovenia. 2003.

[19] K. Andrew "Standup and stabilization of the inverted pendulum. Submitted to the department of mechanical engineering in partial fulfillment of the requirements for the degree of Bachelor of Science," Massachusetts Institute of Technology. June 1999.

[20] O. Orban, " Automatisation d'un pendule inversé," mémoire de fin d'étude, Université Libre de Bruxelles, 2006.

[21] CHR Robotic AN-1006, *Understanding Quaternions*. CHR Robotic : USA, Oct. 2012.

[22] D. KOUMARI, "REGULATION EN CASCADE D'UN MOTEUR A COURANT CONTINU," Mémoire de fin d'étude, FHC, 2008.

[23] M. Camus, E. Deguine, D. Ross, *Régulation par PID,* TELECHOM ParisTech, 2010.

[24] Jérôme Abel, *Xbee Arduino.*2013.

ANNEX

1. Programmations :

1.1 Programmation par régulation TOR (ON/OFF) :

```
#include "mbed.h"
#include "Motor.h"
#include "MODSERIAL.h"     // MBED BUFFERED SERIAL

#include "UM6_usart.h"     // UM6 USART HEADER
#include "UM6_config.h"    // UM6 CONFIG HEADER

 char a ;

float khemissat1=0.35;      // limite 1 pour l'ange d'Euler
float khemissat2=0.45;     //   limite 2 pour l'ange d'Euler
MODSERIAL pc(USBTX, USBRX);    // PC SERIAL OVER USB PORT ON MBED
DigitalOut pc_activity(LED1);            // LED1 = PC SERIAL
DigitalOut uart_activity(LED2);        // LED2 = UM6 SERIAL
float abdou ;
void rxCallback(MODSERIAL_IRQ_INFO *q) {
   if (um6_uart.rxBufferGetCount() >=  MAX_PACKET_DATA) {
      uart_activity = !uart_activity;  // Lights LED when uart RxBuff has > 40 bytes
      Process_um6_packet();
   }
}

Motor m(p23, p15, p16); // pwm, fwd, rev
Motor n(p24, p17, p18 );

int main() {

      um6_uart.baud(115200);
       pc.baud(115200);  //
   um6_uart.attach(&rxCallback, MODSERIAL::RxIrq);

      // pc.attach(fpc);
//int Roll_Counter = 0;
   while (1) {

      //  Roll_Counter++;
        //if (Roll_Counter > 5000000) {
```

```
        abdou = data.Roll ;
       pc.printf("Roll %f deg/s, ",abdou); pc.printf("\n");

        if (abdou<khemissat1){m.speed(-0.95);  n.speed(-0.95);}
     if(abdou>khemissat2){m.speed(0.95);n.speed(0.95);}

        if(abdou>khemissat1 &&  abdou<khemissat2 ) {m.brake(BRAKE_HIGH);
n.brake(BRAKE_HIGH);   }

// pc_activity = !pc_activity;  // Lights LED when uart RxBuff has > 40 bytes
        // Roll_Counter = 0;
}
  }

  }
    }
```

1.2 Programmation par implémentation du contrôleur PID :

```
#include "mbed.h"
#include "MODSERIAL.h"     // MBED BUFFERED SERIAL
#include "math.h"
#include "UM6_usart.h"     // UM6 USART HEADER
#include "UM6_config.h"    // UM6 CONFIG HEADER

PwmOut motorSpeedLeft(p21);                    // PWM port to control speed of left
motor
PwmOut motorSpeedRight(p22);                    // PWM port to control speed of right
motor
DigitalOut motorDirLeft(p24);                  // Digital pin for direction of left motor
DigitalOut NmotorDirLeft(p23);                  // Usually inverse of motorDirLeft
DigitalOut motorDirRight(p26);                 // Digital pin for direction of right motor
DigitalOut NmotorDirRight(p25);                 // Usually inverse of motorDirRight
Ticker start;
//////////////////////////////////////////////////////////////////////
MODSERIAL pc(USBTX, USBRX);  // PC SERIAL OVER USB PORT ON MBED
//////////////////////////////////////////////
// Globals
float kp = 98;   //40                          // 145 Proportional gain kU 400-500
float kd = 200;   //2 was quite good                  // Derivative gain
float ki = 935;    //30                        // Integrative gain
float OVERALL_SCALAR = 170;                          // Overall scalar that speed is
divided by
float accBias = 0;                            // Accelerometer Bias
float gyroBias = 0;                           // Gyro Bias
float accAngle = 0;                           // Global to hold angle measured by
Accelerometer
```

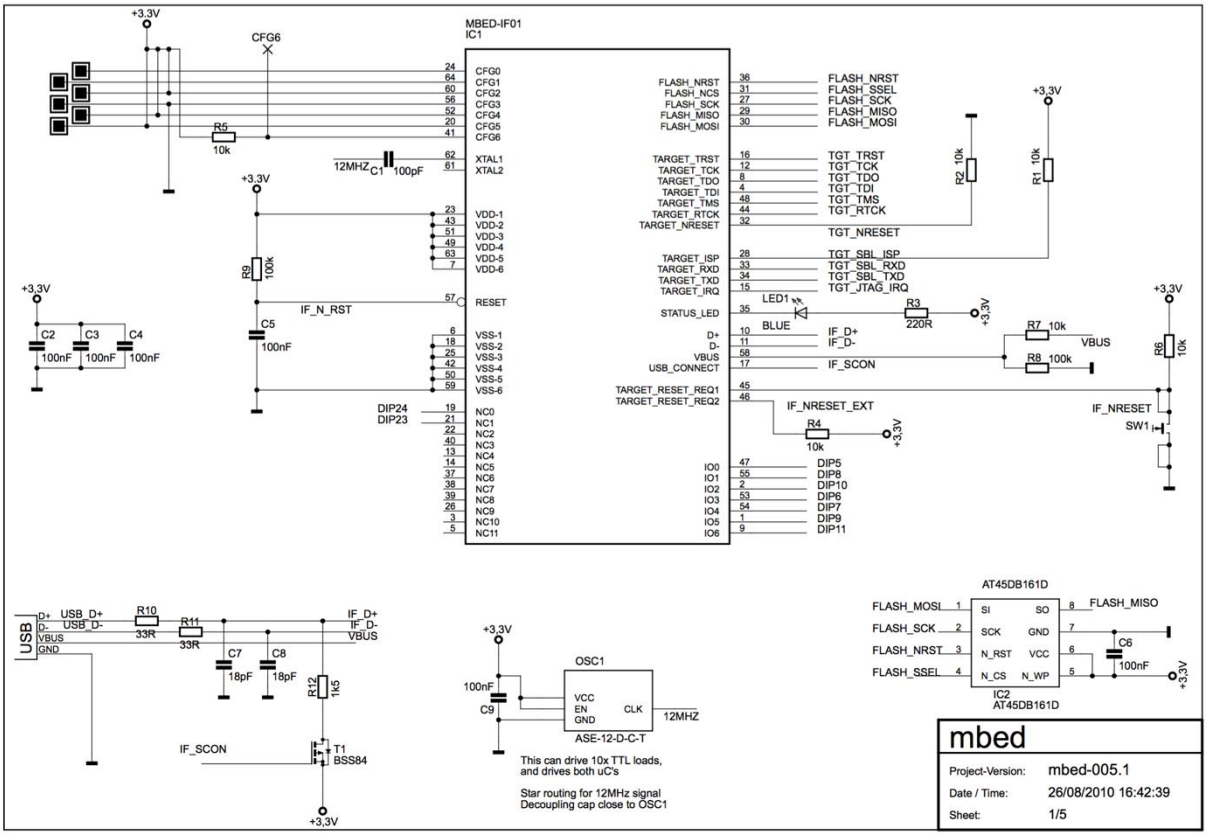
```
float gyroAngle = 0;                              // This variable holds the amount the angle
has changed
float speed = 0;                                 // Variable for motor speed
float iAngle = 0;                                 // Integral value of angle-error (sum of gyro-
angles)NOT EQUAL TO gyroAngle
float dAngle = 0;                                 // Derivative value for angle, angular velocity,
how fast angle is changing
float pAngle = 0;                                 // Proportional value for angle, current angle
(best measurement)
float desiredAngle=0;

//////////////////////////////////////////////////////////////////////

void rxCallback(MODSERIAL_IRQ_INFO *q) {
   if (um6_uart.rxBufferGetCount() >=  MAX_PACKET_DATA) {
      Process_um6_packet();
   }
}
// Function Prototypes
void drive(float);                                // Function declaration for driving the motors
void calibrate();                                 // Function to calibrate gyro and accelerometer
void control();                                   // Function implementing PID control
#ifdef DEBUG
void callback();                                  // Interrupt triggered function for serial
communication
#endif

int main() {
      um6_uart.baud(115200);
      pc.baud(115200);  // pc baud for UM6 to pc interface
      um6_uart.attach(&rxCallback, MODSERIAL::RxIrq);
     // pc.attach(&callback);                              // Attach interrupt to serial communication

       calibrate();                                // Calibrate gyro and accelerometer
   start.attach(&control, 0.01);                        // Looptime 10ms ca. 100Hz
   while(1) {                                     // Stay in this loop forever
#ifdef DEBUG
      pc.printf("%f\n",speed);                        // Print to USB the speed
#endif
   }
}
//////////////////////////////////////////////////////////
void control()
{
   dAngle=pAngle;// remember old p-value
                              // Read the Accelerometer
   accAngle=(-1)*atan2(data.Accel_Proc_Y,data.Accel_Proc_Z)*180/3.142-90-accBias;    //
Like this, 0 is upright, forward is negative, backward positive
  ////////////////////////
   //gyroAngle=data.Roll ;                    // This is deltaangle, how much angle has changed
```

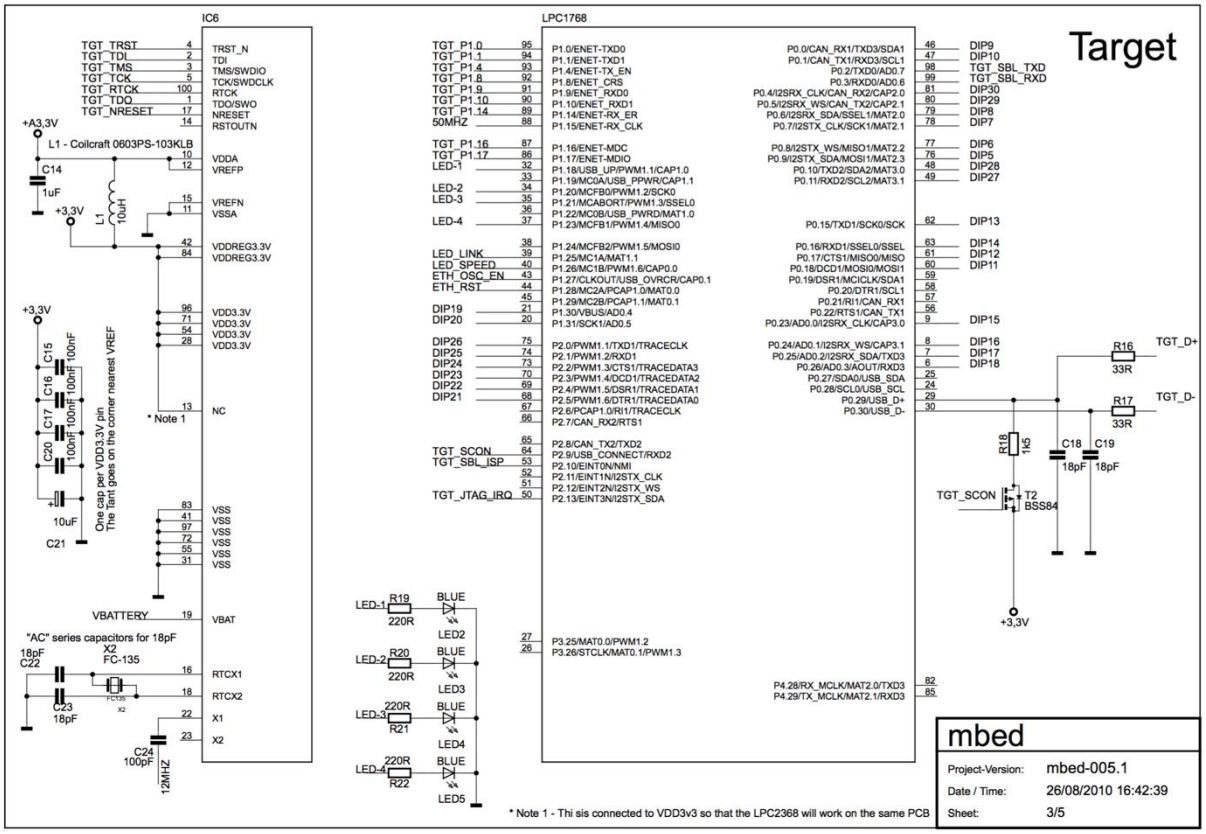
```
      gyroAngle=-(data.Gyro_Proc_X-gyroBias)*0.01;                    // This is deltaangle, how much angle has changed
   //////////////////////
   pAngle=0.98*(pAngle+gyroAngle)+0.02*accAngle-desiredAngle;          // Complementary filter yields best value for current angle
   //iAngle=(0.9*iAngle+0.1*gyroAngle);     DOESNT ACTUALLY INTEGRATE ERROR // Sorta- running average-integral thing
   dAngle=pAngle-dAngle;  //Ang. Veloc. less noisy than dAngle = -(imu.gx-gyroBias);
   iAngle+=(pAngle*.01);// integrate the angle (multiply by timestep to get dt!)
   //if((ki*iAngle/OVERALL_SCALAR)>=3)iAngle=3*OVERALL_SCALAR/ki;// if ki dominates three-fold
   //if((ki*iAngle/OVERALL_SCALAR)<=-3)iAngle=-3*OVERALL_SCALAR/ki;//50 is an arbitrary cap - kind of worked
   // try angle dev. .55 and find value for imu.gx
   if(abs(pAngle-desiredAngle)>=0.6&&abs(pAngle-desiredAngle)<=15) {          // If it is tilted enough, but not too much
      speed=-(ki*iAngle+kd*dAngle+kp*pAngle)/OVERALL_SCALAR;  // drive to correct
      if(speed<-1) speed=-1;                        // Cap if undershoot
      else if(speed>1) speed=1;                     // Cap if overshoot
   } else speed=0;                                  // If we've fallen over or are steady on top
   drive(speed);                                    // Write speed to the motors
}
////////////////////////////////////////////////////////////
// Drive function
void drive(float speed)
{
   int direction=0;                                 // Variable to hold direction we want to drive
   if (speed>0)direction=1;                         // Positive speed indicates forward
   if (speed<0)direction=2;                         // Negative speed indicates backwards
   if(speed==0)direction=0;                         // Zero speed indicates stopping
   switch( direction) {                             // Depending on what direction was passed
      case 0:                                       // Stop case
         motorSpeedLeft=0;                          // Set the DigitalOuts to stop the motors
         motorSpeedRight=0;
         motorDirLeft=0;
         NmotorDirLeft=0;
         motorDirRight=0;
         NmotorDirRight=0;
         break;
      case 1:                                       // Forward case
         motorSpeedLeft=speed;                      // Set the DigitalOuts to run the motors forward
         motorSpeedRight=speed;
         motorDirLeft=1;
         NmotorDirLeft=0;
         motorDirRight=1;
         NmotorDirRight=0;
         break;
      case 2:                                       // Backwards
```

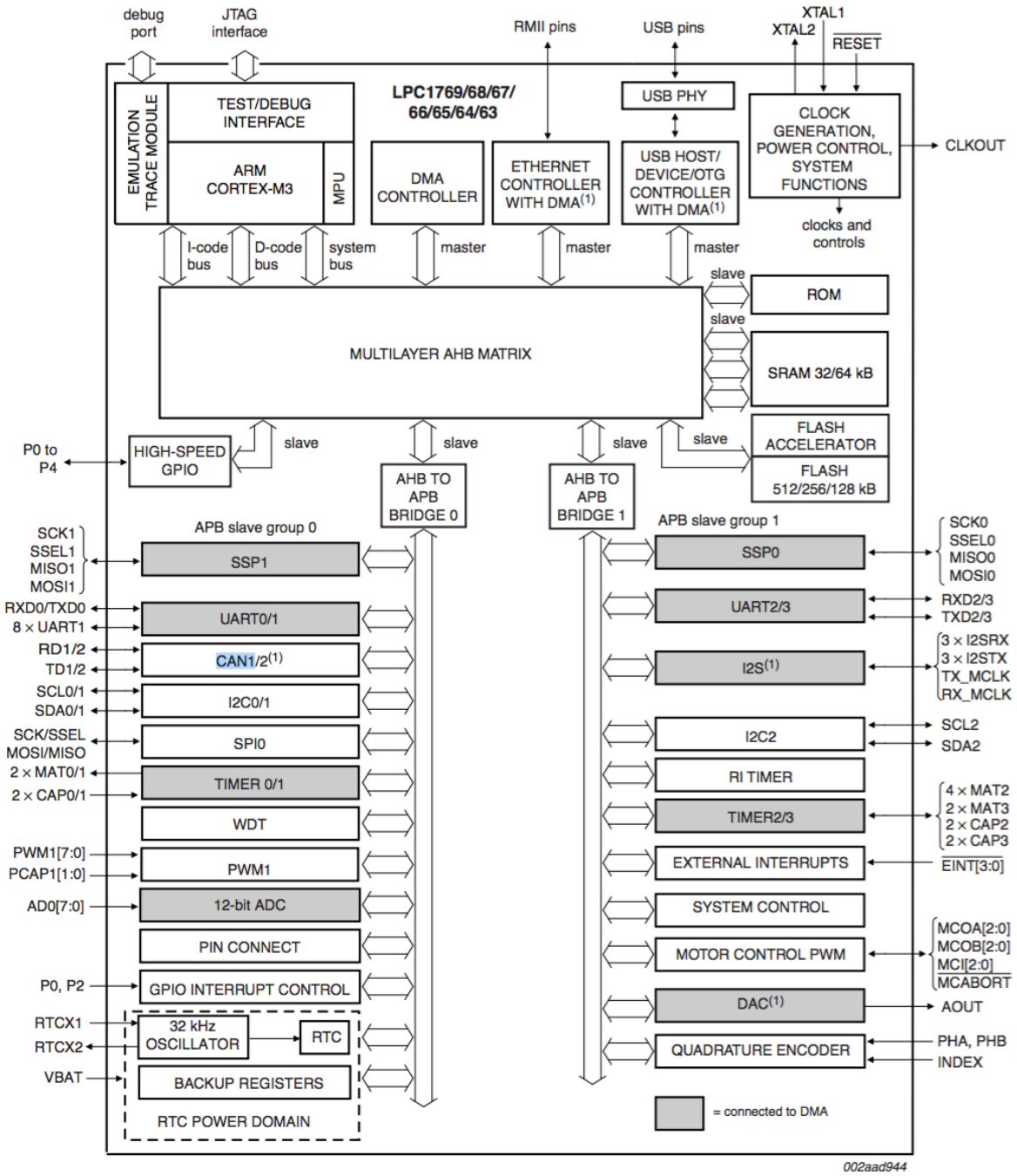
```
            motorSpeedLeft=-speed;                      // Set the DigitalOuts to run the motors
backward
            motorSpeedRight=-speed;
            motorDirLeft=0;
            NmotorDirLeft=1;
            motorDirRight=0;
            NmotorDirRight=1;
            break;
        default:                                    // Catch-all (Stop)
            motorSpeedLeft=0;                       // Set the DigitalOuts to stop the motors
            motorSpeedRight=0;
            motorDirLeft=0;
            NmotorDirLeft=0;
            motorDirRight=0;
            NmotorDirRight=0;
            break;
    }
}
////////////////////////////////////////////////////////////////
// Calibrate function to find gyro drift and accelerometer bias accbias: -0.3 gyrobias: +0.15
void calibrate()
{
    for(int i=0; i<100; i++) {                      // Read a thousand values
                                    // Read the gyro
        accBias=accBias+(-1)*atan2(data.Accel_Proc_Y,data.Accel_Proc_Z)*180/3.142-90; //
Like this, 0 is upright, forward is negative, backward positive
        gyroBias=gyroBias+data.Gyro_Proc_X;                 // Add up all the gyro Biases
    }
    accBias=accBias/100;                            // Convert sum to average
    gyroBias=gyroBias/100;                          // Convert sum to average
#ifdef DEBUG
    pc.printf("accBias: %f gyroBias: %f\n",accBias, gyroBias);  // Print biases to USB
#endif
}
////////////////////////////////////////////////////////////////
// Callback function to change values/gains
#ifdef DEBUG
void callback()
{
    char val;                                   // Needed for Serial communication (need to be
global?)
    val = pc.getc();                            // Reat the value from Serial
    pc.printf("%c\n", val);                         // Print it back to the screen
    if( val =='p') {                            // If character was a 'p'
        pc.printf("enter kp \n");                    // Adjust kp
        val = pc.getc();                        // Wait for kp value
        if(val == 0x2b) {                       // If character is a plus sign
            kp=kp+10;                           // Increase kp
        } else if (val == 0x2d) {                   // If recieved character is the minus sign
            kp=kp-10;                           // Decrease kp
```

```
        } else {
            kp = val - 48;                          // Cast char to float
        }
        pc.printf(" kp = %f \n",kp);                    // Print current kp value to screen
    } else if( val == 'd') {                        // Adjust kd
        pc.printf("enter kd \n");                      // Wait for kd
        val= pc.getc();                             // Read value from serial
        if(val == '+') {                            // If given plus sign
            kd++;                                   // Increase kd
        } else if (val == '-') {                        // If given negative sign
            kd--;                                   // Decrease kd
        } else {                                    // If given some other ascii (a number?)
            kd = val - 48;                          // Set derivative gain
        }
        pc.printf(" kd = %f \n",kd);                    // Print kd back to screen
    } else if( val == 'i') {                        // If given i - integral gain
        pc.printf("enter ki \n");                       // Prompt on screen to ask for ascii
        val= pc.getc();                             // Get the input
        if(val == '+') {                            // If given the plus sign
            ki++;                                   // Increase ki
        } else if (val == '-') {                        // If given the minus sign
            ki--;                                   // Decrease ki
        } else {                                    // If given some other ascii
            ki = val - 48;                          // Set ki to that number
        }
        pc.printf(" ki = %f \n",ki);
    } else if( val == 'o') {
        pc.printf("enter OVERALL_SCALAR \n");
        val= pc.getc();
        if(val == '+') {
            OVERALL_SCALAR=OVERALL_SCALAR+1000;
        } else if (val == '-') {
            OVERALL_SCALAR=OVERALL_SCALAR-1000;
        } else {
            OVERALL_SCALAR=(val-48)*1000;;
        }
        pc.printf(" OVERALL_SCALAR = %f \n",OVERALL_SCALAR);    }
}
#endif.
```

3. Les DATASHEET







(1) Not available on all parts. See Table 2.

Fig 1. Block diagram

ANNEX

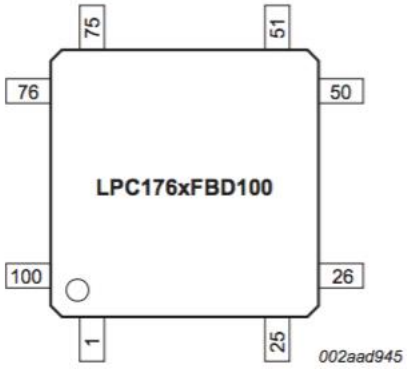


Fig 2. Pin configuration LQFP100 package

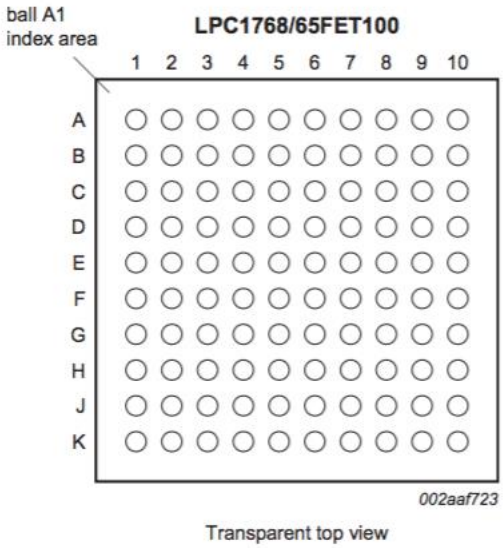
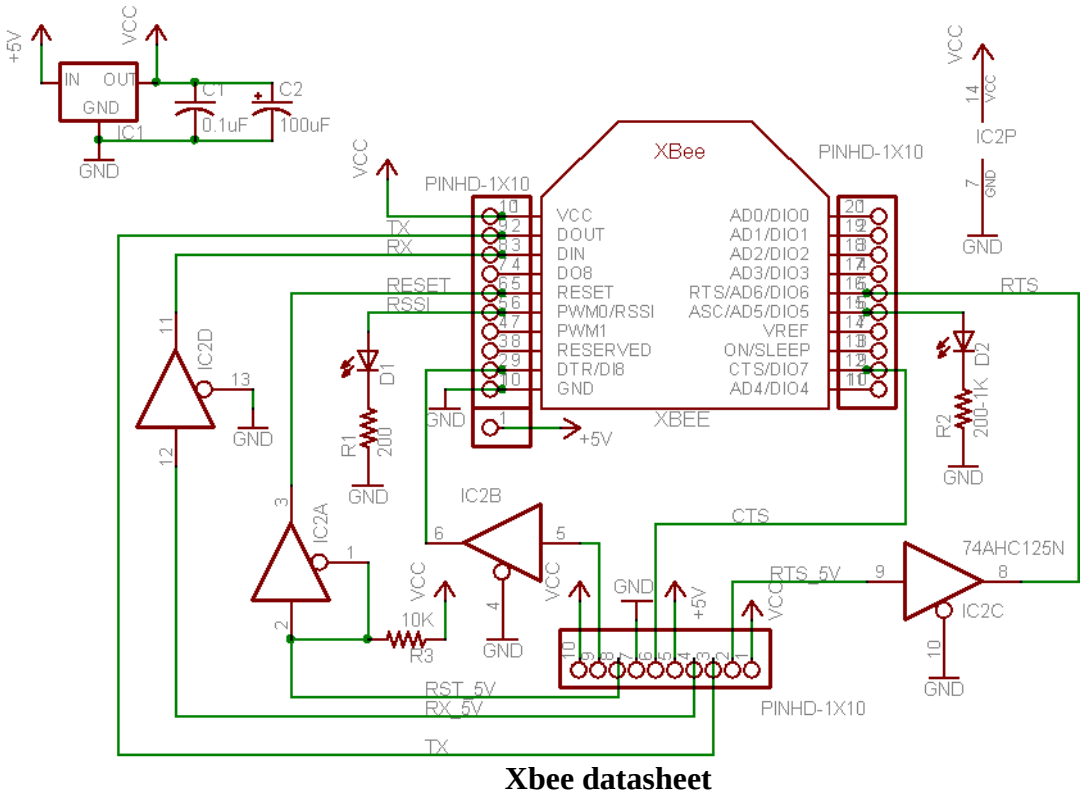


Fig 3. Pin configuration TFBGA100 package

Table 5 - UM6 Pin Descriptions (bottom pin headers)

Pin Description		
Pin #	Pin Name	Description
1	+3.3V	3.3V output
2	GND	Ground
3	MOSI	SPI MOSI pin
4	MISO	SPI MISO pin
5	SCK	SPI SCK pin
6	SS	SPI SS pin
7	Vin	Input voltage: must be between 3.5 V and 5V
8	GND	Ground
9	SCL	i2c SCL line (10k internal pullup)
10	SCL	Connected internally to pin 9
11	SDA	i2c SDA line (10k internal pullup)
12	BOOT0	Programming pin for upgrading firmware. Pulling this pin high before applying power causes the firmware bootloader to start.
13	PA0	MCU pin PA0
14	RX2	RX pin for UART2 (Output from GPS)
15	PA4	MCU pin PA4
16	PA6	MCU pin PA6
17	PB0	MCU pin PB0
18	RX1	RX pin for UART1
19	TX1	TX pin for UART1
20	PB1	MCU pin PB1
21	PA7	MCU pin PA7
22	PA5	MCU pin PA5
23	TX2	TX pin for UART2 (Input from GPS)
24	PA1	MCU pin PA1



Xbee datasheet

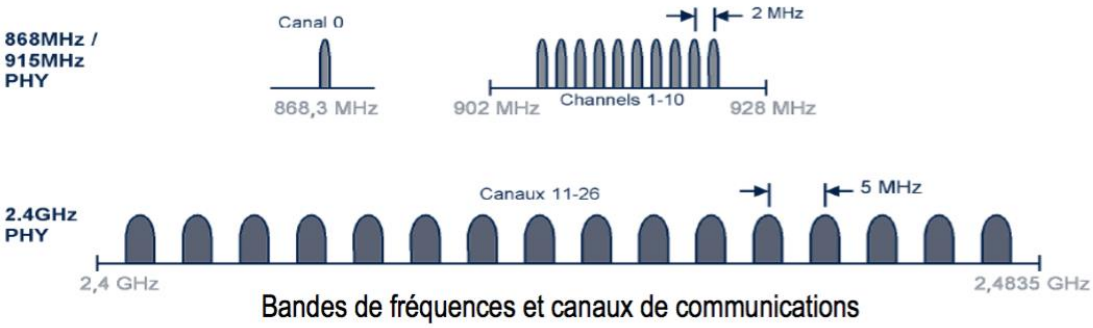
Protocole ZigBee

- Les réseaux de capteurs sans fil ont besoin d'un :
  - Protocole à faible consommation d'énergie,
  - Grand nombre nœuds communicants.
- ZigBee :
  - Protocole de communication des systèmes radio à consommation réduite basé sur la norme IEEE 802.15.4
  - Utilisé pour les réseaux à dimension personnelle (WPAN : Wireless Personal Area Network).
- Comparaison avec les autres protocoles :

Protocole	Zigbee	Bluetooth	Wi-Fi
IEEE	802.15.4	802.15.1	802.11a/b/g/n/n-draft
Besoins mémoire	4-32 ko	250 ko +	1 Mo +
Autonomie avec pile	Années	Mois	Heures
Nombre de nœuds	65 000+	7	32
Vitesse de transfert	250 kb/s	1 Mb/s	11-54-108-320 Mb/s
Portée	1000 m-1500 m	50 m	300 m

Caractéristiques

- Fréquences
  - 868 MHz pour la région d'Europe, elle possède 1 canal (20 kbit/s)
  - 915 MHz pour l'Amérique du nord et l'Australie, elle est devisée en 10 canaux séparés de 2 MHz (40 kbit/s)
  - 2,4 GHz pour le monde, elle est devisée en 16 canaux séparés de 5 MHz (20 kbit/s)
- Le protocole ZigBee offre une alternative de choix de la couche physique (PHY)
  - couche physique pour les bandes de fréquences 868/915MHz
  - couche physique pour la bande fréquence 2.4 GHz



Bandes de fréquences et canaux de communications