

République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et de La Recherche Scientifique
Université Djilali Bounaama Khemis Miliana



Faculté des Sciences et de la Technologie

Département de la Technologie

Mémoire Présenté

Pour l'Obtention de Diplôme

Master

En AUTOMATIQUE

Spécialité Automatique et Informatique Industrielle

Thème

**Contribution à la Commande des
Systèmes Sous-Actionnés**

présenté et soutenu par

Mr. SERDOUN Djillali

Jury :

Mr. CHOUYA Ahmed	UDBKM	Président
Mr. KERRACI Abdelkader	UDBKM.	Encadreur
Mr. BOUREGUIG Kada	UDBKM	Examineur

Année Universitaire 2018/2019

Je tiens à dédier ce mémoire :

A ma très chère **Mère**,

A mon très cher **Père**,

A mon très cher frère **Mohammed** ,

A mes Chères**Soeurs**,

A toute **ma famille**.

A tous mes amis du club **Automatique et informatique industrielle** et spécialement
Rais, Rahmoune et Yacine.

A vous tous un grand merci ...

SERDOUN DJILLALI

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 remerciements à mon encadreur M. KERRACI Abdelkader de m'avoir guidé et encouragé durant ce travail.

Je tiens à les remercier pour avoir accepté d'être directeurs de ce mémoire, pour leurs remarques qui m'ont permis de finaliser au mieux ce travail .

Je tiens à remercier les membres du jury Mr. CHOUYA Ahmed at M. BOUREGUIG Kada qui me font l'honneur d'évaluer ce travail.

Je tiens également à remercier tout les membres de ma famille qui m'ont toujours soutenu tout au long de mes études.

Je souhaite aussi remercier tous mes enseignants du la filière Génie Électriquee.

Résumé

Dans ce travail, nous nous intéressons à la réalisation d'une plateforme de prototypage pour un robot auto-balancé sur deux roue à l'aide du **Carte Arduino MEGA 2560** utilisée comme carte Data-Acquisition pour la commande en temps réel du système. Le robot auto-balancé est un pendule inversé qui est un système non linéaire sous-actionné instable. Deux lois de commande ont été appliqué sur le système dans son état continue, comme dans sont discrète. Un prototype du robot a été réalise et commandé en temps réel par le logiciel Matlab/Simulink via une carte d'acquisition. Les résultats obtenues et surtout dans le cas discret sont très satisfaisantes.

Mots clés : Système sous-actionnée, Robot auto balancier, Modèle dynamique, Systeme continuus, Systemes discrets, LQR, Commande linéaire.

ملخص

في هذا العمل، نحن مهتمون بإنجاز منصة نموذجية للروبوت المتوازن ذاتيًا على عجلتين باستخدام Arduino MEGA 2560 المستخدم كبطاقة لتبادل البيانات للتحكم في الوقت الفعلي للنظام. روبوت التوازن الذاتي هو بولمقل لودنة نظام غير خطي غير مستقر يعمل دون المستوى. تم تطبيق قانونين للتحكم على النظام في حالته التماثلية، كما في الحالة الرقمية. تم إنجاز نموذج أولي من الروبوت والتحكم فيه في الوقت الفعلي بواسطة برنامج Simulink/Matlab. النتائج التي تم الحصول عليها وخاصة في الحالة الرقمية مرضية للغاية.

الكلمات الرئيسية: نظام تحت الطاقة، روبوت ذاتي التوازن، نموذج ديناميكي، نظام مستمر، أنظمة منفصلة، التحكم التريبي الخطي، تحكم خطي.

Abstract

In this work, we are interested in the realization of a prototyping platform for a robot self-balanced on two wheels using the **Arduino MEGA 2560 Card** used as a Data-Acquisition card for real-time control of the system. The self-balancing robot is an inverted pendulum that is an unstable under-actuated non-linear system. Two control laws have been applied on the system in its continuous state, as in are discrete. A prototype of the robot was realized and controlled in real time by the Matlab/Simulink software via an acquisition card. The results obtained and especially in the discrete case are very satisfactory.

Keywords : Under-actuated system, Self-balancing robot, Dynamic model, Continuous system, Discrete systems, LQR, Linear control.

Table des matières

Dédicaces	i
Remerciements	ii
Résumé	iii
Abstract	iii
Table des matières	iv
Liste des Notations et Abréviations	xii
Introduction	1
1 Généralité sur les Systèmes sous Actionnés	3
1.1 Introduction :	3
1.2 Historique	4
1.3 Systèmes mécaniques complètement actionnés	5
1.4 Systèmes mécaniques sous-actionnés	5
1.4.1 Le pendubot	6
1.4.2 L'Acrobot	7
1.4.3 Le pendule de Furuta :	8
1.4.4 Le pendule à roue inertielle	9
1.4.5 Le pendule inversé sur chariot	10
1.5 L'intérêt de l'étude d'un pendule inversé	10
1.6 Applications des systèmes sous actionnée	11
1.6.1 Domaine de Robotique	11
1.6.2 Domaine de la médecine	12
1.6.3 Appareil de transport personnel Segway	12
1.6.4 Le domaine de l'aérospatiale :	13

1.6.5	Domaine de loisir	13
1.7	Les différents types des robots mobiles	14
1.7.1	Robots mobiles à roues :	15
1.7.2	Robots mobiles à pattes	16
1.7.3	Robot auto-balancé (pendule inversé)	16
1.8	Principe de fonctionnement d'un pendule inverse	17
1.9	Conclusion :	17
2	Modélisation d'un robot auto balancier	18
2.1	Introduction	18
2.2	Modélisation mathématique	18
2.2.1	Définition	18
2.3	Modèle linéaire d'un moteur à courant continu (DC)	19
2.4	Modélisation mathématique du pendule inversé	21
2.5	Modélisation du système à deux degrés de liberté	22
2.5.1	Énergie cinétique du système en mouvement	22
2.5.2	Énergie potentielle du système	23
2.5.3	Équation de Lagrange	23
2.6	Linéarisation du modèle autour des points d'équilibre	24
2.6.1	Position d'équilibre instable ($\theta = 0$)	24
2.6.2	Position d'équilibre stable ($\theta = \pi$)	25
2.7	Modèle dynamique pour un pendule inversé à deux roues	25
2.8	Conclusion	30
3	Commande d'un robot auto balancier	31
3.1	Introduction	31
3.2	Commande par retour d'état continue	32
3.2.1	Principe de la commande par retour d'état	32
3.2.2	Résolution d'un problème de retour d'état par placement de pôle	33
3.2.3	Commande par placement de pôle	33
3.2.4	Principe de la commande par placement de pôle	33
3.2.5	Commande optimale	34
3.2.6	Commande linéaire quadratique	34
3.2.7	Définition de la fonction linéaire quadratique définie positive	34
3.2.8	Principe de la commande linéaire quadratique	34
3.2.9	Synthèse de régulateur LQR	35
3.2.10	Choix d'une matrice de pondération	36
3.2.11	Critère d'optimisation pour le régulateur	37

3.2.12	Gains du régulateur	37
3.3	Commande discret	37
3.3.1	L'objectif de la commande discret	39
3.3.2	Commande sous Matlab :	39
3.3.3	Régulateur discret	39
3.4	Résultat de simulation	40
3.4.1	Commande linéaire continue	40
3.4.2	Commande linéaire discrète	42
3.4.3	Interprétation des résultats	43
3.5	Conclusion	44
4	Conception et réalisation d'un robot auto balancier	45
4.1	Introduction :	46
4.2	La structure de notre robot	46
4.3	Architecture générale	47
4.4	Généralité sur les matérielles utilisées	47
4.4.1	Partie hardware	47
4.4.2	L'utilité de la carte ARDUINO MEGA 2560	51
4.5	Partie software	51
4.5.1	MATLAB	52
4.5.2	Simulink	52
4.5.3	Arduino IDE	55
4.5.4	Solidwork	56
4.6	L'environnement Matlab/Simulink	57
4.7	L'interfaçage Arduino/Matlab/Simulink	57
4.7.1	Carte Arduino comme d'interface	57
4.8	ArduinoIO	59
4.9	Arduino Target	59
4.9.1	Installation du package ArduinoIO	59
4.9.2	Exploitation de la bibliothèque ArduinoIO sous Simulink	60
4.9.3	Exploitation du package ArduinoIO sous Matlab	60
4.10	Arduino Target	61
4.11	les bibliothèques utilisent	61
4.11.1	compilateur de C++	61
4.11.2	ArduinoIO	61
4.11.3	Embedded Coder Target for Arduino	62
4.11.4	RASPLIB	62
4.11.5	Afficheur LCD	62

4.11.6 MPU6050	63
4.11.7 Driver moteur	64
4.11.8 Bluetooth	64
4.12 traitement des données	64
4.12.1 présentation de ADC	64
4.12.2 Acquisition des donnée	65
4.12.3 Envoie des données	67
4.13 Détermination de l'angle à partir des données d'accéléromètre et de gyroscope	68
4.13.1 Calculer l'angle de position angulaire avec les données du gyroscope	68
4.13.2 Calculer de la position angulaire avec les données de l'accéléromètre	69
4.13.3 Création du filtre complémentaire	70
4.13.4 Contrôle de la position linéaire à l'aide d'un encodeur incrémental	71
4.13.5 Capteur de distance Ultrason HC-SR04 :	71
4.13.6 commande LQR	72
4.13.7 commande PID	72
4.14 Présentation du robot sous Solidwork	72
4.14.1 l'interface de commande	73
4.15 Réalisation du robot	73
4.15.1 Les étapes de réalisation	73
4.16 Présentation du robot	75
4.17 Conclusion	76
Conclusion Générale	77
Annexe A	78
Annexe B	82
Annexe C	86
Bibliographie	92

Table des figures

1.1	Jœ-le Pendule(EPFL)	4
1.2	Robot industrielle	5
1.3	Exemple de système mécanique sous actionnée : robot Segway	6
1.4	Le Pendubot	7
1.5	L'Acrobot en coordonnées généralisés.	8
1.6	Le pendule de Furuta en coordonnées généralisées	9
1.7	Le pendule à roue inertielle en coordonnées généralisées.	9
1.8	Scooter intelligent (Segway) à une et deux roues, le rebot (MIP one)	10
1.9	Exemples de robots inversés.	11
1.10	a) Nbot et b) Legway [31]	11
1.11	Le corps de l'être humain vu comme double pendule	12
1.12	Le Segway HT [31]	13
1.13	Gyroscopique	13
1.14	Tour d'amusement (Ciseaux)	14
1.15	Exemples de pendules inversés	15
1.16	Robot Auto-Balancé	16
1.17	Exemple sur l'équilibre à base du pendule inversé	17
2.1	Schémas d'un moteur à courant continu	19
2.2	Schéma de l'ensemble chariot et pendule inversé	21
2.3	Schémas des corps libre pour les deux roues	26
2.4	Corps libre d'un châssis	27
2.5	Robot balancier à deux roues	29
2.6	Réponse impulsionnelle du système	30
3.2	Principe de la loi de commande placement de pôles	33
3.1	Bouclage du système par un vecteur de gain	33
3.3	Contrôleur continue	38

3.4	Contrôleur numérique	38
3.5	Résultat des signaux discrets à partir des signaux continus	39
3.6	Résultat de simulation de la commande <i>LQR</i> continue pour $\theta = 0$	40
3.7	Signaux d'erreurs	40
3.8	Résultat de simulation de la loi commande <i>LQR</i> continue pour $\theta = 0$	41
3.9	Signaux d'erreurs	41
3.10	Résultat de simulation de la loi de commande placement de pôle crête pour $\theta = 0$	42
3.11	Résultat de simulation de la commande <i>LQR</i> discrète pour $\theta = 0$	42
3.12	Robot auto balancier en 2D sous MATLAB	43
4.1	Architecture générale du robot	47
4.2	Moteur à courant continu	48
4.3	MPU6050	48
4.4	Driver d'un moteur à courant continu	49
4.5	Afficheur LCD	49
4.6	Module de communication H-05	50
4.7	Alimentation $9v$	50
4.8	Arduino MEGA 2560	51
4.9	Modèle Stateflow sous MATLAB	53
4.10	Génération du code sous MATLAB	53
4.11	L'interface de commande en temps réel	53
4.12	Bibliothèques des blocks sous Simulink	54
4.13	Vérification du code Simulink	54
4.14	Création des objets en temps réel	55
4.15	L'environnement ARDUINO IDE	56
4.16	L'interface de Solidwork avec les différents composants principales	56
4.17	Image réel sous Solidwork	57
4.18	Emplacement de la bibliothèque "Instrument Control Toolbox"	58
4.19	Les blocs pour la communication série	58
4.20	Paramétrage des blocs pour la communication série	58
4.21	Librairies de l'ArduinoIO	60
4.22	Blocs d'ArduinoIO nécessaires pour la commande	60
4.23	Bibliothèque utilisée en Matlab 2018 pour la compilation	61
4.24	Codeur embarqué cible Arduino	62
4.25	Bibliothèque utilisée en Hardware	62
4.26	LCD Hardware	62
4.27	MPU6050	63

4.28	Driver du moteur	64
4.29	Module de communication bluetooth	64
4.30	Convertisseur Analogique Numérique	65
4.31	Modèle Similink sous MATLAB	66
4.32	Modèle Simulink d'un encodeur	66
4.33	Capteur ultra son avec Arduino méga 2560	67
4.34	Modèle Simulink d'un capteur ultra son	67
4.35	Modèle Simulink de position angulaire à l'aide d'un gyroscope	68
4.36	Bloc de paramètres d'un gyroscope	69
4.37	Position angulaire à l'aide d'un accéléromètre	69
4.38	Modèle Simulink d'un angle estimé	70
4.39	Position linéaire en temps réel	71
4.40	Capteur ultra sonic	71
4.41	Branchement du HC-SR04 avec la carte Arduino UNO	72
4.42	Commande LQR avec PID	72
4.43	Robot auto balancier sous Solidwork	73
4.44	L'interface de commande en temps réel	73
4.45	Test de la communication entre MPU6050 et MATLAB	74
4.46	Raccordement des deux roues avec le châssis	74
4.47	Assemblage du Robot	74
4.48	Robot final version 1	75
4.49	Robot final version 2	75
50	Modèle simulink de la commande LQR d'un robot auto balancier	79
51	Régulateur utilisé avec le système	79
52	Modèle Simulink du système	80
53	Schéma synoptique de la commande numérique	82
54	Discretisation d'un système continu	83
55	L'état des systèmes numériques	83
56	Commande numérique en boucle fermée	84
57	Schéma fonctionnel d'une commande numérique	84
58	Modèle Simulink d'un Observateur de sortie	85
59	Caractéristique de l'arduino méga 2560	86
60	Un drone avec deux capteurs, accéléromètre tridimensionnel et gyroscope 3D	87
61	Le modèle d'accéléromètre MEMS utilisé pour les systèmes mécaniques	87
62	mpu6050 réel	88
63	Schéma structurel de la carte du capteur	88
64	Configuration d'un I2C	88

65	Paramétrage du gyroscope	89
66	Configuration du RASPLIB	89
67	Configuration du hardware	90
68	Vue Globale de la Réalisation	90

Liste des Notations et Abréviations

Liste des notations

b	Coefficient de frottement des roues du chariot
C_s	Frottement sec
d	Coefficient de frottement de rotation du pendule
e	Erreur entre la tâche r et la sortie du processus y
E_{cc}	Energie cinétique du pendule
E_{cm}	Energie cinétique du chariot
$F(t)$	Force exercée sur le chariot
f_d	Energie dissipée par frottement
g	Intensité de la pesanteur
i, j	Vecteurs unitaires du repère x, y
I_a	Intensité du courant
J_r	Inertie du rotor
K_r	Constante du couple de charge
K_D	Coefficient dérivé
K_e	Constantes des couples électriques
K_I	Coefficient intégral
K_I	Constante d'intégration
K_m	Constantes des couples mécaniques
K_P	Coefficient proportionnel
l	Demi-longueur du pendule
L	Lagrangien du système
L_a	Inductance de l'induit
m	Masse du pendule
M	Masse du chariot
ω	Vitesse angulaire du rotor
$\omega(t)$	Vitesse angulaire du pendule
r, L	Résistance du moteur et l'inductance
R_a	Résistance de l'induit
r_c	Position du centre de gravité du pendule
$\theta(t)$	Angle de rotation du pendule
u	Variable générique contrôlée
V	Tension d'entrée du moteur
V_c	Tension d'alimentation de l'induit du moteur
V_c	Vitesse de centre de gravité du pendule
$x(t)$	Position du chariot

Liste des abréviations

<i>3D</i>	3 dimensions
<i>AC</i>	Alternative current (Courant Alternatif)
<i>ADC</i>	Analog-Digital Converter (Convertisseur Analogique/Numérique)
<i>AUV</i>	Autonomous Underwater Vehicle (Véhicule sous-marin autonome)
<i>CPU</i>	Central Processing Unit (Unité centrale de traitement)
<i>DC</i>	Direct Current (Courant continu)
<i>DDL</i>	Degré De Liberté
<i>DMP</i>	Digital Motion Processor (Processeur de mouvement numérique)
<i>E/S</i>	Entrées/Sorties
<i>GSC</i>	Ground station control (Station de cotrole au sol)
<i>I/O</i>	Input/Output (Entrée/Sortie)
<i>I2C</i>	Inter-Integrated Circuit (Circuit inter-intégré)
<i>IDE</i>	Integrated Development Environment (Environnement de développement intégré)
<i>IMU</i>	Inertial measurement unit (Unité de mesure inertielle)
<i>INS</i>	inertial navigation systems (systèmes de navigation inertielle)
<i>LCD</i>	Liquid Crystal Display (Affichage à cristaux liquides)
<i>LiPo</i>	Lithium Polymer
<i>LMI</i>	Linear Matrix Inequality (Inégalité matricielle linéaire)
<i>LQ</i>	Linear Quadratic (Quadratique linéaire)
<i>LQG</i>	Linear-Quadratic-Gaussian (Linéaire-quadratique-gaussien)
<i>LQR</i>	Linear-Quadratic Regulator (Régulateur linéaire-quadratique)
<i>LTI</i>	Linear Invariant Time (Temps invariant linéaire)
<i>MEMS</i>	Micro-Electro-Mechanical Systems (Systèmes micro-électro-mécaniques)
<i>MIT</i>	Massacheutts Institute of Technology
<i>MLI</i>	Modulation de Largeur d'Impulsion
<i>MPC</i>	Model Prédictive Control (Modèle de contrôle prédictif)
<i>PD</i>	Proportional Derivative (Proportionnelle Dérivée)
<i>PI</i>	Proportional Integral
<i>PID</i>	Proportional Integral Derivative
<i>PWM</i>	Pulse-Width Modulation (Modulation de largeur d'impulsion)
<i>RGB</i>	Red Green Blue
<i>Rot</i>	Rotationnelle
<i>RPM</i>	Rotation Par Minute
<i>SCL</i>	Serial Clock Line
<i>SDA</i>	Serial Data Line
<i>SIMO</i>	Single Input Multiple Outputs
<i>SPI</i>	Serial Peripheral Interface
<i>UART</i>	Universal Asynchronous Receiver/Transmitter (Récepteur/émetteur asynchrone universel)
<i>USB</i>	Universal Serial Bus (Bus série universel)

Introduction Générale

Ces dernières années, les systèmes sous-actionnés, deviennent un sujet populaire avec de nombreuses recherches en robotique qui ont permis de réaliser plusieurs systèmes basés sur le modèle du pendule inversé, tels que JOE, Segway, Legway, etc.

De nombreux chercheurs se concentrent sur le contrôle des systèmes pendulaires inversés à deux roues, qui sont largement utilisés dans de nombreux domaines, notamment la robotique, les applications médicales et les véhicules intelligents.

Ces systèmes sont capables de se balancer sur deux roues et de faire demi-tour. De tels robots qui ont ces capacités peuvent facilement se déplacer dans des zones très variées : les angles vifs, les zones dangereuses avec des petits pas et des bordures, et des situations comme lorsqu'il n'y a pas d'aide au contrôle de l'IPS (Inverse Pendulum System) sont proposées pour permettre la stabilité des systèmes puisque l'IPS est instable de nature.

En 2006, Nawawi et al. Présentent un modèle mathématique de l'IPS à deux roues. Ils ne déclarent que leur modèle proposé réussit dans un environnement expérimental réaliste grâce à ses deux modèles de contrôleurs linéaires différents[1]. Ching-Chih Tsai et al. Ont mis au point deux types de contrôleurs adaptatifs différents pour assurer l'équilibre et le roulis de l'appareil véhicules à deux roues à équilibrage automatique utilisant la fonction radiale basée sur le réseau neuronal (RFBNN). En 2010, Huang et al. étudient le problème du suivi de la vitesse de l'IPS à roues mobiles. Ching-Chih Tsai et al. ont mis au point deux types de contrôleurs adaptatifs différents pour assurer l'équilibre et le roulis de l'appareil..[1]. Ils énoncent deux méthodes de contrôle en mode glissant pour éliminer les incertitudes des systèmes et les perturbations externes. En 2013, Jian-Xin Xu et al. ont proposé une nouvelle implémentation du contrôleur logique flou de type TakagiSugeno pour un robot mobile à deux roues et pendule inversé[1]. En 2014, Jian-Xin Xu et al. ont appliqué une méthode de contrôle intégral en mode glissant pour équilibrer les robots mobiles à deux roues.

Le principal problème de l'IPS sur roues est que l'utilisation du contrôle de rotation de la roue, la stabilité simultanée de la position verticale du pendule inversé (IPS) normalement instable et la position souhaitée de la roue doit être atteinte.

Pour bien traiter ce problème nous avons opté l'organisation du mémoire comme suit :

Dans le Chapitre 01, nous allons présenter des généralités sur les robots mobiles autonomes en donnant quelques exemples des modèles mathématiques des systèmes sous-actionnés et complètement actionnés.

Le chapitre 2 sera consacré à la modélisation mathématiques du moteur à courant continu et des systèmes mécaniques sous actionnée à deux degrés de liberté (2DDL) pour les utiliser dans la phase de commande de notre système.

Dans le chapitre 3, le modèle d'état calculé dans le chapitre 2 sera utilisé avec deux types de

commande linéaire et qui seront validés par simulation sous l'environnement Matlab/Simulink. les mêmes commande seront appliquées après sur le modèle discret du système.

Enfin et dans le chapitre 4, nous allons présenté toute la démarche suivit dans la réalisation de notre prototype de robot auto balancier, ainsi nous donnons d'une façons détaillée la méthode de mise en marche de notre plateforme.

Généralité sur les Systèmes sous Actionnés

Sommaire

1.1 Introduction :	3
1.2 Historique	4
1.3 Systèmes mécaniques complètement actionnés	5
1.4 Systèmes mécaniques sous-actionnés	5
1.4.1 Le pendubot	6
1.4.2 L'Acrobot	7
1.4.3 Le pendule de Furuta :	8
1.4.4 Le pendule à roue inertielle	9
1.4.5 Le pendule inversé sur chariot	10
1.5 L'intérêt de l'étude d'un pendule inversé	10
1.6 Applications des systèmes sous actionnée	11
1.6.1 Domaine de Robotique	11
1.6.2 Domaine de la médecine	12
1.6.3 Appareil de transport personnel Segway	12
1.6.4 Le domaine de l'aérospatiale :	13
1.6.5 Domaine de loisir	13
1.7 Les différents types des robots mobiles	14
1.7.1 Robots mobiles à roues :	15
1.7.2 Robots mobiles à pattes	16
1.7.3 Robot auto-balancé (pendule inversé)	16
1.8 Principe de fonctionnement d'un pendule inverse	17
1.9 Conclusion :	17

1.1 Introduction :

La commande des véhicules non-holonomes est un domaine de recherche très actif. Plusieurs raisons contribuent à cet engouement. La première est que les véhicules sur roues constituent de nos jours le moyen de transport individuel principal. Leur automatisation, précédemment limité aux expérimentations en laboratoire et à des utilisations professionnelles, les chariots de manutention dans les usines par exemple est maintenant envisagée pour des applications grand public : convois de véhicule sur autoroute, systèmes de transport urbain intelligent.[2]

Ces nouvelles applications, qui nécessitent de coordonner les mouvements de plusieurs véhicules, donnent lieu à de nouveaux problèmes d'automatique. Une autre raison plus technique tient au fait que les équations régissant le déplacement des véhicules non-holonomes sont fortement non-linéaires et revêtent un intérêt théorique particulier dans le domaine de l'automatique non-linéaire.

1.2 Historique

Les systèmes robotiques sont des systèmes mécaniques dotés d'actionneurs permettant de contrôler l'évolution du système dans le temps. Suivant le nombre de degrés de liberté et d'actionneurs du système, il en découle 3 classes de systèmes mécaniques, à savoir : les systèmes mécaniques complètement actionnés, les systèmes mécaniques sous-actionnés, et les systèmes mécaniques sur-actionnés (ou redondants)[3].

Il faut rappeler que l'idée d'un tel véhicule avait germé en 1996, au laboratoire d'électronique Industrielle. Après une étude théorique démontrant la faisabilité, qui a conduit à une thèse, suivie du processus de développement, le robot miniature Joe-le Pendule (figure 1.1).



FIGURE 1.1 – Joe-le Pendule(EPFL)

Ce développement est directement inspiré du célèbre modèle académique du pendule inversé. Il est constitué d'un chariot mobile surmonté d'un pendule inversé, librement articulé autour d'un axe transversal. La particularité de ce système c'est que le chariot et le pendule ne font qu'un, l'articulation étant elle-même l'axe des roues.

1.3 Systèmes mécaniques complètement actionnés

Considérons le système :

$$M(q)\ddot{q} + C(q, \dot{q})\dot{q} + G(q) = B(q)u \quad (1.1)$$

Ce système mécanique est dit complètement actionné si le nombre des entrées de commande est égal au nombre de degrés de liberté $\text{rang}B(q) = m = n$ ou, autrement dit, $B(q)$ est une matrice carrée inversible. Par conséquent, les systèmes mécaniques complètement actionnés sont linéarisables par retour d'état statique. Ceci peut être montré en appliquant le contrôle suivant :

$$u = B(q)^{-1}(M(q)v + C(q, \dot{q})\dot{q} + G(q)) \quad (1.2)$$

Il existe plusieurs systèmes mécaniques complètement actionnés spécialement dans les domaines industriels comme les robots manipulateurs (voir figure 1.2).

qui représente dans la figure 1.2



FIGURE 1.2 – Robot industrielle

1.4 Systèmes mécaniques sous-actionnés

On considère le système mécanique, décrit par :

$$\ddot{q} = f(q, \dot{q}) + G(q)u \quad (1.3)$$

Ou :

q : Est le vecteur d'état de coordonnées généralisées.

$f(q, \dot{q})$: Est le champ de vecteur qui capture la dynamique du système.

\dot{q} : Est le vecteur de vitesse généralisé.

G : Est la matrice de l'entrée.

u : Est le vecteur des entrées généralisées

Un système mécanique est dit sous-actionné s'il admet moins d'actionneurs que de degrés de liberté, soit $m = \text{rang}G(q) < n = \text{dim}(q)$ Cette restriction empêche une linéarisation par bouclage statique de la dynamique complète du système.



FIGURE 1.3 – Exemple de système mécanique sous actionnée : robot Segway

Dans ce mémoire, nous ne considérerons que des systèmes mécaniques sous-actionnés ayant deux degrés de liberté dont, évidemment, l'un est actionné et l'autre pas. On ne s'intéresse pas aux systèmes ayant plus que deux degrés de liberté. Comme par exemple, le pendule inversé, le pendu-bot, l'acrobate, le pendule de Fureta et le pendule à roue inertielle.

Nous présenterons brièvement dans cette section chacun de ces systèmes, puis nous présenterons en détail le système pendule-chariot (ou pendule inversé) qui nous servira d'exemple d'application des lois de commande qui seront développées dans la suite du mémoire. Notons que ces lois de commande peuvent aussi être appliquées aux autres systèmes.

Afin de faciliter la modélisation de ces systèmes, on considéra que les pendules admettent des masses ponctuelles localisées à leurs extrémités, c'est-à-dire qu'on supposera que l'on s'est déjà ramené à un système où les tiges ont des masses nulles. [4]

1.4.1 Le pendubot

Le pendu-bot, représenté dans la figure 1.4, est constitué de deux tiges qui peuvent tourner autour de leurs axes respectifs. La tige 1, de masse m_1 et de longueur l_1 ; est actionnée par un couple de contrôle τ tandis que la tige 2; de masse m_2 et de longueur l_2 , est en rotation libre autour de la tige 1.

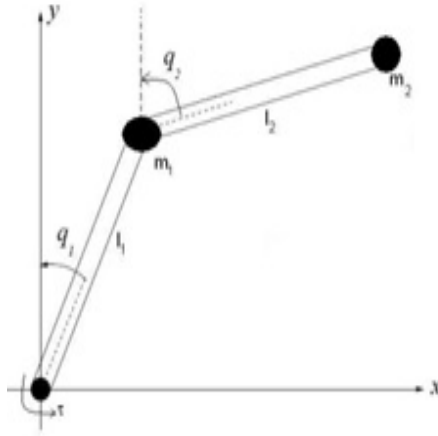


FIGURE 1.4 – Le Pendubot

A vitesse nulle, le pendubot admet une infinité de points d'équilibre instables donnés par $\dot{q}_1 = \dot{q}_2 = \dot{q}_2 = 0$, $q_1 = cte$, correspondant à la position haute de la tige 2 pour toute position de la première tige, et une infinité de points d'équilibre stables donnés par $\dot{q}_1 = \dot{q}_2 = 0$; $q_2 = \pi$; correspondant à la position basse de la tige numéro 2 : Le lagrangien et les équations dynamiques du mouvement sont donnés par :

$$\begin{aligned} \mathcal{L} = & \frac{1}{2}(m_1 + m_2)l_1^2\dot{q}_1^2 + \frac{1}{2}m_2l_2^2\dot{q}_2^2 + 2m_2l_1l_2\dot{q}_1\dot{q}_2 \cos(q_1 - q_2) \\ & + m_1gl_1(\cos q_1 - 1) + m_2g(l_1(\cos q_1 - 1) + l_2(\cos q_2 - 1)) \end{aligned} \quad (1.4)$$

Après simplification on trouve :

$$m_2l_1l_2 \cos(q_2 - q_1)\ddot{q}_1 + m_2l_2^2\ddot{q}_2 + m_2l_1l_2 \sin(q_2 - q_1)\dot{q}_1^2 - m_2gl_2 \sin q_2 = 0 \quad (1.5)$$

$$(m_1 + m_2)l_1^2\ddot{q}_1 + m_2l_1l_2 \cos(q_2 - q_1)\ddot{q}_2 + m_2l_1l_2 \sin(q_2 - q_1)\dot{q}_2^2 - (m_1 + m_2)gl_1 \sin q_1 = \tau \quad (1.6)$$

1.4.2 L'Acrobot

L'acrobot est similaire au pendubot à la différence que c'est l'articulation joignant les 2 tiges qui est actionnée par un couple τ (voir figure 1.5). les équations dynamiques du mouvement sont données par :

$$m_2l_1l_2 \cos(q_2 - q_1)\ddot{q}_1 + m_2l_2^2\ddot{q}_2 + m_2l_1l_2 \sin(q_2 - q_1)\dot{q}_1^2 - m_2gl_2 \sin q_2 = \tau \quad (1.7)$$

$$(m_1 + m_2)l_1^2\ddot{q}_1 + m_2l_1l_2 \cos(q_2 - q_1)\ddot{q}_2 + m_2l_1l_2 \sin(q_2 - q_1)\dot{q}_2^2 - (m_1 + m_2)gl_1 \sin q_1 = 0 \quad (1.8)$$

où m_1 , l_1 , m_2 , et l_2 sont respectivement les masses et longueurs des tiges 1 et 2 ; τ est le couple de commande.

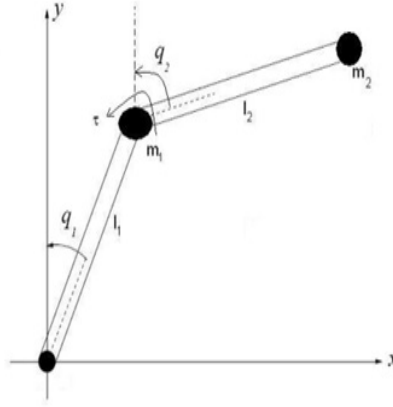


FIGURE 1.5 – L'Acrobot en coordonnées généralisés.

L'acrobot admet aussi une infinité de points d'équilibre stables et instables. L'ensemble des points d'équilibre instables correspond aux positions où la verticale passant par le centre de gravité du système global passe aussi par l'axe de rotation de la tige 1. Les positions d'équilibres stables sont identiques aux points instables, la seule différence étant que le centre de gravité est dans le demi-plan inférieur. Un simple calcul des moments d'ordre 1 des 2 masses ponctuelles m_1 et m_2 par rapport à l'axe de rotation de la tige numéro 1 donne $m_1 l_1 \sin(q_1) = m_2 l_2 \sin(q_2)$; correspondant aux points d'équilibres.[4]

1.4.3 Le pendule de Furuta :

Le pendule de Furuta a été conçu par K. Furuta pour contrer l'handicap de la course limitée (en translation horizontale) du pendule inversé classique. Le bras actionné en rotation dans le plan horizontal permet une course infinie (voir figure 1.6)[4], ce qui facilite la conception du contrôle. A l'autre extrémité du bras vient s'ajouter un pendule libre en rotation dans le plan vertical orthogonal au bras. Le lagrangien est donné par :

$$\mathcal{L} = \frac{1}{2} J_p \dot{q}_2^2 + \frac{1}{2} (J_b \sin^2 q_2) \dot{q}_1^2 - m_p l r \dot{q}_1 \dot{q}_2 \cos q_2 - m_p g r (\cos q_2 - 1) \quad (1.9)$$

Les équations dynamiques du mouvement sont données par :

$$-m_p l r \cos q_2 \ddot{q}_1 + J_p \ddot{q}_2 - J_p \sin q_2 \cos q_2 \dot{q}_1^2 - m_p g r \sin q_2 = 0 \quad (1.10)$$

$$(J_b + J_p \sin^2 q_2) \ddot{q}_1 - m_p l r \cos q_2 \ddot{q}_2 + 2 J_p \sin q_2 \dot{q}_2 + m_p l r \sin q_2 \dot{q}_2^2 = \tau \quad (1.11)$$

Où J_p et J_b représentent les moments d'inertie du pendule et du bras respectivement, m_p la masse du pendule, l et r les longueurs respectives du bras et du pendule. Les positions d'équilibre stable (instables) correspondent à $\dot{q}_1 = \dot{q}_2 = 0$, $q_1 = cte$, ($\dot{q}_1 = \dot{q}_2 = 0$; $q_2 = \pi$)

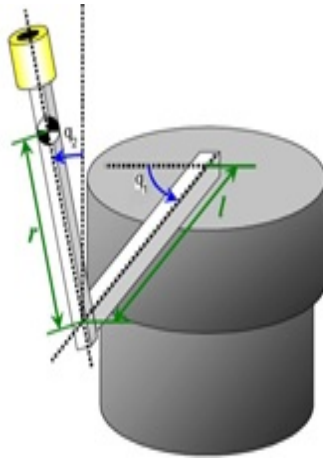


FIGURE 1.6 – Le pendule de Furuta en coordonnées généralisées

1.4.4 Le pendule à roue inertielle

Le pendule à roue inertielle, représenté sur la figure 1.7, est constitué d'un pendule libre en rotation autour d'un axe lié au sol, l'autre extrémité du pendule étant reliée à un disque actionné qui ne peut que tourner[4]. Le lagrangien est donné par :

$$L = \frac{1}{2}(m_1 + m_2)l^2\dot{q}_1^2 + \frac{1}{2}J\dot{q}_2^2 + (m_1 + m_2)gl(\cos q_1 - 1) \quad (1.12)$$

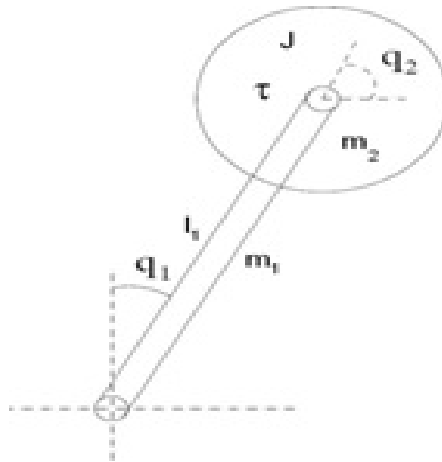


FIGURE 1.7 – Le pendule à roue inertielle en coordonnées généralisées.

La dynamique est décrite par :

$$[(m_1 + m_2) + l^2 + J]\ddot{q}_1 + J\ddot{q}_2 - (m_1 + m_2)gl \sin q_1 = 0 \quad (1.13)$$

$$J\ddot{q}_1 + J\ddot{q}_2 = \tau \quad (1.14)$$

où m_1 , m_2 , l , et J représentent respectivement la masse du pendule, la masse du disque, la longueur du pendule et l'inertie du disque. Les points d'équilibre stables (respectivement, instables) correspondent à $q_1 = \dot{q}_1 = \dot{q}_2 = 0$ (respectivement, $\dot{q}_1 = \dot{q}_2 = 0$; $q_1 = \pi$).

1.4.5 Le pendule inversé sur chariot

Nous présentons ici le pendule inversé sur chariot (également appelé système chariot pendule ou cart-pendulum system), en rentrant plus dans les détails que pour les systèmes précédents.

Le pendule inversé est un système classique très intéressant et largement étudié dans la communauté automatique, vu sa nature non linéaire et instable. Il a toujours constitué un défi intéressant pour le contrôle et a servi à la compréhension des notions de l'automatique comme à l'élaboration des lois de commande. Par ailleurs, son principe se retrouve dans plusieurs applications de véhicules légers (par exemple le Segway).

Mais qu'est-ce qu'un pendule inversé ?

Souvenez-vous quand vous étiez petit et que vous essayiez de tenir en équilibre une tige sur votre doigt. Vous deviez en permanence ajuster la position de votre main pour stabiliser la tige. Le pendule inversé fait la même chose, la seule différence étant que son support (le chariot) se translate dans une seule direction, alors que le doigt se déplace dans toutes les directions.

Depuis 1950, les pendules inversés sont des plates-formes d'expérimentation classiques dans les laboratoires d'automatique. Ils ont été utilisés pour illustrer des idées de commande linéaire comme la stabilisation des systèmes instables. Vue leur nature non linéaire, les pendules sont aussi utilisés pour illustrer des idées émergent du contrôle non linéaire. Des exemples typiques sont la stabilisation par retour de sortie[3].

1.5 L'intérêt de l'étude d'un pendule inversé

L'étude de pendule inversé est idéale pour les étudiants comme nous en Automatique et en Robotique qui font les laboratoires de recherches parce qu'ils font intervenir beaucoup de notions intéressantes pour eux : la programmation, l'automatisation, la mécanique et l'électronique. Le bébé après son premier pas de marche marche comme un pendule inversé dont les deux axes de rotation fondamentaux sont les hanches et les chevilles. Lorsqu'il est en position de debout, ses articulations de corps travaillent sans s'arrêter pour le maintenir en équilibre. [5]

Dans le même ordre d'idée, la robotique utilise ce genre de concept. On voit apparaître des moyens de locomotion dotés de deux roues montées position debout (voir figure(1.9), (1.12)) penchant en arrière. Le système est le même que le pendule inversé. La stabilité est aussi assurée par des gyroscopes mais nous n'entrerons pas dans ce genre de détails.[3] On représente dans les figures (1.13) pendule inversé :



FIGURE 1.8 – Scooter intelligent (Segway) à une et deux roues, le robot (MIP one)

Mentionné dans cette section sont quelques applications de ce concept au monde réel dans les domaines variés.

1.6 Applications des systèmes sous actionnée

1.6.1 Domaine de Robotique

La robotique utilise ce type de concept dans leur principe de commande. Le robot auto-balancé est constitué d'une plateforme solidaire de deux roues. L'axe des roues est perpendiculaire à l'axe de déplacement du robot et le principe de contrôle de position est basé sur le pendule inversé. La figure 1.10 à gauche est une image d'un robot de voiture dansante qui s'agit d'un jouet pour les enfants qui utilise le principe du pendule inversé pour se tenir debout, exposant ainsi un spectacle intéressant. A droite nous avons un robot auto balancé et assez stable pour servir les boissons comme un serveur humain.[5]



FIGURE 1.9 – Exemples de robots inversés.

Nbot (figure 1.11 à gauche) est un robot d'équilibrage à deux roues construit par David.P Anderson, ce robot utilise un capteur inertielle disponible dans le commerce et positionne l'information à partir du codeur du moteur pour équilibrer le système. Steven Hassenplug a construit avec succès un robot d'équilibrage appelé Legway (figure 1.11 à droite) en utilisant le kit de robotisation LEGO Mindstorms. Deux capteurs de détecteur de proximité électro-optique (EOPD) ont été utilisés pour fournir l'angle d'inclinaison du robot au contrôleur qui est programmé dans BrickOS, un langage de programmation similaire à C/C ++.



FIGURE 1.10 – a) Nbot et b) Legway [31]

1.6.2 Domaine de la médecine

Les spécialistes qui travaillent dans le domaine de réalisation de prothèses pour les hanches (remplacement chirurgical d'un organe, la pièce ou l'appareil de remplacement) sont amenés à utiliser un pendule double inversé pour calculer l'ensemble des contraintes qui sont soumises à la prothèse comme le montre la figure suivante. Le premier pendule est articulé à la cheville et représente les membres inférieurs considérés groupés. Le second pendule est articulé à la hanche et représente la partie supérieure du corps. On accélère en se penchant en avant et on a même que le pendule inversé voir figure 1.12.[6]



FIGURE 1.11 – Le corps de l'être humain vu comme double pendule

1.6.3 Appareil de transport personnel Segway

Montré en (Figure 1.13) ci-dessous est un exemple pratique d'un pendule inversé, où la motion des roues est commandée par un ordinateur avec le but de maintenir le corps du véhicule dans une position verticale. Quand le cycliste incline en avant, il pousse le contrôleur de tourner les roues avec l'accélération nécessaire pour rester en balance.

Dee Kamen qui détient plus de 150 brevets américains et étrangers liés aux dispositifs médicaux, aux systèmes de contrôle climatique et à la conception d'hélicoptères, a inventé le «SEGWAY HT». Cette innovation utilise cinq gyroscopes et une collection d'autres capteurs d'inclinaison pour se maintenir debout. Seuls trois gyroscopes sont nécessaires pour l'ensemble du système, les capteurs supplémentaires sont inclus comme mesure de sécurité. [4]



FIGURE 1.12 – Le Segway HT [31]

1.6.4 Le domaine de l'aérospatiale :

Dans ce domaine l'étude des pendules inversés a une grande importance, par exemple pour commander et stabiliser l'attitude du satellite, le lancement des fusées...etc.[7]

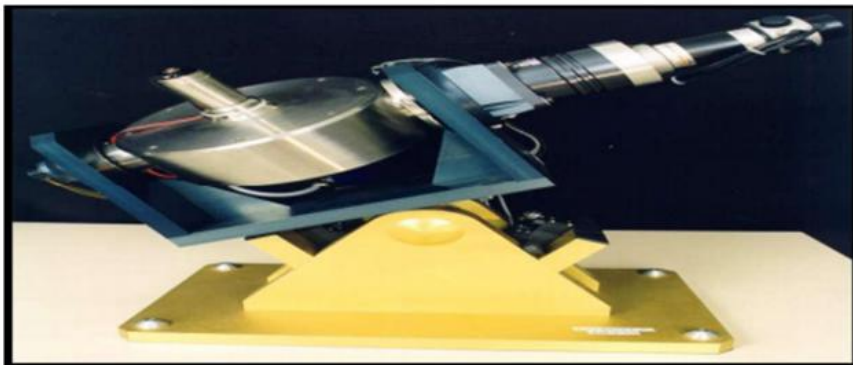


FIGURE 1.13 – Gyroscopique

1.6.5 Domaine de loisir

Dans les parcs d'attractions on trouve les tours qui utilisent le principe du pendule inversé. La machine en figure 1.15 bascule vers le haut autour d'un axe de sorte qu'il agit comme un pendule inversé mais dans sa première phase de mouvement il agit comme un pendule normal.



FIGURE 1.14 – Tour d’amusement (Ciseaux)

1.7 Les différents types des robots mobiles

Il est possible de classer 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 classer, puisque leur mobilité est l’une de leurs caractéristiques essentielles, selon leur mode de locomotion.[8]

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).



FIGURE 1.15 – Exemples de pendules inversés

1.7.1 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 compliquées (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 militaires 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.

1.7.2 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 Massachusetts 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 où les progrès dans ce segment sont assez significatifs. 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 [8].
- 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.

1.7.3 Robot auto-balancé (pendule inversé)

C'est le cas d'application de notre projet, Le robot auto-balancé est constitué d'une plateforme solidaire de deux roues. L'axe des roues est perpendiculaire à l'axe de déplacement du robot. 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 α , va provoquer le déplacement du robot, de telle sorte que l'angle redevienne nul. D'où le nom de " Robot Auto-Balancé"

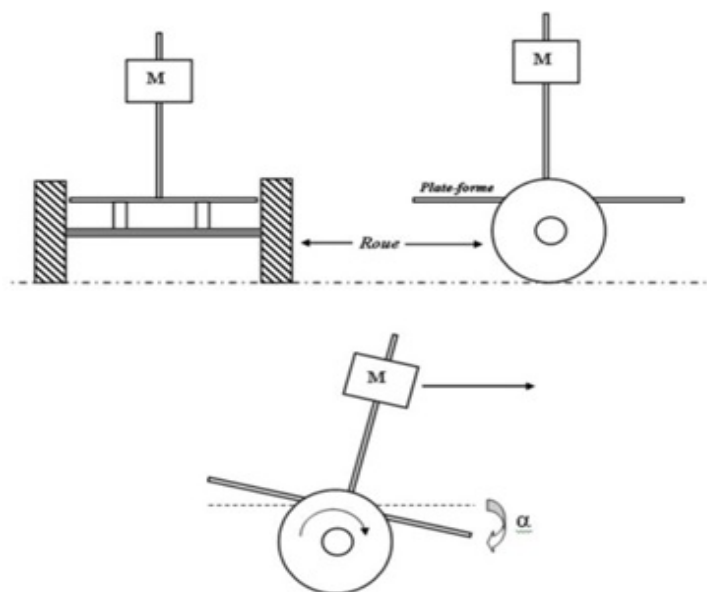


FIGURE 1.16 – Robot Auto-Balancé

Si le dispositif réagit suffisamment vite, l'angle est en permanence proche de zéro. La plateforme 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.

1.8 Principe de fonctionnement d'un pendule inverse

Le principe est simple théoriquement : quand le pendule penche vers l'un des cotés (gauche ou droit), le chariot doit le rattraper en effectuant un mouvement vers ce côté.[9] Le problème se pose dans le réglage de l'intensité et la forme de réaction que doit subir le chariot en fonction de l'angle que le pendule fait avec la verticale. Pour cela, on est amené à prendre en compte plusieurs aspects ou particularités dans notre étude, soit :

- L'instabilité du système ;
- La non linéarité ;
- Modèle de connaissance incomplet ;
- Un seul actionneur ;
- Deux grandeurs à asservir.

L'ensemble du chariot-pendule a deux degrés de liberté qui sont représentés par deux coordonnées généralisées, pour le déplacement horizontale du chariot, pour la rotation du pendule. La direction positive d'est le sens à droite en mètre et celui de l'angle est le sens des aiguilles d'une montre en radian.

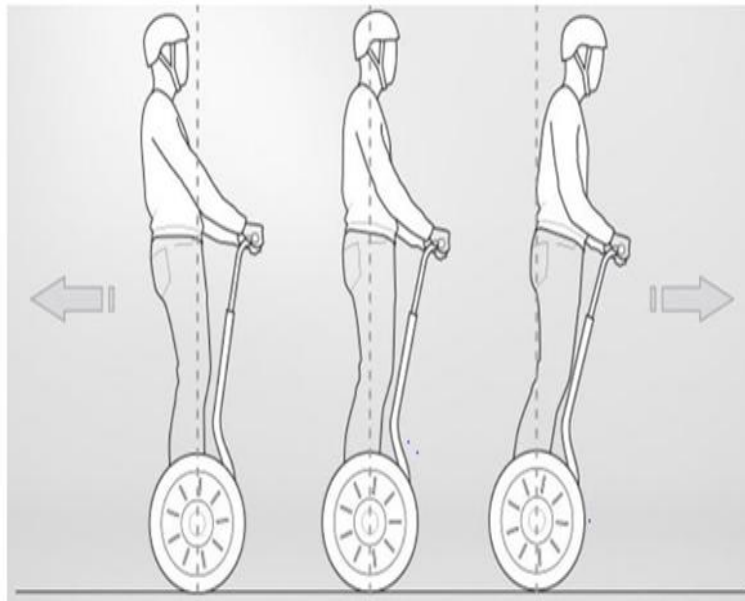


FIGURE 1.17 – Exemple sur l'équilibre à base du pendule inversé

1.9 Conclusion :

L'objectif de notre étude consiste principalement à l'étude et réalisation d'un robot mobile à pendule inverse. Pour cela nous avons fait une généralité sur les systèmes sous actionnée et différents robots mobiles, ainsi que nous avons définie et énoncée le robot auto-balancé qu'il est le cas d'application de notre étude.

Modélisation d'un robot auto balancier

Sommaire

2.1	Introduction	18
2.2	Modélisation mathématique	18
2.2.1	Définition	18
2.3	Modèle linéaire d'un moteur à courant continu (DC)	19
2.4	Modélisation mathématique du pendule inversé	21
2.5	Modélisation du système à deux degrés de liberté	22
2.5.1	Énergie cinétique du système en mouvement	22
2.5.2	Énergie potentielle du système	23
2.5.3	Équation de Lagrange	23
2.6	Linéarisation du modèle autour des points d'équilibre	24
2.6.1	Position d'équilibre instable ($\theta = 0$)	24
2.6.2	Position d'équilibre stable ($\theta = \pi$)	25
2.7	Modèle dynamique pour un pendule inversé à deux roues	25
2.8	Conclusion	30

2.1 Introduction

Dans le domaine scientifique, l'automatique a souvent recours à des cas d'études particuliers, qui sont représentatifs de grandes classes d'applications, et dont le caractère spectaculaire est confirmé. De plus, avec l'expérience, la connaissance de ces cas s'est affinée et ils fournissent aujourd'hui une base idéale pour comparer de façon valable les avantages et les inconvénients d'approches différentes[10]. Le pendule inversé est un de ces cas-types.

2.2 Modélisation mathématique

2.2.1 Définition

Le procédé par lequel nous utilisons des expressions mathématiques pour décrire une situation quantitative réelle s'appelle la modélisation. Modéliser consiste à écrire en notation mathématique ce qui est exprimé d'abord en mots en faisant intervenir des variables au besoin[1].

2.3 Modèle linéaire d'un moteur à courant continu (DC)

Le robot est mobile grâce à deux moteurs à courant continu (DC). Dans cette section, le modèle d'espace d'état du moteur à courant continu est décrit. Ce modèle est ensuite utilisé dans le modèle dynamique du robot équilibriste pour fournir une relation entre la tension d'entrée aux moteurs et le couple de contrôle nécessaire pour équilibrer le robot [11][12].

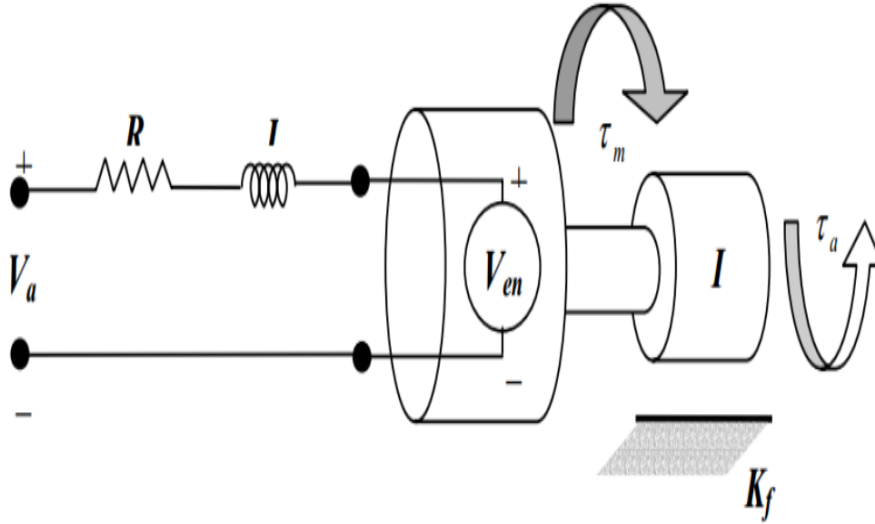


FIGURE 2.1 – Schémas d'un moteur à courant continu

2.3.0.1 Les équations électriques

Équation de l'induit

$$v(t) = Ri(t) + l \frac{di(t)}{dt} \quad (2.1)$$

Équation de la femc :

$$l(t) = K_b \Omega_m(t) \quad (2.2)$$

2.3.0.2 Les équations mécaniques :

Le couple mécanique est donné par la formule :

$$C_m(t) = J_m \frac{d\Omega_m(t)}{dt} + C_r(t) + f_m \Omega_m(t) \quad (2.3)$$

Équation de couple :

$$C_m(t) = K_m i(t) \quad (2.4)$$

La Figure (2.1) illustre un modèle linéaire efficace pour un moteur à courant continu. Lorsqu'une tension est appliquée aux bornes du moteur, un courant « i » est généré dans l'armature du moteur. Le moteur produit un couple, qui est proportionnel au courant [13]. Cette relation peut être exprimée par :

$$\tau_m = K_m i(t) \quad (2.5)$$

Une paire de résistance- inductance en série avec une tension V_a , peut être utilisée pour modéliser le circuit électrique du moteur. Une force électromotrice est produite parce que les bobines du

moteur sont traversées par un champ magnétique. La tension produite peut être approximée par une fonction linéaire de la vitesse de l'arbre :

$$V_e = K_e \omega \quad (2.6)$$

À ce point, une équation différentielle linéaire pour le circuit électrique du moteur à courant continu est déduite en utilisant la loi des tensions de « Kirchhoff ». Cette loi indique que la somme de toutes les tensions dans le circuit doit être égale à zéro. Pour le moteur à courant continu, cela peut s'écrire comme :

$$V_a - Ri - L \frac{di}{dt} - V_e = 0 \quad (2.7)$$

Pour décrire l'équation de mouvement pour le moteur, le frottement sur l'arbre du moteur est approximée par une fonction linéaire de la vitesse de l'arbre. L'approximation est effectuée pour que le frottement sur l'arbre du moteur k_f soit une fonction linéaire de la vitesse de l'arbre. La loi de Newton sur le mouvement indique que la somme de tous les couples produit sur l'arbre est liée linéairement à l'accélération de l'arbre par la force d'inertie de l'induit. Soit l'équation :

$$\sum M = \tau_m - K_f \omega - \tau_a = I_R \dot{\omega} \quad (2.8)$$

En introduisant les équations (2.1) et (2.2) dans les équations (2.3) et (2.4) et en réarrangeant les termes des dérivés, on obtient les deux équations fondamentales suivantes et qui résument le mouvement du moteur.

$$\frac{di}{dt} = \frac{V_a}{L} - \frac{R}{L}i - \frac{K_e}{L}\omega \quad (2.9)$$

$$\frac{d\omega}{dt} = \frac{k_m}{I_R} - \frac{K_f}{I_R}i - \frac{\tau_a}{I_R} \quad (2.10)$$

Les deux équations sont des fonctions linéaires du courant et de la vitesse et elles contiennent des dérivés du premier ordre. Un modèle simplifié du moteur à courant continu est suffisant pour l'équilibre du robot. Pour cette raison, l'inductance du moteur est prise nulle et les frottements du moteur sont négligeables. Par conséquent, (2.5) et (2.6) deviennent :

$$i = \frac{V_a}{R} - \frac{K_e}{R}\omega \quad (2.11)$$

$$\frac{d\omega}{dt} = \frac{k_m}{I_R} - \frac{\tau_a}{I_R} \quad (2.12)$$

En remplaçant l'équation (2.7) dans l'équation (2.8), une relation approximative du moteur à courant continu est obtenue en fonction de la vitesse actuelle du moteur, la tension appliquée et le couple comme :

$$\frac{d\omega}{dt} = \frac{k_m k_a}{I_R R} - \frac{k_m V_e}{I_R R} \omega - \frac{\tau_a}{I_R} \quad (2.13)$$

Étant donné que l'inductance du moteur est négligée, le courant à travers les enroulements n'est pas pris en compte dans l'équation de mouvement du moteur. Le courant atteint son état constant par rapport à la vitesse de l'arbre, qui va prendre beaucoup plus de temps pour arriver à la vitesse finale lors d'un changement de la tension d'entrée. La dynamique du moteur peut être représentée par un modèle d'espace d'état[8]. C'est un système d'équations différentielles de premier ordre avec les paramètres position θ , et vitesse ω , qui représentent uniquement le fonctionnement. Les entrées du moteur sont alors la tension appliquée et le couple appliqué. $\dot{x} = Ax + Bv$.

$$\begin{aligned} \begin{bmatrix} \dot{\theta} \\ \dot{\omega} \end{bmatrix} &= \begin{bmatrix} 0 & 1 \\ \frac{K_m}{I_{RR}} & \frac{-1}{I_R} \end{bmatrix} \begin{bmatrix} V_a \\ \tau_a \end{bmatrix} \\ y &= \begin{bmatrix} 0 & 1 \end{bmatrix} \begin{bmatrix} \theta \\ \omega \end{bmatrix} + \begin{bmatrix} 0 & 0 \end{bmatrix} \begin{bmatrix} V_a \\ \tau_a \end{bmatrix} \end{aligned} \quad (2.14)$$

2.4 Modélisation mathématique du pendule inversé

Le système du pendule inversé est un exemple couramment utilisé dans les manuels de systèmes de contrôle et la littérature de recherche. Sa popularité découle en partie du fait qu'elle est instable sans contrôle, c'est-à-dire que le pendule tombera simplement si le chariot n'est pas déplacé pour l'équilibrer. En outre, la dynamique du système est non-linéaire. L'objectif de la commande du système est d'équilibrer le pendule inversé en appliquant une force au chariot sur lequel le pendule est fixé.

Dans ce cas, nous considérons un problème bidimensionnel où le pendule est forcé de se déplacer dans le plan vertical représenté sur la figure ci-dessous. Pour ce système, l'entrée de commande est la force F qui déplace le chariot horizontalement et les sorties sont la position angulaire du pendule θ et la position horizontale du chariot x .

Soit :

- m : masse du pendule
- b : frottements de déplacement du chariot
- M : masse du chariot
- $x(t)$: position du chariot
- l : demi longueur du pendule
- $\theta(t)$: l'angle du pendule
- $F(t)$: force exercée sur le chariot
- g : intensité de pesanteur

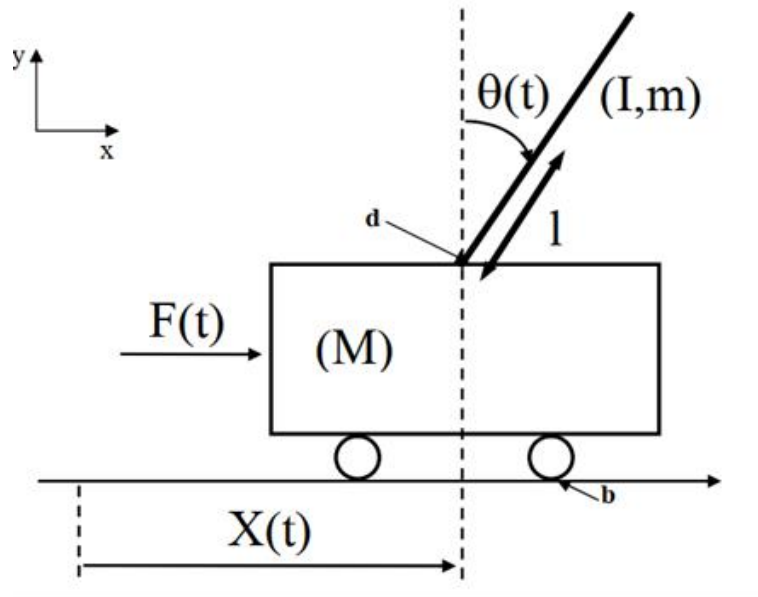


FIGURE 2.2 – Schéma de l'ensemble chariot et pendule inversé

Il existe deux méthodes de détermination des équations du mouvement d'un pendule inversé :

Celle de la loi fondamentale de la dynamique (LFD) de Newton qui est basée sur le concept de force, et celle de formalisme d'Euler-Lagrange qui est basée sur le principe de la conservation d'énergie mécanique. Dans ce travail on s'intéresse au formalisme d'Euler-Lagrange dans lequel le lagrangien (L) est défini comme une différence entre l'énergie cinétique (T) et l'énergie potentielle (V) du système :

$$L = T - V \quad (2.15)$$

2.5 Modélisation du système à deux degrés de liberté

2.5.1 Énergie cinétique du système en mouvement

L'énergie cinétique du chariot est donnée par l'équation :

$$T_M = \frac{1}{2}M\dot{x}^2 \quad (2.16)$$

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

$$T_m = \frac{1}{2}mv_c^2 + \frac{1}{2}I\omega^2 \quad (2.17)$$

avec :

v_c : La vitesse de centre de gravité du pendule ;

θ : La vitesse angulaire du pendule ;

I : Le moment d'inertie du pendule.

La position du centre de gravité du pendule, notée \vec{r}_c à partir de ces coordonnées, est donnée par :

$$r_c = (x + l\sin\theta)\hat{i} + l\cos\theta\hat{j} \quad (2.18)$$

i, j : étant les vecteurs unitaires du repère x, y .

D'où la vitesse du centre de gravité du pendule est :

$$v_c = \frac{dr_c}{dt} = (\dot{x} + l\cos\theta\dot{\theta})\hat{i} + l\sin\theta\dot{\theta}\hat{j} \quad (2.19)$$

En substituant les équations (2.18) et (2.19) dans l'équation (2.17) on trouve :

$$T_m = \frac{1}{2}m(\dot{x}^2 + 2\dot{x}l\cos\theta\dot{\theta} + l^2\cos^2\theta\dot{\theta}^2 + l^2\sin^2\theta\dot{\theta}^2) + \frac{1}{2}I\dot{\theta}^2 \quad (2.20)$$

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

Avec simplification l'équation (2.19), l'énergie cinétique prend alors l'expression :

$$T_m = \frac{1}{2}m(\dot{x}^2 + 2\dot{x}l\cos\theta\dot{\theta} + l^2\dot{\theta}^2) + \frac{1}{2}I\dot{\theta}^2 \quad (2.21)$$

L'énergie cinétique de l'ensemble chariot (2.16) et pendule (2.20) est exprimée par :

$$T = T_M + T_m$$

$$T = \frac{1}{2}M\dot{x}^2 + \frac{1}{2}m(\dot{x}^2 + 2\dot{x}l\cos\theta\dot{\theta} + l^2\dot{\theta}^2) + \frac{1}{2}I\dot{\theta}^2 \quad (2.22)$$

2.5.2 Énergie potentielle du système

L'énergie potentielle du centre de gravité de la barre est :

$$V = mgl\cos\theta \quad (2.23)$$

2.5.3 Équation de Lagrange

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{\zeta}}\right) - \frac{\partial L}{\partial \zeta} + \frac{\partial D_f}{\partial \dot{\zeta}} = F_j \quad (2.24)$$

ou :

ζ : Degrés de liberté, dans ce cas $x(t)$ et $\theta(t)$.

D_f : Énergie dissipée par frottement.

F_j : Force généralisée dans le sens du degré de liberté ζ_j .

Le lagrangien du système est la différence entre son énergie cinétique et son énergie potentielle :

$$L = T - V$$

En utilisant l'expression (2.22) et (2.23) le lagrangien peut être écrit :

$$L = \frac{1}{2}M\dot{x}^2 + \frac{1}{2}m(\dot{x}^2 + 2\dot{x}l\cos\theta\dot{\theta} + l^2\dot{\theta}^2) + \frac{1}{2}I\dot{\theta}^2 - mgl\cos\theta \quad (2.25)$$

2.5.3.1 L'équation de Lagrange pour le degré de liberté $\zeta(t) = x(t)$

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{x}}\right) - \frac{\partial L}{\partial x} = F - b\dot{x} \quad (2.26)$$

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

$$\frac{d}{dt}(M\dot{x} + m\dot{x} + ml\cos\theta\dot{\theta}) - 0 = F - b\dot{x} \quad (2.27)$$

Donc la **première équation** de Lagrange :

$$(M + m)\ddot{x} + ml\cos\theta\ddot{\theta} - ml\sin\theta\dot{\theta}^2 = F - b\dot{x} \quad (2.28)$$

2.5.3.2 L'équation de Lagrange pour le degré de liberté $\zeta(t) = \theta(t)$

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{\theta}}\right) - \frac{\partial L}{\partial \theta} = -d\dot{\theta} \quad (2.29)$$

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

$$\frac{d}{dt}(-ml\dot{x}\cos\theta + ml^2\dot{\theta} + I\dot{\theta}) - (ml\dot{x}\sin\theta\dot{\theta} - mgl\sin\theta) = -d\dot{\theta} \quad (2.30)$$

Donc la **deuxième équation** de Lagrange :

$$(ml^2 + I)\ddot{\theta} + ml\ddot{x}\cos\theta + ml\dot{x}\sin\theta\dot{\theta} - ml\dot{x}\sin\theta\dot{\theta} - mgl\sin\theta = -d\dot{\theta} \quad (2.31)$$

Le modèle de connaissance du système chariot pendule est donné par le système d'équations suivant :

$$\begin{cases} (M + m)\ddot{x} + b\dot{x} + ml\cos\theta\ddot{\theta} - ml\sin\theta\dot{\theta}^2 = F \\ ml\ddot{x}\cos\theta + (ml^2 + I)\ddot{\theta} + d\dot{\theta} - mgl\sin\theta = 0 \end{cases} \quad (2.32)$$

2.6 Linéarisation du modèle autour des points d'équilibre

Avant de procéder à l'analyse du modèle de système, on linéarise les équations différentielles (2.32). Il a deux points d'équilibre : ($\theta = 0$) point d'équilibre instable et ($\theta = \pi$) point d'équilibre instable. Pour des petites variations de θ autour du point d'équilibre θ_0

$$\begin{cases} \theta = \theta_0 + \varepsilon \\ \dot{\theta} = \dot{\varepsilon} \end{cases} \quad (2.33)$$

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

$$f(\theta) \approx f(\theta_0) + \varepsilon \frac{df}{d\theta} \Big|_{\theta_0} \quad (2.34)$$

Et les termes du haut ordre sont négligés

$$\dot{\varepsilon} \approx 0 \quad (2.35)$$

2.6.1 Position d'équilibre instable ($\theta = 0$)

Si on se limite aux petites variations de θ autour du point de fonctionnement $\theta_0 = 0$ correspondant à la position verticale de la barre. Pour ($\theta = 0$), le développement limite du premier ordre des équations (2.33) et (2.34) est

$$\begin{cases} \cos\theta \approx \cos(0) - \theta \sin 0 = 1 \\ \sin\theta \approx \sin(0) + \theta \cos 0 = \theta \\ \dot{\theta}^2 = 0 \end{cases} \quad (2.36)$$

En substituant ces linéarisations dans le système d'équation (2.32) et en négligeant les termes du haut ordre, on trouve le système d'équation linéarisé suivant :

$$\begin{cases} (M + m)\ddot{x} + b\dot{x} + ml\ddot{\theta} = F \\ ml\ddot{x} + (ml^2 + I)\ddot{\theta} + d\dot{\theta} - mgl\theta = 0 \end{cases} \quad (2.37)$$

En appliquant la transformée de Laplace sur le système d'équation linéarisé ci-dessus, on trouve :

$$\begin{cases} (M + m)s^2 X(s) + bsX(s) + mls^2 \Theta(s) = F(s) \\ mls^2 X(s) + (ml^2 + I)s^2 \Theta(s) + ds\Theta(s) - mgl\Theta(s) = 0 \end{cases} \quad (2.38)$$

ou : $X(s) = \mathcal{L}x(t)$ et $\Theta(s) = \mathcal{L}(\theta(t))$

$$\begin{cases} G_{char}(s) = \frac{X(s)}{F(s)} = \frac{a_2 s^2 + a_1 s + a_0}{b_4 s^4 + b_3 s^3 + b_2 s^2 + b_1} \\ G_{pend}(s) = \frac{\Theta(s)}{F(s)} = \frac{c_2 s}{b_4 s^3 + b_3 s^2 + b_2 s + b_1} \end{cases} \quad (2.39)$$

avec :

$$\begin{aligned} a_2 &= ml^2 + I \\ a_1 &= d \\ a_0 &= -mgl \\ b_4 &= (M + m)(ml^2 + I) - m^2 l^2 \\ b_3 &= (M + m)d + (ml^2 + I)b \end{aligned}$$

$$\begin{aligned} b_2 &= -(M+m)mgl + db \\ b_1 &= -mglb \\ c_2 &= -ml \end{aligned}$$

Les fonctions de transfert respectivement de la position du chariot et de la rotation du pendule sont :

$$\begin{cases} G_{char}(s) = \frac{(ml^2+I)s^2+(d)s-mgl}{((M+m)(ml^2+I)-m^2l^2)s^4+((M+m)d+(ml^2+I)b)s^3-((M+m)mgl+db)s^2-mglbs} \\ G_{pend}(s) = \frac{-mls}{((M+m)(ml^2+I)-m^2l^2)s^3+((M+m)d+(ml^2+I)b)s^2+(-(M+m)mgl+db)s-mglb} \end{cases} \quad (2.40)$$

2.6.2 Position d'équilibre stable ($\theta = \pi$)

Pour ($\theta = \pi$), le développement limite du premier ordre des équations (2.33) et (2.34) est :

$$\begin{cases} \cos\theta \approx \cos(\pi) + (\pi - \theta) - \sin\pi = -1 \\ \sin\theta \approx \sin(\pi) + (\pi - \theta)\cos\pi = -\theta \\ \dot{\theta}^2 = 0 \end{cases} \quad (2.41)$$

La substitution de ces linéarisations dans le système d'équation (2.32) et la négligence des termes du haut ordre, permettent de trouver le système d'équation linéarisé suivant :

$$\begin{cases} (M+m)\ddot{x} + b\dot{x} - ml\ddot{\theta} = F \\ -ml\ddot{x} + (ml^2+I)\ddot{\theta} + d\dot{\theta} + mgl\theta = 0 \end{cases} \quad (2.42)$$

Enfin l'application de la transformée de Laplace sur le système d'équation linéarisé ci dessus, se traduit par :

$$\begin{cases} (M+m)s^2X(s) + bsX(s) - mls^2\Theta(s) = F(s) \\ -mls^2X(s) + (ml^2+I)s^2\Theta(s) + ds\Theta(s) + mgl\Theta(s) = 0 \end{cases} \quad (2.43)$$

En substituant pour éliminer $X(s)$ ou $\Theta(s)$ dans (2.43), on trouve les deux fonctions de transfert suivantes :

$$\begin{cases} G_{char}(s) = \frac{X(s)}{F(s)} = \frac{a_2s^2+a_1s+a_0}{b_4s^4+b_3s^3+b_2s^2+b_1s} \\ G_{pend}(s) = \frac{\Theta(s)}{F(s)} = \frac{c_2s}{b_4s^3+b_3s^2+b_2s+b_1} \end{cases} \quad (2.44)$$

avec :

$$\begin{aligned} a_2 &= ml^2 + I \\ a_1 &= d \\ a_0 &= mgl \\ b_4 &= (M+m)(ml^2+I) - m^2l^2 \\ b_3 &= (M+m)d + (ml^2+I)b \\ b_2 &= (M+m)mgl + db \\ b_1 &= mglb \\ c_2 &= ml \end{aligned}$$

2.7 Modèle dynamique pour un pendule inversé à deux roues

Le Pendule inversé à deux roues, a un comportement similaire avec un pendule sur un chariot[14][15]. La dynamique des roues et le pendule sont analysés séparément au début, mais

cela finira par deux équations de mouvement qui décrivent complètement le comportement du robot équilibriste. Comme le comportement du robot peut être influencé par des perturbations ainsi que le couple de rotation de moteur, le modèle mathématique devra tenir compte de ces forces [12]. Premièrement on obtient les équations de mouvement associé aux roues gauche et droite [13]. La figure (2.4) ci-dessous montre le diagramme de corps libre pour les deux roues. Étant donné que les équations des roues gauche et droite sont tout à fait analogues, seule l'équation.

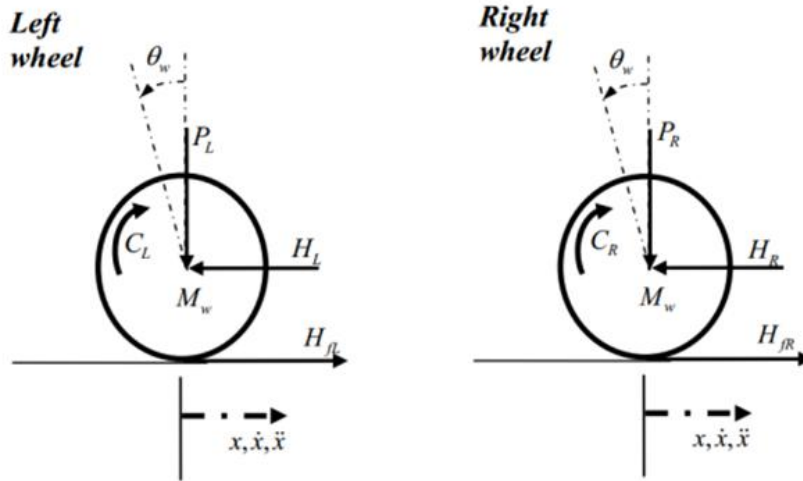


FIGURE 2.3 – Schémas des corps libre pour les deux roues

Utilisant la loi de mouvement de Newton, la somme des forces sur la direction x est :

$$\sum F_x = Ma \quad (2.45)$$

$$M_w \ddot{x} = H_{fr} - H_R \quad (2.46)$$

La Somme des forces autour du centre de la roue donne :

$$\sum M_0 = I\alpha \quad (2.47)$$

$$I_w \ddot{\theta} = C_R - H_{fr}r \quad (2.48)$$

À partir de la dynamique du moteur à courant continu, le couple du moteur peut être exprimé sous la forme :

$$\tau_m = I_R \frac{d\omega}{dt} + \tau_a \quad (2.49)$$

En réarrangeant l'équation et remplaçant par les paramètres du moteur à courant continu, le couple de sortie aux roues est atteint :

$$C = I_R \frac{d\omega}{dt} = \frac{-K_m K_e}{R} \dot{\theta}_\omega + \frac{K_m}{R} V_a \quad (2.50)$$

Par conséquent, l'équation (2.48) devient :

$$I_w \ddot{\theta} = \frac{-K_m K_e}{R} \dot{\theta}_\omega + \frac{K_m}{R} V_a - H_{fr}r \quad (2.51)$$

Ainsi

L'équation (2.50) est remplacée dans (2.46) pour obtenir l'équation des roues gauche et droite.

2.7.0.1 Pour la roue gauche

$$M_\omega \ddot{x} = \frac{-K_m K_e}{Rr} \dot{\theta}_\omega + \frac{K_m}{R} V_a - \frac{I_\omega}{r} \ddot{\theta}_\omega - H_L \quad (2.52)$$

2.7.0.2 Pour la roue droite

$$M_\omega \ddot{x} = \frac{-K_m K_e}{Rr} \dot{\theta}_\omega + \frac{K_m}{R} V_a - \frac{I_\omega}{r} \ddot{\theta}_\omega - H_R \quad (2.53)$$

Parce que le mouvement linéaire agit sur le centre de la roue, la rotation angulaire peut être transformée en mouvement linéaire par une simple transformation.

$$\ddot{\theta}_\omega r = \ddot{x} \Rightarrow \ddot{\theta}_\omega = \frac{\ddot{x}}{r} \quad (2.54)$$

$$\dot{\theta}_\omega r = \dot{x} \Rightarrow \dot{\theta}_\omega = \frac{\dot{x}}{r} \quad (2.55)$$

Par la transformation linéaire, l'équation (2.52) et (2.53) devient

Pour la roue gauche

$$M_\omega \ddot{x} = \frac{-K_m K_e}{Rr} \dot{x} + \frac{K_m}{R} V_a - \frac{I_\omega}{r^2} \ddot{x} - H_L \quad (2.56)$$

Pour la roue droite

$$M_\omega \ddot{x} = \frac{-K_m K_e}{Rr} \dot{x} + \frac{K_m}{R} V_a - \frac{I_\omega}{r^2} \ddot{x} - H_R \quad (2.57)$$

Ajoutant les équations (2.56) et (2.57)

$$2\left(M_\omega + \frac{I_\omega}{r^2}\right) \ddot{x} = \frac{-K_m K_e}{Rr^2} \dot{x} + \frac{2K_m}{Rr} V_a - (H_L + H_R) \quad (2.58)$$

Le châssis du robot peut être modélisé comme un pendule inversé. La figure (2.4) montre le schéma du corps libre du châssis

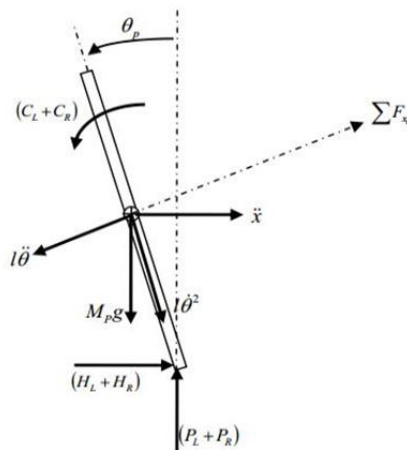


FIGURE 2.4 – Corps libre d'un châssis

Encore une fois, en utilisant la loi du mouvement de Newton, la somme des forces dans le sens horizontal est

$$\sum F_x = M_p \ddot{x} \quad (2.59)$$

$$(H_L + H_R) - M_p l \ddot{\theta}_p \cos \theta_p + M_p l \dot{\theta}_p^2 \sin \theta_p = M_p \ddot{x} \quad (2.60)$$

Ainsi

$$(H_L + H_R) = M_p \ddot{x} + M_p l \ddot{\theta}_p \cos \theta_p - M_p l \dot{\theta}_p^2 \sin \theta_p \quad (2.61)$$

La somme des forces perpendiculaires au pendule

$$\sum F_{xp} = M_p \ddot{x} \cos \theta_p \quad (2.62)$$

$$(H_L + H_R) \cos \theta_p + (P_L + P_R) \sin \theta - M_p g \sin \theta_p - M_p l \ddot{\theta}_p = M_p \ddot{x} \cos \theta_p \quad (2.63)$$

Somme des moments autour du centre de masse du pendule

$$\sum M_0 = I \alpha - (H_L + H_R) l \cos \theta_p - (P_L + P_R) l \sin \theta_p - (C_L + C_R) = I_p \ddot{\theta}_p \quad (2.64)$$

Le couple, appliqué sur le pendule à partir du moteur tel qu'il est définie dans l'équation (2.50), après une transformation linéaire, est :

$$\sum M_0 = I \alpha (C_L + C_R) = \frac{-2k_m K_e \dot{x}}{R} \frac{\dot{x}}{r} + \frac{2K_m}{R} V_a \quad (2.65)$$

Remplacer ceci dans l'équation (2.64) donne

$$-(H_L + H_R) l \cos \theta_p - (P_L + P_R) l \sin \theta_p - \left(\frac{-2k_m K_e \dot{x}}{R} \frac{\dot{x}}{r} + \frac{2K_m}{R} V_a \right) = I_p \ddot{\theta}_p \quad (2.66)$$

Ainsi

$$-(H_L + H_R) l \cos \theta_p - (P_L + P_R) l \sin \theta_p = I_p \ddot{\theta}_p - \frac{2k_m K_e}{Rr} \dot{x} + \frac{2K_m}{R} V_a \quad (2.67)$$

Multiplier l'équation (2.63) par -1

$$[-(H_L + H_R) l \cos \theta_p - (P_L + P_R) l \sin \theta_p] + M_p g l \sin \theta_p + M_p l^2 \ddot{\theta}_p = -M_p l \ddot{x} \cos \theta_p \quad (2.68)$$

Pour éliminer $(H_L + H_R)$ de la dynamique du moteur, l'équation (2.61) est remplacée dans l'équation (2.58).

$$(M_\omega + \frac{I_\omega}{r^2}) \ddot{x} = -\frac{K_m K_e}{Rr^2} \dot{x} + \frac{2K_m}{Rr} V_a - M_p \ddot{x} - M_p l \ddot{\theta}_p \cos \theta_p + M_p l \dot{\theta}_p^2 \sin \theta_p \quad (2.69)$$

Réorganisant les équations (2.68) et (2.69) donne les équations non linéaires de mouvement du système,

$$(I_p + M_p l^2) \ddot{\theta}_p - \frac{-2k_m K_e}{Rr} \dot{x} + \frac{2K_m}{R} V_a + M_p g l \sin \theta_p = -M_p l \ddot{x} \cos \theta_p \quad (2.70)$$

$$\frac{2K_m}{R} V_a = (2M_\omega + \frac{2I_\omega}{r^2} + M_p) \ddot{x} + \frac{2k_m K_e}{Rr^2} \dot{x} + M_p l \ddot{\theta}_p \cos \theta_p - M_p l \dot{\theta}_p^2 \sin \theta_p \quad (2.71)$$

Les deux équations ci-dessus peuvent être linéarisés en supposant $\theta_p = \pi + \phi$, où ϕ représente un petit angle de la direction verticale vers le haut. Cette simplification a été utilisée pour permettre d'obtenir de façon linéaire les contrôleurs de l'espace d'état pouvant être mis en ouvre.

donc, $\cos \theta_p = -1$, $\sin \theta_p = -\phi$ et $(\frac{d\theta_p}{dt})^2 = 0$

L'équation linéaire du mouvement est :

$$(I_p + M_p l^2) \ddot{\phi} - \frac{-2k_m K_e}{Rr} \dot{x} + \frac{2K_m}{R} V_a - M_p g l \phi = -M_p l \ddot{x} \quad (2.72)$$

Donc

$$\frac{2K_m}{R}V_a = (2M_\omega + \frac{2I_\omega}{r^2} + M_p)\ddot{x} + \frac{2k_m K_e}{Rr^2}\dot{x} - M_p l \ddot{\phi} \quad (2.73)$$

Afin d'obtenir la représentation dans l'espace d'état du système, les équations (2.72) et (2.73) sont réarrangées de telle façon que :

$$\ddot{\phi} = -\frac{M_p l}{(I_p + M_p l^2)}\ddot{x} + \frac{2k_m K_e}{Rr(I_p + M_p l^2)}\dot{x} - \frac{2K_m}{R(I_p + M_p l^2)}V_a + \frac{M_p g l}{(I_p + M_p l^2)}\phi \quad (2.74)$$

$$\ddot{x} = \frac{2K_m}{Rr(2M_\omega + \frac{2I_\omega}{r^2} + M_p)}V_a + \frac{2k_m K_e}{Rr^2(2M_\omega + \frac{2I_\omega}{r^2} + M_p)}\dot{x} + \frac{M_p l}{(2M_\omega + \frac{2I_\omega}{r^2} + M_p)}\ddot{\phi} \quad (2.75)$$

En injectant l'équation (2.74) dans l'équation (2.73) et l'équation (2.75) dans l'équation (2.72), et après quelques opérations algébriques, l'équation dans l'espace d'état du système est obtenue.

$$\begin{bmatrix} \dot{x}(t) \\ \dot{\phi}(t) \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & \frac{2K_m K_e (M_p l r - I_p - M_p l^2)}{Rr^2 \alpha} & \frac{M_p g l^2}{\alpha} & 0 \\ 0 & 0 & 0 & 1 \\ 0 & \frac{2K_m K_e (r\beta - M_p l)}{Rr^2 \alpha} & \frac{M_p g l^2}{\alpha} & 0 \end{bmatrix} \begin{bmatrix} x(t) \\ \dot{x}(t) \\ \phi(t) \\ \dot{\phi}(t) \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{2K_m (-M_p l r + I_p + M_p l^2)}{Rr^2 \alpha} \\ 0 \\ \frac{2K_m (r\beta - M_p l)}{Rr \alpha} \end{bmatrix} V_a$$

où

$$\alpha = I_p \beta + 2M_p l^2 (M_\omega + \frac{I_\omega}{r^2})$$

$$\beta = (2M_\omega + \frac{2I_\omega}{r^2} + M_p)$$

Le système peut être décrit par le schéma fonctionnel qui représente dans (2.5)

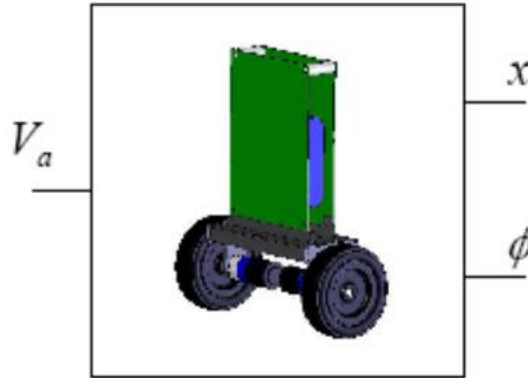


FIGURE 2.5 – Robot balancier à deux roues

Dans le modèle ci-dessus, on suppose que les roues du véhicule resteront toujours en contact avec le sol, et qu'il n'y a pas de glissement au niveau des roues. Les forces de virage sont également considérées comme négligeables.

Les valeurs des paramètres du moteur sont obtenues à partir d'un modèle de conception mécanique assisté par ordinateur utilisant le logiciel « SolidWork ». Le tableau (2.1) regroupe toutes les valeurs des paramètres précédents.

Variables	Nomenclatures	Valeurs	unités
K_m	Constante de couple moteur	0.006123	$N.m.A^{-1}$
K_e	Constante de force électromotrice	0.0069203	$V.s.rad^{-1}$
M_p	Masse du corps	1.2	kg
M_ω	Masse des roues	0.03	Kg
l	Longueur ua centre de masse du corps	0.12	M
r	Rayons des roues	0.035	M
I_p	Inertie du corps	0.0041	$Kg.m^2$
I_ω	Inertie des roues	0.000039	Kg^2
R	Résistance nominale	3	Ohm
g	La gravité	9.81	$m.s^{-2}$

TABLE 2.1 – Les valeurs des paramètres

Comme indiqué, la fonction de transfert est une matrice de (2x1) éléments, chacun décrivant la relation de l'entrée à l'une des sorties. La réponse impulsionnelle de chaque chemin de sortie (pendant les 3 premières secondes)

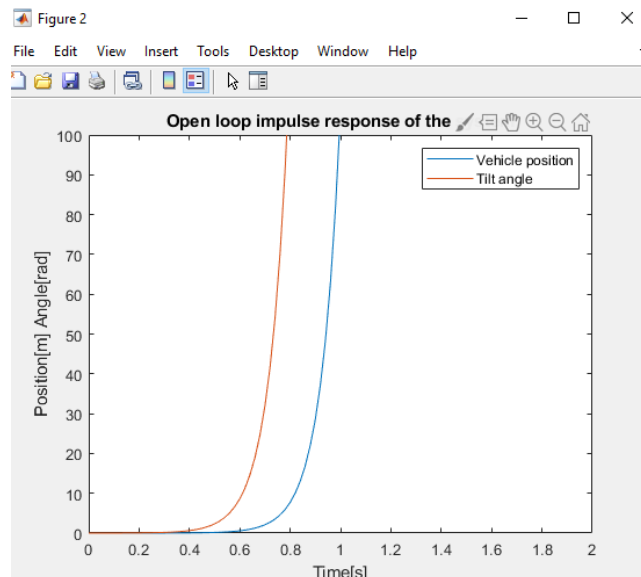


FIGURE 2.6 – Réponse impulsionnelle du système

Il est maintenant clair que le système est instable (augmentation très rapide du couple de sortie à un signal d'entrée faible) et qu'il a donc besoin d'un contrôleur robuste.

2.8 Conclusion

Dans ce chapitre nous avons parlé de trois étapes principales. En première étape, nous avons présenté des généralité sur les systèmes mécaniques sous-actionnée à deux degrés de liberté, ainsi nous avons fait la modélisation mathématique d'un moteur à courant continu par l'application des équation Euler-Lagrange (Lagrangien). En deuxième étape nous avons appliqué le formalisme de lagrangien sur un pendule inversé à deux degrés de liberté.

Dans le dernier étape nous avons appliqué ces équation sur un modèle dynamique pour faire la commande dans le chapitre 3.

Commande d'un robot auto balancier

Sommaire

3.1	Introduction	31
3.2	Commande par retour d'état continu	32
3.2.1	Principe de la commande par retour d'état	32
3.2.2	Résolution d'un problème de retour d'état par placement de pôle	33
3.2.3	Commande par placement de pôle	33
3.2.4	Principe de la commande par placement de pôle	33
3.2.5	Commande optimale	34
3.2.6	Commande linéaire quadratique	34
3.2.7	Définition de la fonction linéaire quadratique définie positive	34
3.2.8	Principe de la commande linéaire quadratique	34
3.2.9	Synthèse de régulateur LQR	35
3.2.10	Choix d'une matrice de pondération	36
3.2.11	Critère d'optimisation pour le régulateur	37
3.2.12	Gains du régulateur	37
3.3	Commande discret	37
3.3.1	L'objectif de la commande discret	39
3.3.2	Commande sous Matlab :	39
3.3.3	Régulateur discret	39
3.4	Résultat de simulation	40
3.4.1	Commande linéaire continue	40
3.4.2	Commande linéaire discrète	42
3.4.3	Interprétation des résultats	43
3.5	Conclusion	44

3.1 Introduction

le système sous actionnée est fondamentalement instable. Laissé sans régulateur stabilisateur, il ne pourra pas rester en position verticale s'il est dérangé[16]. La tâche du contrôleur consistera à modifier la tension continue en fonction des paramètres suivante : les deux variables Position du pendule (angle) et Position du chariot, de telle sorte que la position souhaitée du pendule (angle) la tâche de commande est accomplie (stabilisation en position

verticale, balancement ou commande de grue). Dans ce chapitre, nous allons introduire des rappels théoriques sur les différents outils de commandes appliqués au robot auto balancier, dont nous aurons besoin lors de la synthèse de notre schéma de commande pour la stabilisation robot auto balancier[3]. La première stratégie appelée **commande lineare continue** dans cette commande on traité deux types de cammande, la premiere commande appelle **commande par placement de pôle** et le deuxième **commande optimal** appellé **commande lineare quadratique (LQR)**proposée pour la phase de stabilisation du robot. Quant à la deuxième stratégie appelée **commande linéaire discret**, elle est basée sur la commande **LQR et PID** d'un robot auto balancie.

3.2 Commande par retour d'état continue

La commande par retour d'état est à la commande des systèmes modélisés par leur représentation d'état, ce que la boucle fermée est aux systèmes représentés par une fonction de transfert. L'idée consiste toujours à piloter le système par un signal de consigne et à générer automatiquement le signal de commande en confrontant en permanence la valeur de la consigne et le comportement réel du système. L'écart entre consigne et comportement réel sert de base au signal de commande du système. Dans la commande par retour d'état, nous n'allons pas mesurer le signal de sortie pour le boucler sur l'entrée, mais nous allons nous servir du vecteur d'état complet pour prendre connaissance du comportement du système. [17]

3.2.1 Principe de la commande par retour d'état

Le principe est de déterminer une commande telle que les pôles du système de la fonction de transfert du système bouclé soient convenablement placés dans le plan complexe et satisfasse des spécifications d'amortissement, de rapidité...

Les pôles de la fonction de transfert étant les valeurs propres de la matrice d'état, le but est donc de réaliser un asservissement modifiant convenablement la matrice d'état du système [11]. Soit un système décrit par l'équation d'état suivant :

$$\begin{cases} \dot{x} = Ax(t) + Bu(t) \\ y = Cx(t) + Du(t) \end{cases} \quad (3.1)$$

Dans le cadre de cette étude, on se restreint à la commande linéaire construite par rétroaction linéaire de l'état du système sur l'entrée :

$$u(t) = r(t) - kx(t) \quad (3.2)$$

Le signal de commande du système (autrement dit l'écart 2 doit être construit en soustrayant au signal de consigne un signal qui dépend du vecteur d'état. Ce vecteur d'état étant composé de n signaux $x_1(t), x_2(t), x_3(t) \dots x_n(t)$, on le multiplie par un vecteur ligne ($K2$) appelé vecteur de gain pour pouvoir effectuer cette soustraction. On a alors : $K=[k_1 k_2 k_3 \dots k_n]$

Les équations du système en boucle fermé sont :

$$\begin{cases} \dot{x} = Ax(t) + Bu(t) \\ u(t) = r(t) - kx(t) \\ y(t) = Cx(t) + Du(t) \end{cases} \quad (3.3)$$

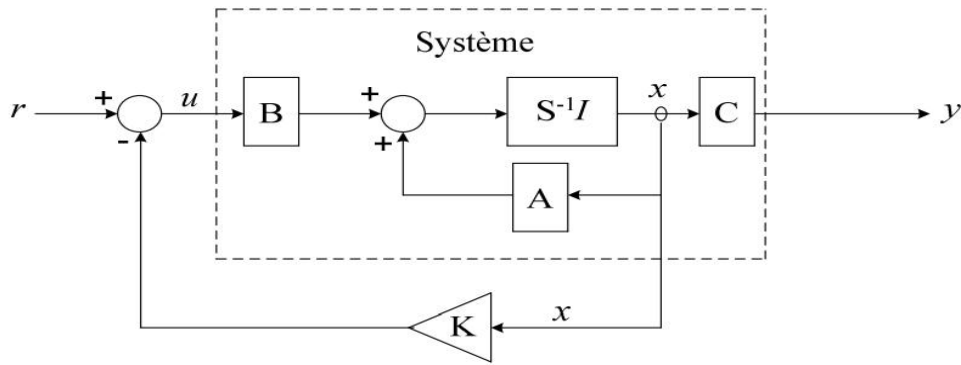


FIGURE 3.2 – Principe de la loi de commande placement de pôles

Par conséquent, la matrice d'état du système en boucle fermée vaut : $(A - BK)$. La dynamique du système bouclé est donc fixée par les valeurs propres de la matrice $(A - BK)$; ces valeurs propres sont les racines de l'équation caractéristique :

$$\det(pI - (A - BK)) = Q(p)_{A-BK} = 0 \tag{3.4}$$

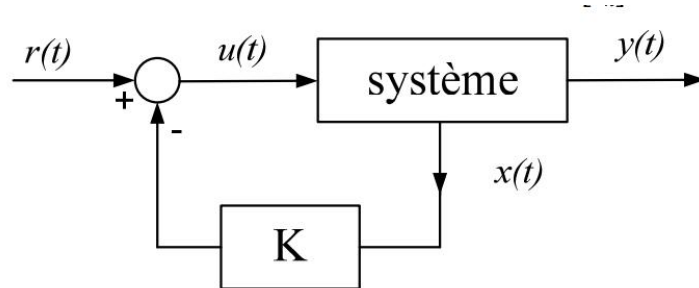


FIGURE 3.1 – Bouclage du système par un vecteur de gain

3.2.2 Résolution d'un problème de retour d'état par placement de pôle

On considère le système supposé commandable $\dot{x} = ax + bu$ et on cherche un régulateur pour ce système de la forme $u = r - kx$, où r est la nouvelle entrée. Il est légitime de vouloir choisir la matrice de régulation de façon à imposer les pôles du système bouclé[11].

3.2.3 Commande par placement de pôle

3.2.4 Principe de la commande par placement de pôle

Le but de la commande par placement de pôles est d'imposer au système un comportement spécifié ou des performances désirées à travers l'application d'une loi de commande qui place les pôles du système en boucle fermée aux positions qui réalisent ces performances. La détermination des pôles désirés en boucle fermée est basée sur des critères de réponse transitoire et/ou de réponse fréquentielle, telles que la vitesse, l'amortissement, ou la bande passante, aussi bien que sur des conditions sur le régime permanent fig(3.2)[17]. Cette tâche peut être accomplie en utilisant la loi de commande suivante : $u = -Kx$

Ce qui signifie que le signal de commande est déterminé par le vecteur d'état instantané. Une

telle loi de commande est appelée retour d'état. La matrice K de dimension $(1 * n)$ est appelée matrice de gain.

La figure (3.3) montre le système défini par l'équation (3.1), en boucle ouverte puis en boucle fermée, sous l'effet de la loi de commande (3.2).

La substitution de l'équation (3.2) dans l'équation (3.1) donne :

$$\begin{cases} \dot{x} = Ax(t) + B(r(t) - kx(t)) \\ = (A - BK)x + Br \end{cases} \quad (3.5)$$

3.2.5 Commande optimale

Les problèmes de commande optimale se rencontrent dans la vie de tous les jours : comment arriver à destination le plus rapidement possible, comment minimiser sa consommation... Pour un système dynamique donné et dont les équations sont connues, le problème de commande optimale consiste alors à trouver la commande minimisant un critère donné[17].

on s'intéressera plus particulièrement aux systèmes linéaires dans le cas d'un critère quadratique, cas connu sous le nom de commande linéaire quadratique (LQR : Linear Quadratic Regulator).

3.2.6 Commande linéaire quadratique

En Automatique la commande linéaire quadratique (**LQ**), est une méthode qui permet de calculer la matrice de gains d'une commande par retour d'état.

3.2.7 Définition de la fonction linéaire quadratique définie positive

La fonction quadratique $u(x) = x^T Q x$ est une matrice réelle symétrique, est dite définie positive si toute les valeurs propres de la matrice $Q(n \times n)$ sont strictement positives[11].

Les fonctions quadratiques sont souvent utilisées dans l'analyse des systèmes dynamiques. Notamment : l'énergie cinétique, l'énergie potentielle, élastique ou de gravité et l'énergie totale sont des fonctions quadratiques de l'état pour les systèmes mécaniques[11].

3.2.8 Principe de la commande linéaire quadratique

L'objectif de la commande linéaire quadratique (**LQ**) est de formuler le problème de commande par retour d'état en termes d'optimisation d'un critère qui traduit un compromis entre l'effort demandé à la commande (actionneurs) et les contraintes qu'on veut imposer à l'état.

Le contrôle optimal fournit une stratégie de conception alternative par laquelle tous les paramètres de conception de contrôle peuvent être même pour les systèmes à entrées et sorties multiples (MIMO)[2]. Il nous permet de formuler directement les objectifs de rendement d'un système de contrôle. De plus, il produit le meilleur système de contrôle possible pour un ensemble donné d'objectifs de rendement[12]. Le LQR est l'une des méthodes de rétroaction statique d'état les plus utilisées et les plus simples, principalement comme le LQR donc **Comment fonctionne LQR ?**

LQR est ce contrôleur optimal où la fonction object *ve* est une intégrale temporelle de la somme de l'énergie transitoire et de contrôler l'énergie exprimée en fonction du temps, c'est pourquoi nous essayons ici de minimiser cet objectif particulier.

En sélectionnant convenablement les matrices de pondération de performance et de contrôle des coûts, Q et R et en résolvant les problèmes suivant l'équation de Riccati soumise à une condition terminale afin de déterminer le gain optimal du régulateur, K . Une rétroaction d'état peut être généralisée pour un système de LTI est donnée ci-dessous :

Ici, tout en travaillant avec **LQR**, nous supposons que tous les n états sont disponibles pour le retour d'information et que les états sont complètement contrôlable, il existe une matrice de gain de rétroaction K , telle que l'entrée de commande de rétroaction d'état est donné par : $u = r(t) - Kx(t)$ voire la fig(3.2)

3.2.9 Synthèse de régulateur LQR

Considérons un système linéaire sous forme d'équation d'état suivant :

$$\begin{cases} \dot{x} = Ax(t) + Bu(t) \\ y(t) = Cx(t) + Du(t) \end{cases} \quad (3.6)$$

Où on va supposés que La paire (A, B) est stabilisable, c'est-à-dire qu'il n'y a pas de mode instable et ingouvernable dans le système.

Soit un régulateur par retour d'états dont le processus a pour équation d'état l'équation (3-5). Le problème simplifié du régulateur linéaire quadratique consiste à trouver la matrice du correcteur K qui minimise la fonction du coût (ou le critère de performance) suivante :

$$J(x_0, u) = \frac{1}{2}x^T(t_f)Sx(t_f) + \frac{1}{2} \int_{t_0}^{t_f} ((x(t)^T Q(t)x(t) + u(t)^T R(t)u(t))dt \quad (3.7)$$

Les matrices de pondération Q et R sont définies positives et symétriques. Et S est la matrice de solution de l'équation de Riccati (est définie positives et symétrique). Le Lagrangien s'écrit alors :

$$\mathcal{L}(x, u, p, t) = p^T A(t)x + p^T B(t)u + \frac{1}{2}(x(t)^T Q(t)x(t) + u(t)^T R(t)u(t)) \quad (3.8)$$

La loi de commande optimale est obtenue si la dérivée de lagrangien par rapport à la loi de commande est nulle :

$$\frac{\partial l}{\partial t} = B^T(t)p + R(t) = 0 \quad (3.9)$$

Donc, on peut tirer la commande optimale à partir d'équation (3-8) :

$$u_{opt} = -R^{-1}(t)B^T p(t) \quad (3.10)$$

Où :

$$p(t_f) = Sx(t_f) \quad (3.11)$$

Le principe du maximum donne la condition suivante :

$$\dot{p} = -\frac{\partial l}{\partial x} = -A^T(t)p - Q(t)x \quad (3.12)$$

Alors l'équation dynamique du système en boucle fermée s'écrit :

$$\dot{x} = A^T(t)x(t) - B^T(t)R^{-1}(t)B^T(t)p(t) \quad (3.13)$$

Les équations (3-10) et (3-12) peuvent se mettre sous la forme d'un système matriciel appelé système Hamiltonien :

$$\begin{bmatrix} \dot{x}(t) \\ \dot{p}(t) \end{bmatrix} = \begin{bmatrix} A(t) & -B(t)R^{-1}(t)B^T(t) \\ -Q(t) & -A^T(t) \end{bmatrix} \begin{bmatrix} x(t) \\ p(t) \end{bmatrix} \quad (3.14)$$

Ecrivons $p(t) = P(t)x(t)$, avec comme condition finale $P(t_f) = S$. L'équation (3-12) s'écrit alors :

$$\dot{p} = -(A^T(t)P(t) + Q(t))x(t) \quad (3.15)$$

Avec $\dot{p} = \dot{P}x(t) + P(t)\dot{x}(t)$ et l'équation d'état (3-1) du système, l'équation (3-10) s'écrit (en omettant la référence au temps pour alléger les notations) :

$$(\dot{P} + PA + A^T P - PBR^{-1}B^T P + Q)x = 0 \quad (3.16)$$

La solution est alors obtenue en résolvant l'équation de Riccati suivante :

$$\dot{P} + PA + A^T P - PBR^{-1}B^T P + Q = 0 \quad (3.17)$$

Avec la condition finale $P(t_f) = S$. Remarquons que la condition :

$$x^T(P + PA + A^T \dot{p} - PBR^{-1}P + Q)x = 0 \quad (3.18)$$

s'écrit aussi :

$$\frac{d}{dt}(x^T P x) + x^T Q x + u^T R u \quad (3.19)$$

Il est intéressant de noter que la commande optimale obtenue s'écrit comme un retour d'état $u = -Kx(t)$ avec :

$$K = -R^{-1}B^T P \quad (3.20)$$

3.2.10 Choix d'une matrice de pondération

Il est intéressant de remarquer d'abord que la multiplication des pondérations Q et R par un même scalaire laisse inchangé le gain K . En effet, soit P solution de (3-16) et soit le nouveau problème basé sur les pondérations $\hat{Q} = \rho Q$ et $\hat{R} = \rho R$

On vérifie que $\hat{p} = \rho p$ est solution de l'équation de Riccati correspondante. En effet :

$$\hat{k} = -\hat{R}^{-1}B^T \hat{p} = -R^{-1}B^T P = K \quad (3.21)$$

Sans restriction, les pondérations peuvent être choisies symétriques. Elles sont généralement choisies diagonales. Ainsi, on se ramène au choix de n scalaires pour l'état et de p scalaires pour la commande. Voici une méthode simple de choix et de modification des pondérations en vue d'aboutir à un correcteur satisfaisant.

1. Au départ, on choisit généralement des pondérations égales aux matrices identité.
2. Dans une seconde étape, on accélère ou décélère globalement le système en multipliant la matrice Q par un scalaire ρ (accélération avec $\rho < 1$ et décélération avec $\rho > 1$, jusqu'à obtenir une dynamique moyenne adaptée[3]).
3. Dans le cas où certains états auraient des dynamiques trop lentes par rapport à d'autres, on peut choisir d'augmenter la pondération de Q correspondant aux premiers.
4. Dans le cas où certains actionneurs seraient trop sollicités par rapport à d'autres, on peut choisir d'augmenter la pondération de R leur correspondant.

3.2.11 Critère 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[17] :

$$J(Q, R) = \int_0^{\infty} (X^T(t)QX(t) + R.u^2(t))dt \quad (3.22)$$

Avec :

$$Q = \begin{bmatrix} Q_x & 0 & 0 & 0 \\ 0 & Q_{\dot{x}} & 0 & 0 \\ 0 & 0 & Q_{\theta} & 0 \\ 0 & 0 & 0 & Q_{\dot{\theta}} \end{bmatrix} \quad (3.23)$$

Nous fixons les coefficients de pondération et nous adoptons les critères suivants :

1. l'amplitude du débattement angulaire n'est pas très importante.
2. le déplacement du robot 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

3.2.12 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 voire le programme dans l'annexe 3[17]. Remarque : pour le modèle simulink contrôle le mode pôle, il s'agit du même modèle mais doit être commuté le gain k_{lqr} par k_{place} , voir le programme dans l'annexes B.

3.3 Commande discret

La représentation des systèmes dynamiques sous forme discrète est souvent nécessaire lorsqu'on doit réaliser de manière pratique le contrôle avec un système numérique d'acquisition, de traitement des données et des algorithmes ainsi que leur application. Alors que jusqu'à l'année 1960, les contrôleurs utilisaient seulement des composants analogiques, avec des circuits électriques assurant les tâches de mesure, de calcul et de transmission des ordres vers les actionneurs.

Aujourd'hui la majorité des systèmes de contrôle utilisent des convertisseurs analogiques-numériques, des microprocesseurs, et des convertisseurs numériques-analogiques. Dans un tel schéma, les lois de commande sont exécutées en temps discret, avec une périodicité (constante) correspondant à une fréquence choisie en fonction de la réactivité naturelle du système à contrôler[18].

On peut considérer qu'en aéronautique, les boucles de guidage-pilotage sont exécutées tous les quelques 10 millisecondes, alors que dans le génie des procédés de la chimie lourde, les périodes sont plutôt de l'ordre de la minute. Ainsi, dans la chaîne d'action entourant le système physique, on a des signaux numérisés utilisables par des microprocesseurs fonctionnant suivant des cycles d'horloge cadencés à une fréquence d'échantillonnage $1/Te > 0$.

En outre, on doit souvent considérer plusieurs horloges, dont les fréquences n'ont pas nécessairement de multiple commun. Nous laissons volontairement de côté ces problèmes difficiles dits de multi-échantillonnage, même s'ils ont une importance pratique certaine[18].

Dans cette partie, nous discutons l'opération de conversion des modèles à temps continu en modèles à temps discret (ou équation des différences)[14]. Nous présenterons également la transformation en z et nous montrerons comment l'utiliser pour analyser et concevoir des contrôleurs pour les systèmes à temps discret.

La figure ci-dessous montre le système typique de rétroaction en temps continu que nous avons considéré jusqu'ici dans ce chapitre . Presque tous les régulateurs à temps continu peuvent être mis en oeuvre en utilisant une électronique analogique.

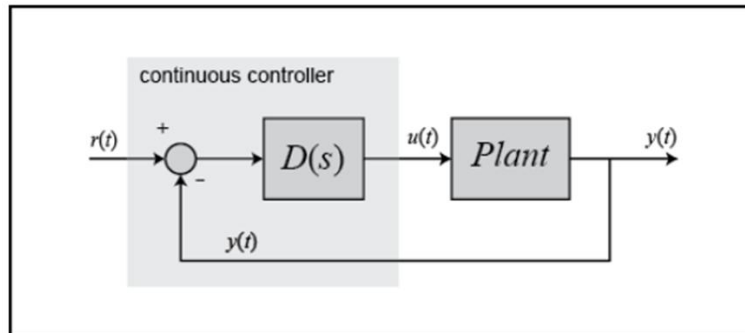


FIGURE 3.3 – Contrôleur continue

Le régulateur continu, enfermé dans le rectangle ombragé, peut être remplacé par un régulateur numérique, illustré ci-dessous, qui effectue la même tâche de régulation que le régulateur continu. La différence fondamentale entre ces contrôleurs est que le système numérique fonctionne sur des signaux discrets (échantillons des signaux détectés) plutôt que sur des signaux continus. Un tel changement peut s'avérer nécessaire si nous souhaitons implémenter notre algorithme de contrôle dans un logiciel sur un ordinateur numérique, ce qui est souvent le cas[19].

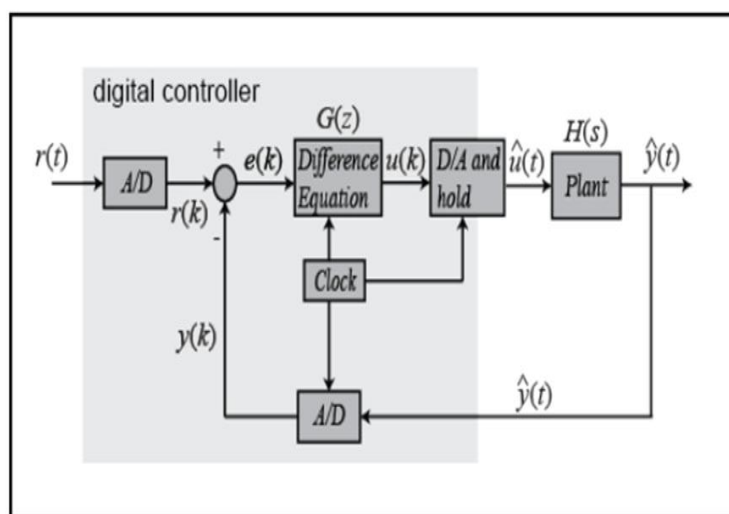


FIGURE 3.4 – Contrôleur numérique

Les différents signaux du schéma du système numérique ci-dessus peuvent être représentés par les graphiques suivants.

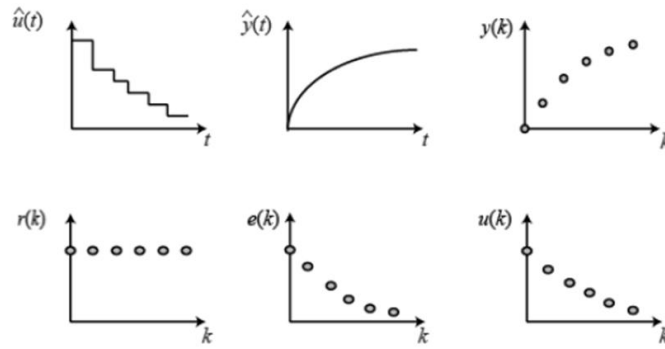


FIGURE 3.5 – Résultat des signaux discrets à partir des signaux continus

3.3.1 L'objectif de la commande discret

Le but de cette partie est de démontrer comment utiliser MATLAB pour travailler avec des fonctions discrètes, sous forme des fonctions de transfert ou d'espace d'état, afin de concevoir des systèmes de contrôle numérique.

3.3.2 Commande sous Matlab :

Il existe une fonction **MATLAB c2d()** qui convertit un système continu donné (sous forme de fonction de transfert ou d'espace d'état) en un système discret en utilisant l'opération de maintien d'ordre zéro expliquée ci-dessus. La syntaxe de base pour cela dans MATLAB est $sys_d = c2d(sys, Ts, 'zoh')$. Le temps d'échantillonnage (T_s en s/échantillon) doit être inférieur à $1/(30BW)$, où BW est la fréquence de bande passante en boucle fermée du système.

3.3.3 Régulateur discret

Le régulateur présenté dans la simulation de ce mémoire est un contrôleur continu qui n'est pas implémentable dans notre système réel ce qui nous oblige d'utiliser un régulateur numérique.

Avec MATLAB, nous convertissons le modèle linéaire à temps invariant continu du système en un modèle discret. Le code MATLAB suivant convertit le système en un modèle discret avec un temps d'échantillonnage de 0.1 :

- $twip_d = c2d(dt\text{wip}(), 0.1)$;
- $[A_d, B_d, C_d, D_d] = ssdata(twip_d)$;

La première commande $c2d()$ convertit un modèle LTI continu en modèle discret. Bien que le seconde commande $ssdata()$ extrait les matrices State-Space du modèle pour la variable l'objet du calcul des gains du contrôleur.

Maintenant que nous avons notre modèle discret, nous calculons les gains de la commande LQR : Nous utilisons le code suivant pour le notre contrôleur :

- » $A1 = Ad(1 : 4, 1 : 4)$;
- » $B1 = Bd(1 : 4, 1)$;
- » $x = 10$; $y = 50$;
- » $Q = [x \ 0 \ 0 \ 0; 0 \ 1 \ 0 \ 0; 0 \ 0 \ y \ 0; 0 \ 0 \ 0 \ 1]$;
- » $R = 1$;
- » $K_{dlqr} = dlqr(A1, B1, Q, R)$;

Le programme et le modèle Simulink se trouvent dans l'**annexe B** et les résultats expérimentales dans le **chapitre 4**.

3.4 Résultat de simulation

3.4.1 Commande linéaire continue

3.4.1.1 Commande par placement de pôle

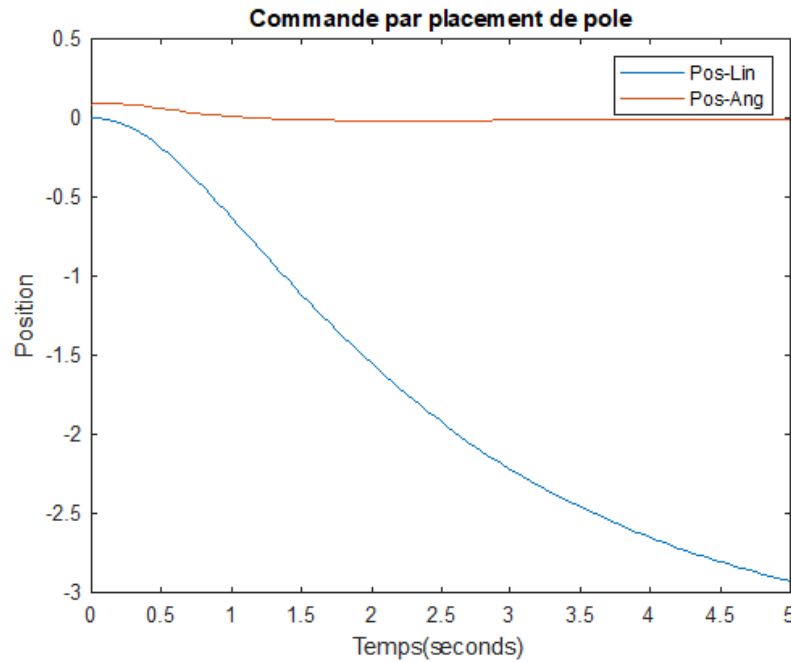


FIGURE 3.6 – Résultat de simulation de la commande LQR continue pour $\theta = 0$

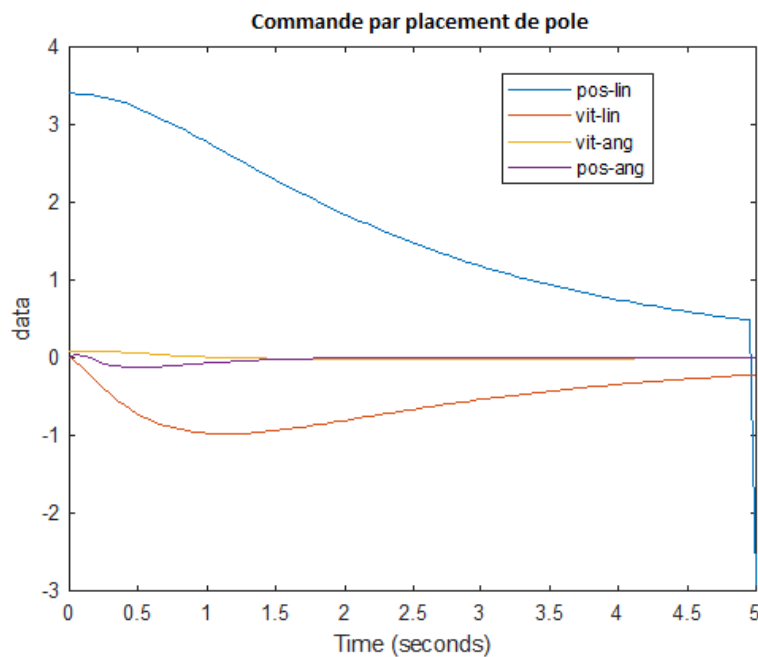


FIGURE 3.7 – Signaux d'erreurs

3.4.1.2 Commande LQR

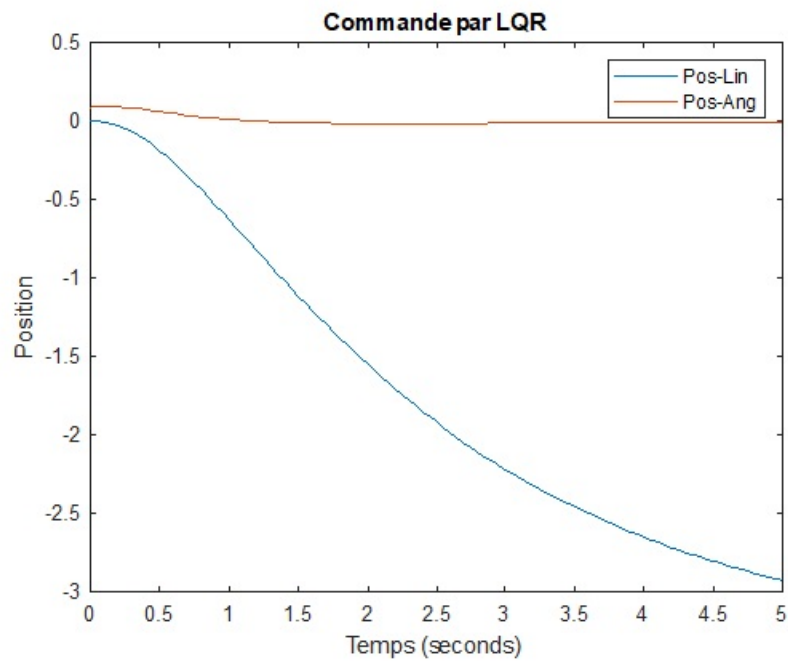


FIGURE 3.8 – Résultat de simulation de la loi commande LQR continue pour $\theta = 0$

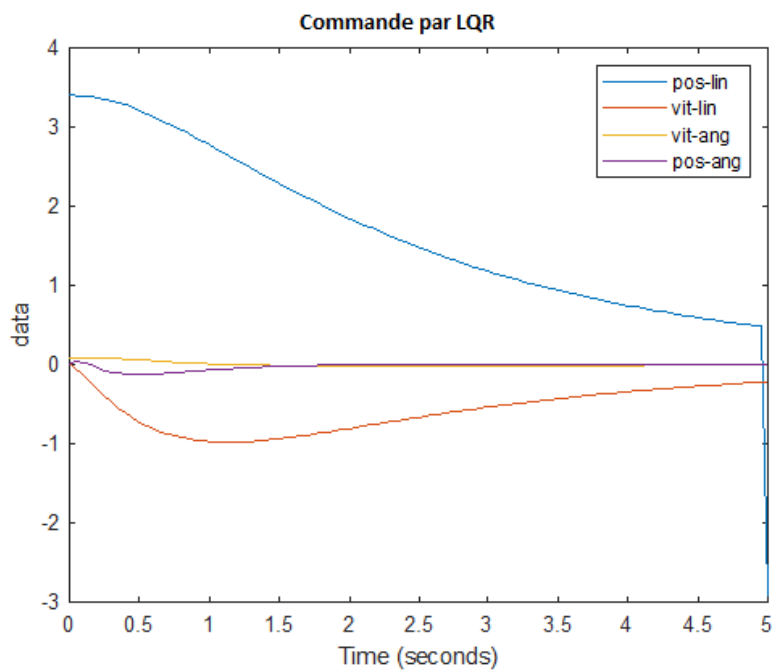


FIGURE 3.9 – Signaux d'erreurs

3.4.2 Commande linéaire discrète

3.4.2.1 Commande par placement de pôle

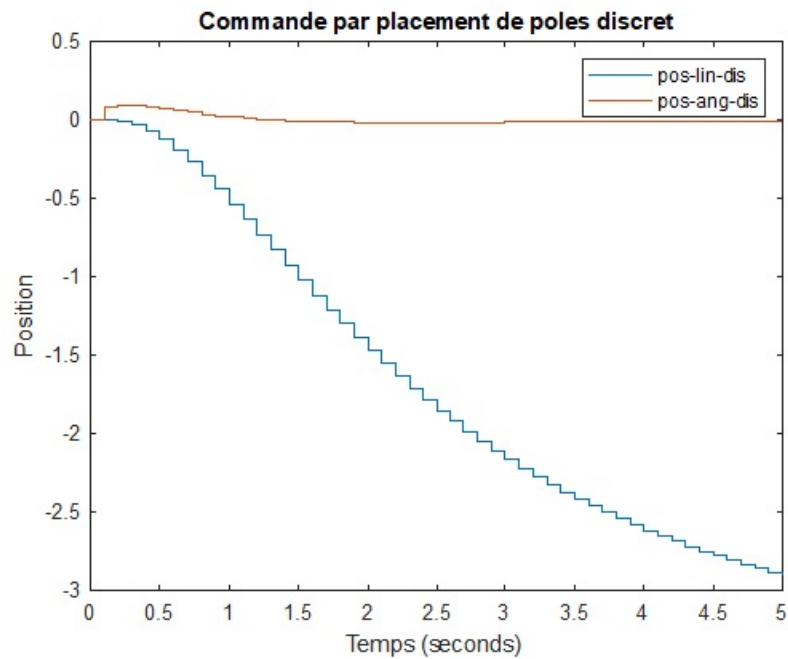


FIGURE 3.10 – Résultat de simulation de la loi de commande placement de pôle discrète pour $\theta = 0$

3.4.2.2 Commande LQR discret

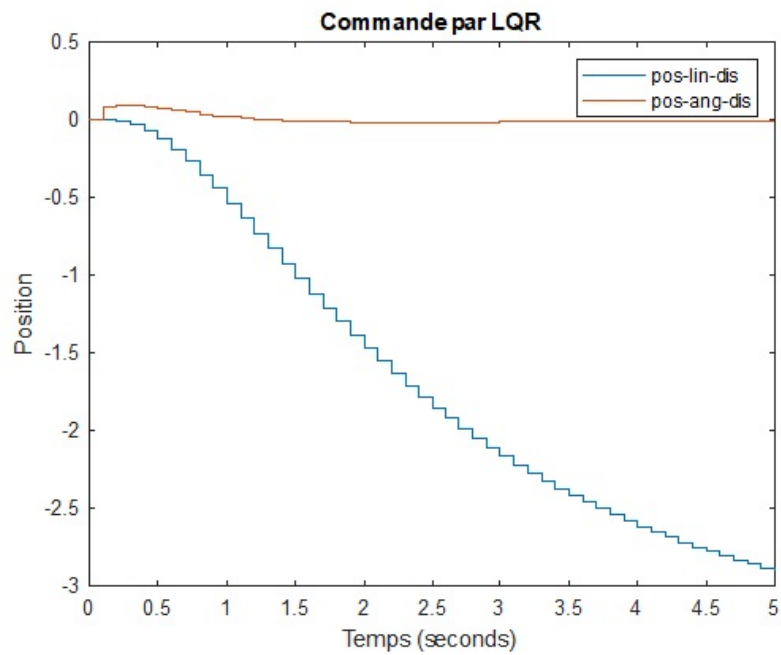


FIGURE 3.11 – Résultat de simulation de la commande LQR discrète pour $\theta = 0$

3.4.2.3 L'animation sous MATLAB

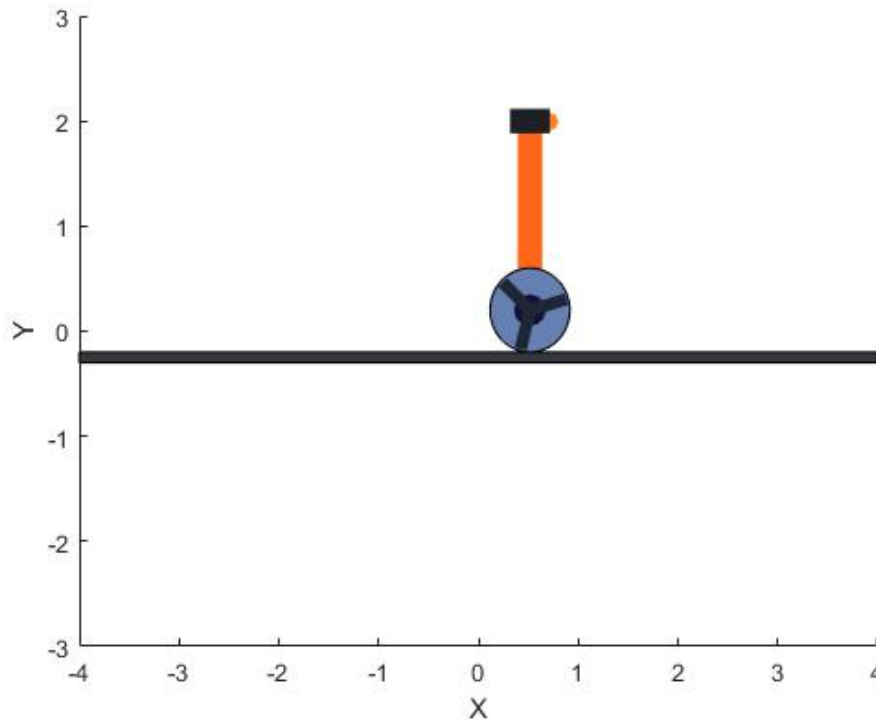


FIGURE 3.12 – Robot auto balancier en 2D sous MATLAB

3.4.3 Interprétation des résultats

De la figure 3.7, nous constatons que la commande basée sur le principe de minimum d'énergie assure le balancement du robot (la convergence) vers la zone linéaire.

Pour les systèmes linéaire continue :

Dans les figure 3.7, nous constatons que la commande appelé commande par placement de pôles sur la position angulaire et la position linéaire, les deux grandeurs convergent et les erreurs tendent vers zéro. Ainsi, lorsque nous modifions la position du robot à l'aide du curseur sous MATLAB, le robot reste stable, c'est-à-dire que l'angle de rotation maintient une position d'équilibre verticale $\theta = 0$

Dans cette partie, nous sommes concentrés sur le contrôle de l'accélération angulaire et linéaire du système, sachant que, les deux étant couplée de manière linéaire. Dans ce cas, il est possible de trouver une accélération nulle (angulaire et linéaire) pour stabiliser le robot en point avec le pendule dans la position verticale (Les moteur s'arrêtent), Si le robot est poussé vers l'avant, la vitesse augmente dans le même sens de rotation jusqu'à ce que le robot se balance en arrière, et inversement correct.

Pour les systèmes linéaire discret : Dans les figures(3.10 et 3.11) on remarque que les grandeurs du système ont les mêmes allure du système continu mais la seule différence c'est que la position lineaire pour le système continues arrive jusqu'à 3m par contre dans le systèmes discret la position est moins de 0.7m. Pour le pas d'échantillonnage nous avons pris $T_e < 10^{-6}$ en respectant biensur le theoreme de Shannon(il y'aura plus de détails dans le chapitre 4).

3.5 Conclusion

Dans ce chapitre, nous avons simulé sous Matlab deux lois de commande pour stabiliser notre robot. Dans un premier temps, nous avons appliqué les deux lois de commande (Commande linéaire par placement de pôles et la commande LQR) sur le modèle continu du robot ou nous avons trouvé de bonnes résultats, après nous avons appliqué les mêmes lois de commande sur le système discrétisé et on a obtenu des résultats meilleurs puisque la position linéaire pour stabiliser le pendule à la position verticale était beaucoup mieux que le cas continu. Et puisque la commande continue n'existe pas en réalité, dans le chapitre qui suit nous allons appliquer des lois de commande discrètes sur notre prototype de robot auto-balancier.

Conception et réalisation d'un robot auto balancier

Sommaire

4.1	Introduction :	46
4.2	La structure de notre robot	46
4.3	Architecture générale	47
4.4	Généralité sur les matérielles utilisées	47
4.4.1	Partie hardware	47
4.4.2	L'utilité de la carte ARDUINO MEGA 2560	51
4.5	Partie software	51
4.5.1	MATLAB	52
4.5.2	Simulink	52
4.5.3	Arduino IDE	55
4.5.4	Solidwork	56
4.6	L'environnement Matlab/Simulink	57
4.7	L'interfaçage Arduino/Matlab/Simulink	57
4.7.1	Carte Arduino comme d'interface	57
4.8	ArduinoIO	59
4.9	Arduino Target	59
4.9.1	Installation du package ArduinoIO	59
4.9.2	Exploitation de la bibliothèque ArduinoIO sous Simulink	60
4.9.3	Exploitation du package ArduinoIO sous Matlab	60
4.10	Arduino Target	61
4.11	les bibliothèques utilisées	61
4.11.1	compilateur de C++	61
4.11.2	ArduinoIO	61
4.11.3	Embedded Coder Target for Arduino	62
4.11.4	RASPLIB	62
4.11.5	Afficheur LCD	62
4.11.6	MPU6050	63
4.11.7	Driver moteur	64
4.11.8	Bluetooth	64
4.12	traitement des données	64
4.12.1	présentation de ADC	64

4.12.2	Acquisition des donnée	65
4.12.3	Envoie des données	67
4.13	Détermination de l'angle à partir des données d'accéléromètre et de gyroscope	68
4.13.1	Calculer l'angle de position angulaire avec les données du gyroscope	68
4.13.2	Calculer de la position angulaire avec les données de l'accéléromètre	69
4.13.3	Création du filtre complémentaire	70
4.13.4	Contrôle de la position linéaire à l'aide d'un encodeur incrémental	71
4.13.5	Capteur de distance Ultrason HC-SR04 :	71
4.13.6	commande LQR	72
4.13.7	commande PID	72
4.14	Présentation du robot sous Solidwork	72
4.14.1	l'interface de commande	73
4.15	Réalisation du robot	73
4.15.1	Les étapes de réalisation	73
4.16	Présentation du robot	75
4.17	Conclusion	76

4.1 Introduction :

Un robot capable d'équilibrer verticalement sur ses deux roues est connu sous le nom de robot à deux roues. Robot d'équilibrage sur roues. Le processus d'équilibrage est généralement appelé contrôle de stabilité. Les deux roues sont situées en dessous de la base et permettre au châssis du robot de maintenir une position verticale en se déplaçant dans l'espace de travail.

D'inclinaison, vers l'avant ou vers l'arrière, dans le but de maintenir le centre de l'axe de basculement la masse au-dessus des essieux de roue. Les roues assurent également la locomotion ainsi permettant au robot de traverser différents terrains et environnements. Ce type de robot pose un problème difficile et a donné lieu à de nombreux problèmes. Des conceptions utiles et intéressantes en cours d'élaboration. Un tel robot à deux roues qui est devenu un succès commercial est le Segway. L'a eu des répercussions immédiates dans le secteur du transport personnel, où une alternative aux fauteuils roulants encombrants est maintenant disponible. Le Segway prouve une des possibilités de mobilité confortables pour les personnes âgées ou les personnes handicapées améliorer leur sentiment d'indépendance individuelle en même temps.

4.2 La structure de notre robot

On subdivise ce chapitre par 04 parties principales pour la conception et la réalisation d'un robot auto balancier à deux roues comme suivant :

Dans la partie 01 : on donne une généralité sur les matérielles (hardware) et software utilise dans notre projet.

Dans la partie 02 : on parle sur l'interface hardware /software et software/software.

Dans la partie 03 : on parle sur la transmission et la réception des donnes.

Dans la partie 04 : on donne le modèle Simulink qui utilise pour la commande de notre robot en temps réelle avec une application graphique sous MATLAB.

4.3 Architecture générale

La figure suivante présente l'architecture générale de notre robot et ces différents organes (Figure 4. 1), le robot mobile est équipé par :

- Deux moteurs à courant continu.
- Unité de traitement (Mbed).
- Centrale inertielle (IMU).
- Organe de puissance (L293D) .

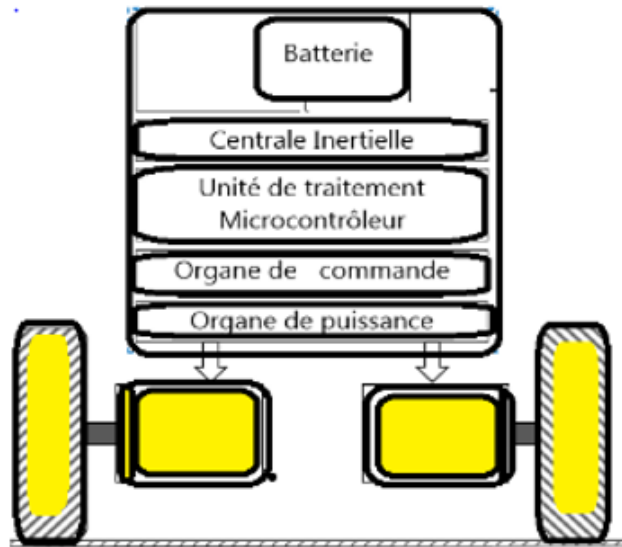


FIGURE 4.1 – Architecture générale du robot

4.4 Généralité sur les matérielles utilisées

4.4.1 Partie hardware

- a) MPU6050.
- b) Arduino atmega 2560.
- c) L293D.
- e) 02 Moteur à courant continu.
- f) LCD, Bluetooth, alimentation.

4.4.1.1 DC moteur

Pour construire un robot auto-équilibrant, nous pouvons utiliser n'importe quel type de moteur comme le moteur à courant continu, le moteur brushless, l'équipement automatique, etc., entre 50 rpm à 150 rpm. Mais nous avons considéré le moteur DC comme il est en général, trois courbes distinctes sont considérées comme importantes pour les moteurs à courant continu qui sont :

- couple par rapport au moteur.
- Vitesse par rapport au courant moteur .
- Vitesse par rapport au couple.

Et tout cela nous avons parlé dans le chapitre 02 en détail. Donc représente4.2 qui illustre le moteur utilise dans notre projet et les caractéristiques de ce dernier et sous l'annexe(4.1).

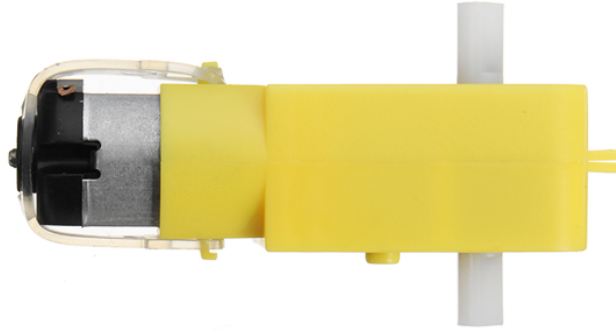


FIGURE 4.2 – Moteur à courant continu

4.4.1.2 MPU6050

Pour obtenir l'angle d'inclinaison du robot d'équilibrage, on utilise l'unité de mesure inertielle à six degrés de liberté (IMU) comme sur la figure 4.3. Le MPU 6050 est un 6 DOF, ce qui signifie qu'il donne six valeurs en sortie. La valeur se compose de trois valeurs de l'accéléromètre et de trois valeurs du gyroscope. Cette puce utilise le protocole I2C (Inter Integrated Circuit) pour la communication. Le module est équipé d'un processeur de mouvement numérique (DMP) capable de traiter des algorithmes complexes de fusion de mouvement à 9 axes. Les broches SDA et SCL sont utilisées pour établir une connexion avec les broches Arduino 20 et 21 pour recevoir les données de l'accéléromètre et du gyroscope. La pinte d'interruption (INT) sert à indiquer à l'Arduino quand lire les données du module et cette pinte n'indique l'Arduino que lorsque les valeurs changent [20].

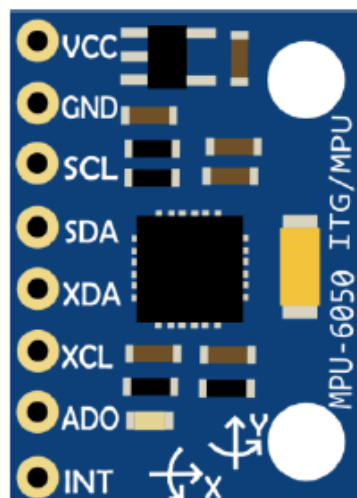


FIGURE 4.3 – MPU6050

4.4.1.3 l'unité de puissance

Dans cette partie on parle sur le driver d'un moteur a courant continue d'appelé L293D ce dernier c'est un carte de commande de puissance pour contrôler le sens de rotation et la vitesse de rotation d'un moteur à courant continue qui utilise pour plusieurs types d'un dc moteur par exemple servomoteur, stepper moteur...etc ,voire la figure(4.3)

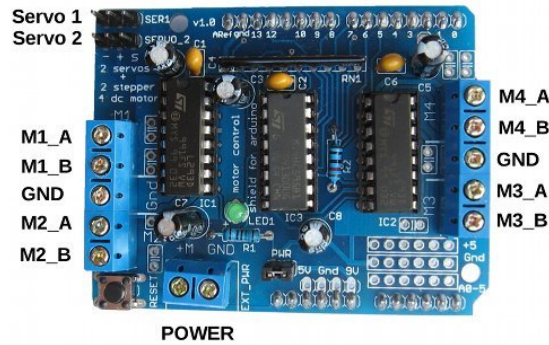


FIGURE 4.4 – Driver d'un moteur à courant continu

4.4.1.4 LCD

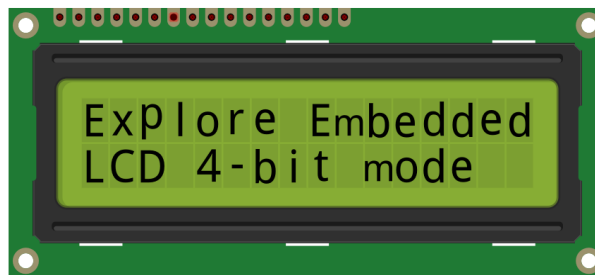


FIGURE 4.5 – Afficheur LCD

Dans ce projet on utilise un afficheur LCD (liquide Crystal display) pour afficher la vitesse de rotation de chaque moteur, LCD qui utilise dans notre projet est voire la figure 4.5.

4.4.1.5 Module de communication

Module de communication série HC-05 Bluetooth intégré (peut être un raccourci pour module) Deux modes de fonctionnement : Mode de travail Réponse à la demande et numérotation automatique Système de travail. Il existe trois rôles professionnels (maître, esclave et bouclage) en mode de connexion automatique. Lorsque l'appareil est en connexion automatique en mode de travail, la méthode par défaut suivra finalement automatiquement le transfert de données spécifié. Lorsque l'unité est en mode d'action Réponse à la demande, l'utilisateur peut envoyer une commande à unité pour ajuster les paramètres de contrôle et envoyer la commande de contrôle. Méthode de travail L'unité peut être commutée en contrôlant le niveau d'entrée PIN (PIO11). PIO11 : est le commutateur de mode de travail. Lorsque ce port PIN est entré de haut niveau, le travail Le mode deviendra le mode de travail de réponse à la commande. Tant que ce port PIN est entré au niveau bas ou suspendu dans l'air, le mode de travail deviendra le

mode de travail de connexion automatique. Le module de communication qui utilise dans notre projet est pour commander notre robot à distance voir les détails en partie 04[20] .

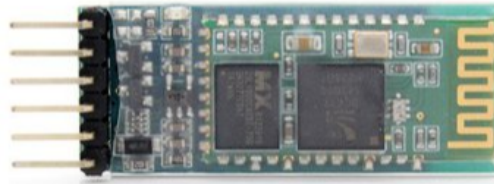


FIGURE 4.6 – Module de communication H-05

4.4.1.6 Alimentation

Dans notre projet on utilise 02 batteries de 9v pour le driver de moteur chaque batterie alimente une seule moteur pour donne la vitesse acceptable de notre robot, et on utilise un autre alimentation de 3v pour alimenter la carte de commande. Voire la figure 4.6 (type de batteries utilise)



FIGURE 4.7 – Alimentation 9v

4.4.1.7 L'unité de commande

L'unité de commande qui utilise dans notre robot est la carte arduino mega 2560, ce dernier est un carte électronique programmable qui permet de traiter les informations qui entre ou sorte vers la carte, Il est similaire au cerveau humain. Le cerveau reçoit des informations de l'environnement externe, puis les traite dans des conditions Standard, puis envoie une dynamo dynamique aux membres pour remplir sa fonction. La rapidité d'application dépend de la rapidité de traitement des données, les processeurs fabriqués aujourd'hui ont des vitesses très élevées mais très faibles comparées à celles de l'Allah sobhanah Wa ta3ala. donc cette carte qui utilise pour commander la vitesse et la position de notre robot, la figure 4.7 illustre ce type de carte.

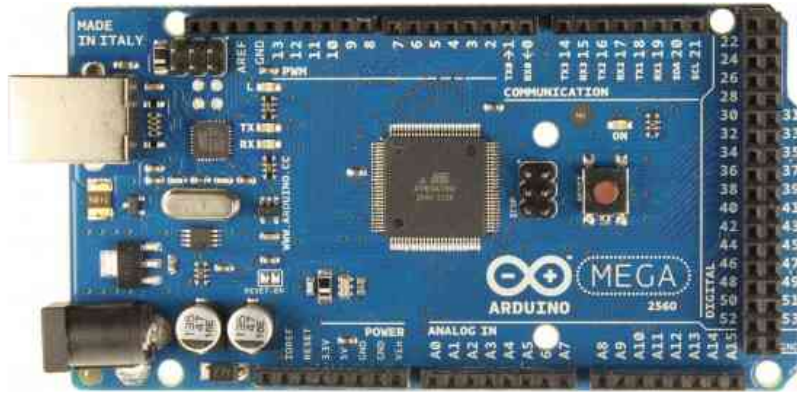


FIGURE 4.8 – Arduino MEGA 2560

Chacune des 54 broches numériques de l'Arduino 2560 Mega peut être utilisée comme entrée ou sortie, en utilisant les fonctions `pinMode()`, `digitalWrite()` et `digitalRead()`. Ils fonctionnent à 5 volts. Chaque broche peut fournir ou recevoir un maximum de 40 mA et possède une résistance pull-up interne (déconnectée par défaut) de 20 à 50 kOhms.

De plus, certaines épines ont des fonctions spécialisées :

Série 0 : 0 (RX) et 1 (TX) ;

Série 1 : 19 (RX) et 18 (TX) ;

Série 2 : 17 (RX) et 16 (TX) ;

Série 3 : 15 (RX) et 14 (TX).

Utilisé pour recevoir (RX) et transmettre (TX) les données série TTL.

PWM : (2 à 13 et 44 à 46). Fournir une sortie PWM 8 bits avec la fonction `analogWrite()`.

TWI : 20 (SDA) et 21 (SCL). Prise en charge de la communication TWI à l'aide de la bibliothèque `Wire`.

Le Mega2560 dispose de 16 entrées analogiques, dont chacune fournit 10 bits de résolution 1024 valeurs différentes.

Les caractéristiques de cette carte est sur l'annexe C.

4.4.2 L'utilité de la carte ARDUINO MEGA 2560

Nous utiliserons cette carte c'est comme une interface entre MATLAB et notre robot seulement parce que l'interface entre le système et MATLAB est très chère et n'existe pas dans notre laboratoire mais cette carte moins chère et facile d'utilisation.

4.5 Partie software

Nous nous utiliserons dans notre projet des logiciels très important pour la programmation et la commande et la visualisation en 2D (2 dimension) et en 3D, on parle dans ce sous partie sur l'utilité de chaque software et le but de leur utilisation dans notre projet.

Donc pour les softwares qui utilisent sont :

- Matlab/Simulink.

- Arduino IDE.
- Solidwork.

4.5.1 MATLAB

MATLAB est un langage de programmation et un environnement d'analyse numérique de quatrième génération. Il est utilisé pour les calculs matriciels, le développement et l'exécution d'algorithmes, la création d'interfaces utilisateur (UI) et la visualisation de données. L'environnement de calcul numérique multi-paradigmes permet aux développeurs de s'interfacer avec des programmes développés dans différents langages, ce qui permet d'exploiter les forces uniques de chaque langage à des fins diverses.

MATLAB est utilisé par les ingénieurs et les scientifiques dans de nombreux domaines tels que le traitement d'images et de signaux, les communications, les systèmes de contrôle pour l'industrie, la conception de réseaux intelligents, la robotique ainsi que la finance informatique[13].

4.5.1.1 l'utilité de MATLAB

MATLAB combine un environnement de bureau adapté aux processus itératifs d'analyse et de conception avec un langage de programmation qui exprime directement les mathématiques matricielles et de matrice. Il inclut l'éditeur Live Editor pour créer des scripts qui combinent code, sortie et texte formaté dans un bloc-notes exécutable.

4.5.1.2 Construction professionnelle

Les boîtes à outils MATLAB sont développées par des professionnels, rigoureusement testées et entièrement documentées[13].

4.5.1.3 Avec des applications interactives

Les applications MATLAB vous permettent de voir comment différents algorithmes fonctionnent avec vos données. Itéré jusqu'à ce que vous obteniez les résultats souhaités, puis générez automatiquement un programme MATLAB pour reproduire ou automatiser votre travail.

4.5.2 Simulink

Simulink, développé par MathWorks, est un outil commercial pour la modélisation, la simulation et l'analyse de systèmes dynamiques multi-domaines, son interface principale est un outil graphique de schéma-bloc et un ensemble personnalisable de bibliothèques de blocs.

Il offre une intégration étroite avec le reste de l'environnement MATLAB et peut soit piloter MATLAB, soit faire l'objet d'un script. Simulink est largement utilisé dans la théorie du contrôle et le traitement numérique du signal pour la simulation multi-domaine et la simulation basée sur modèle[13].

le modèle Simulink il est combine les bibliothèque suivante :

Stateflow : permet de développer des machines d'états et des organigrammes, représente4.9

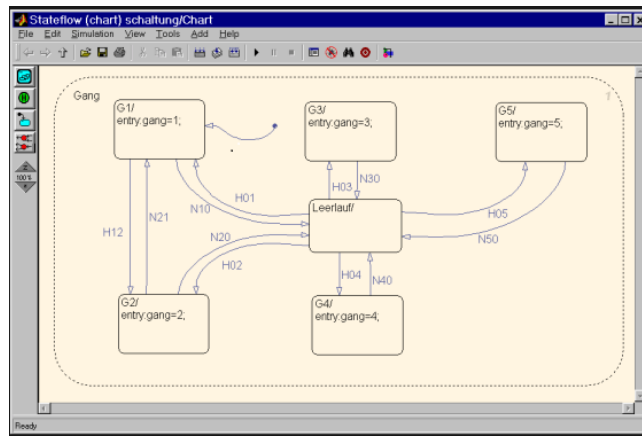


FIGURE 4.9 – Modèle Stateflow sous MATLAB

Simulink Coder : permet la génération automatique de code source C pour l'implémentation en temps réel des systèmes, représenté 4.10.

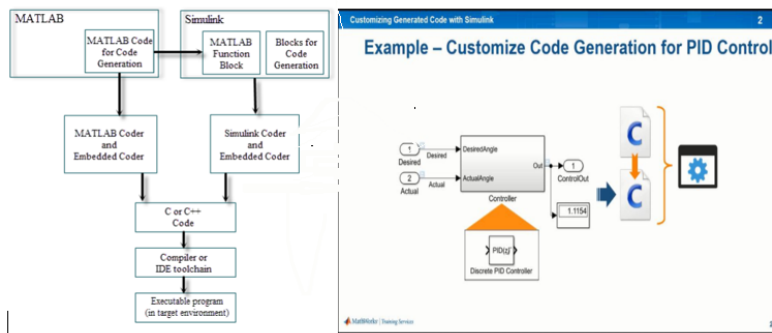


FIGURE 4.10 – Génération du code sous MATLAB

xPC Target et les systèmes temps réel : basés sur x86 fournissent un environnement pour simuler et tester les modèles Simulink et Stateflow en temps réel sur le système physique, représenté 4.11.

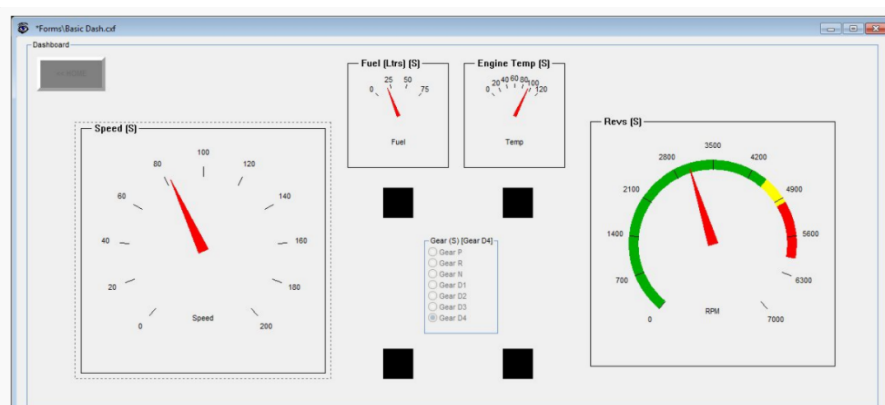


FIGURE 4.11 – L'interface de commande en temps réel

SimEvents : fournit une bibliothèque de blocs de construction graphiques pour la modélisation des systèmes de files d'attente, représentée 4.12

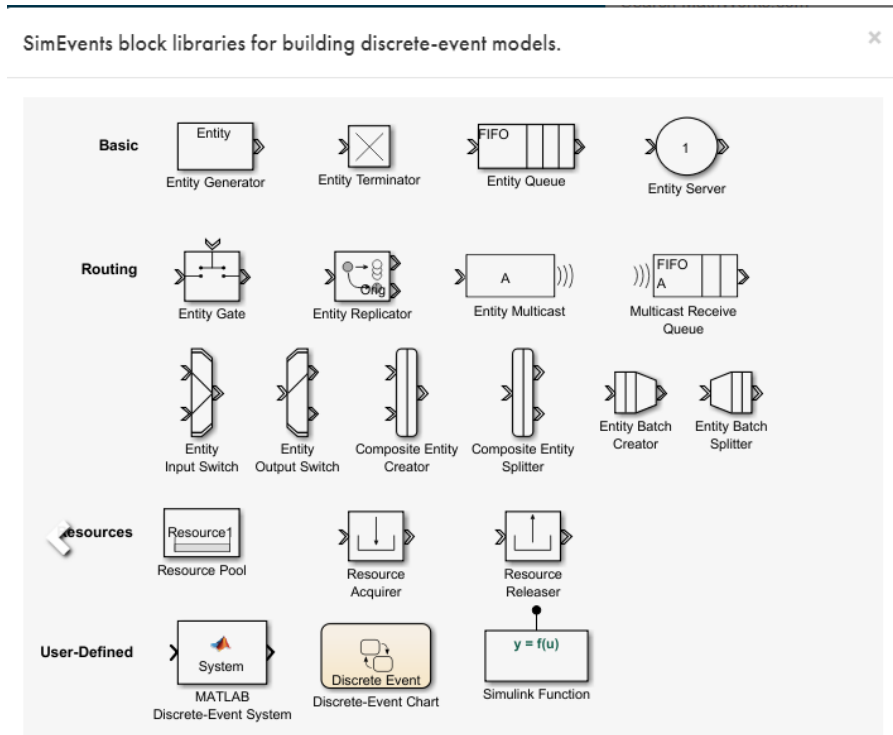


FIGURE 4.12 – Bibliothèques des blocs sous Simulink

Simulink Design Vérifier : capable de vérifier et de valider systématiquement les modèles par la vérification du style de modélisation, la traçabilité des exigences et l'analyse de la couverture des modèles, représentée 4.13.

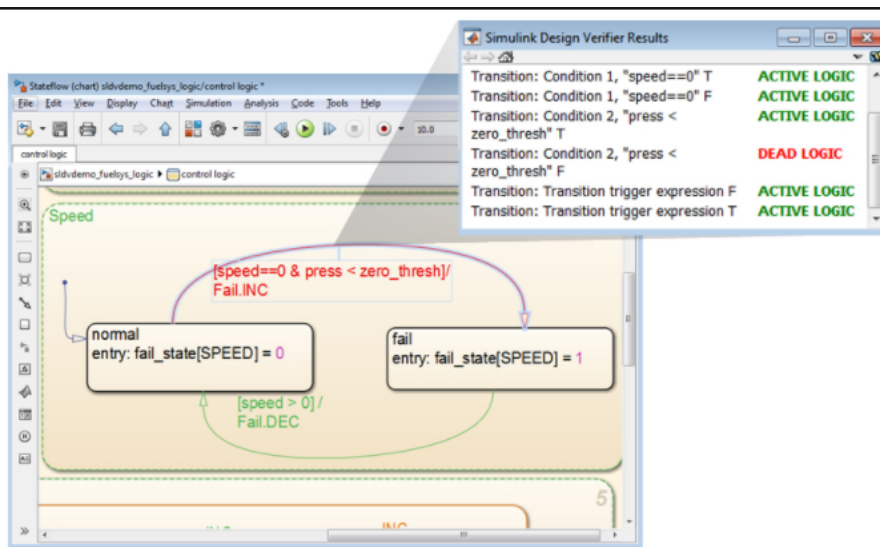


FIGURE 4.13 – Vérification du code Simulink

Simulink real time : Simulink Real-Time vous permet de créer des applications en temps réel à partir de modèles Simulink et de les exécuter sur du matériel informatique cible dédié connecté à votre système physique. Il prend en charge la simulation et les tests en temps réel,

y compris le prototypage rapide de commandes, le prototypage DSP et de systèmes de vision, et la simulation Hardware, représentée 4.14.

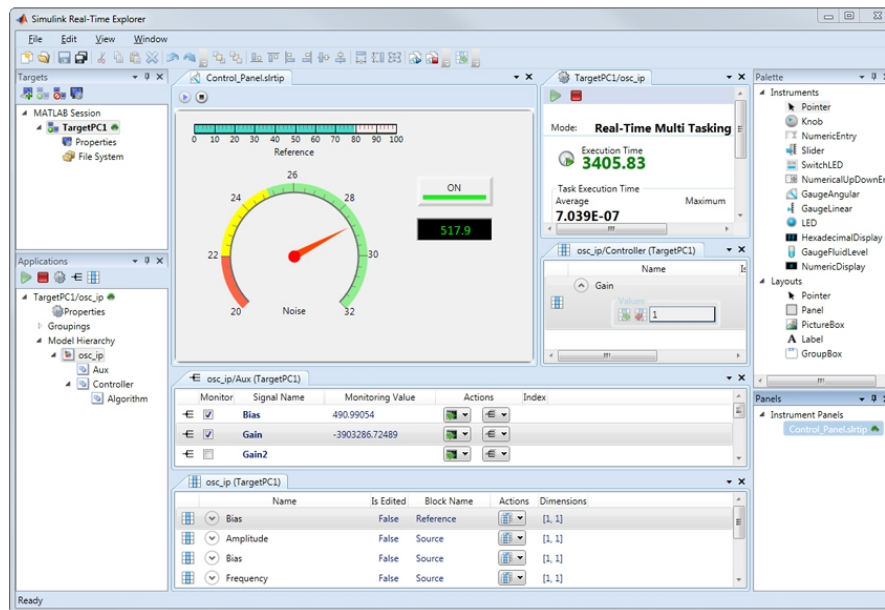


FIGURE 4.14 – Création des objets en temps réel

Avec Simulink Real-Time, vous pouvez étendre vos modèles Simulink avec des blocs pilotes, générer automatiquement des applications en temps réel, définir l'instrumentation et effectuer des exécutions interactives ou automatisées sur un ordinateur cible dédié équipé d'un noyau temps réel, d'un CPU multicœur, d'interfaces E/S et protocoles, et de FPGAs (Field Programmable Gate Array). Les matériels informatiques cibles Simulink Real-Time sont expressément conçus pour travailler ensemble afin de créer des systèmes en temps réel pour les environnements de bureau, de laboratoire et de terrain. Cette bibliothèque est ajoutée en 2017 dans MATLAB[13].

4.5.3 Arduino IDE

L'environnement de développement intégré Arduino (IDE) est une application multi plateforme (pour Windows, MacOS, Linux) écrite dans le langage de programmation Java. Il provient de l'IDE pour les langages Traitement et Câblage. Il inclut un éditeur de code avec des fonctions telles que le copier/coller de texte, la recherche et le remplacement de texte, l'indentation automatique, l'appariement d'accolades et la coloration syntaxique, et fournit des mécanismes simples en un clic pour compiler et télécharger des programmes sur une carte Arduino. Il contient également une zone de message, une console texte, une barre d'outils avec des boutons pour les fonctions communes et une hiérarchie des menus d'opération. Le code source de l'EDI est publié sous la GNU Général Public Licence, version 2, représentée 4.15.

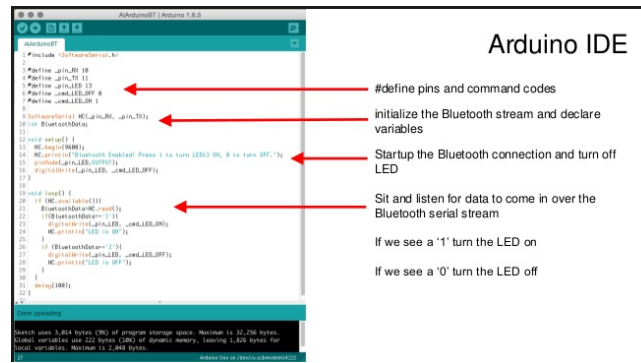


FIGURE 4.15 – L’environnement ARDUINO IDE

4.5.4 Solidwork

SolidWorks est un logiciel de modélisation 3D solide qui permet aux utilisateurs de développer des modèles solides complets dans un environnement simulé pour la conception et l’analyse. Dans SolidWorks, vous pouvez esquisser des idées et expérimenter avec différents designs pour créer des modèles 3D. SolidWorks est utilisé par les étudiants, les concepteurs, les ingénieurs et autres professionnels pour produire des pièces, assemblages et dessins simples et complexes. La conception dans un package de modélisation tel que SolidWorks est bénéfique car elle permet d’économiser du temps, des efforts et de l’argent qui seraient autrement consacrés au prototypage de la conception. Avant de commencer à examiner le logiciel, il est important de comprendre les différents composants qui composent un modèle SolidWorks[21].

Le premier élément de base d’un modèle SolidWorks est **une pièce**.

Les pièces se composent d’une géométrie primitive et de caractéristiques telles que les extrusions, les révolutions, les lofts, les balayages, etc.

Les pièces seront les éléments constitutifs de tous les modèles que vous allez créer.

Le deuxième composant est **l’assemblage**. Les assemblages sont des ensembles de pièces qui sont assemblées d’une manière particulière à l’aide de mats (contraintes).

Tout modèle complexe se compose généralement d’un ou de plusieurs assemblages.

Le troisième et dernier composant de SolidWorks est **le dessin**. Un dessin est la façon typique de représenter un dessin, Un dessin est la façon typique de représenter un modèle 3D afin que n’importe quel ingénieur (ou fabricant) puisse recréer votre pièce. Les dessins sont importants parce qu’ils constituent un moyen standard de partager votre dessin[21].

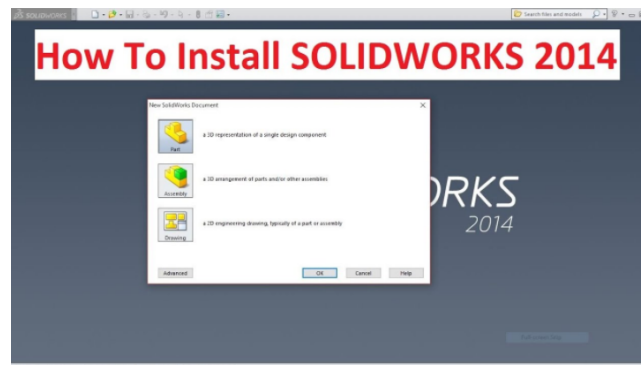


FIGURE 4.16 – L’interface de Solidwork avec les différents composants principales



FIGURE 4.17 – Image réel sous Solidwork

4.6 L'environnement Matlab/Simulink

C'est un logiciel de calcul mathématique pour les ingénieurs et les scientifiques créé par Mathworks.

MATLAB est un environnement de programmation pour le développement d'algorithme, d'analyse de données, de visualisation, et de calcul numérique. En utilisant MATLAB, la résolution des problèmes de calcul complexes se fait plus rapidement qu'avec des langages de programmation traditionnels, tels que C, C++, et le Fortran[12].

Simulink est un environnement pour la simulation multidomaine. Il fournit un environnement graphique interactif et un ensemble de bibliothèques de bloc qui permettent de concevoir, simuler, mettre en application, et examiner une variété de systèmes, tel que les systèmes de communications, de commandes, de traitement des signaux, de traitement visuel, et de traitement d'image[22].

4.7 L'interfaçage Arduino/Matlab/Simulink

Il existe trois possibilités d'interfacer la carte Arduino avec Matlab/Simulink, à savoir :

1. Programmation de la carte Arduino comme carte d'interface.
2. Utilisation du package ArduinoIO.
3. Utilisation du package Arduino Target.

4.7.1 Carte Arduino comme d'interface

Arduino qui permet d'envoyer et d'acquérir des données binaires via le port série (USB) et d'autre part à développer sous Simulink un programme pour traiter ou visualiser ces données.

Configuration de la carte Arduino Mega 2560 : Les fonctions Arduino permettant cette configuration sont les suivantes :

Serial() : Cette fonction est utilisée pour la communication entre la carte Arduino et un ordinateur ou un autre dispositif. Toutes les cartes Arduino ont au moins un port série

(également connue sous le nom d'UART ou USART). Serial, communique sur les pins (0 :RX) et 1 :(TX)) avec l 'ordinateur par l 'intermédiaire d 'USB.

available() : Permet d 'obtenir le nombre de bit (caractères) disponibles pour lire du port série. Ces données sont stockées dans le buffer qui peut sauvegarder 64 bit.

- **read ()** : Permet la lecture des bits entrants sur le port série (acquisition des données).
- **write ()** : Permet l 'écriture des bits sur le port série. (envoi des données)

4.7.1.1 Traitement des données sous Simulink

La bibliothèque Instrument Control Toolbox offre les blocs qui permettent l 'échange des données binaires, représente 4.18 :

Ces blocs sont les suivants :

- **Serial Configuration** : Configuration des paramètres du port série.
- **Serial Send** : Envoi des données binaires via le port série.
- **Serial Receive** : Acquisition des données binaires via le port série.

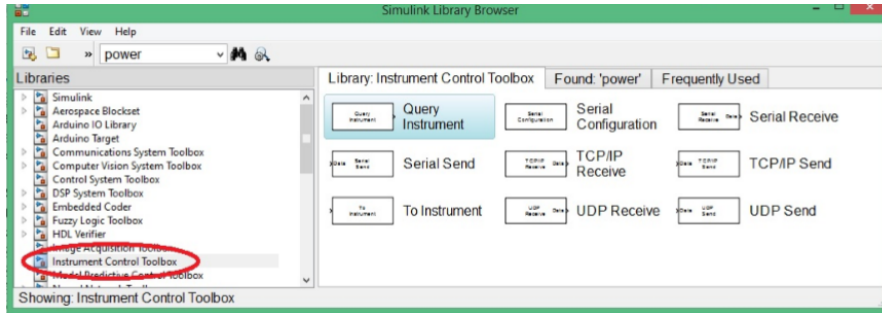


FIGURE 4.18 – Emplacement de la bibliothèque "Instrument Control Toolbox"

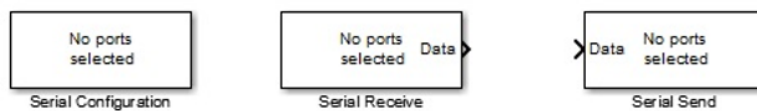


FIGURE 4.19 – Les blocs pour la communication série

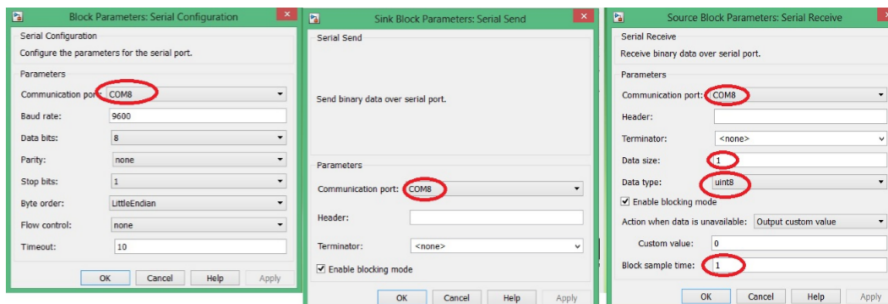


FIGURE 4.20 – Paramétrage des blocs pour la communication série

Les paramètres à configurer sont :

- Communication Port
- Data size
- Data type
- Block sample time.

4.8 ArduinoIO

Cette solution consiste à utiliser la carte arduino comme une interface d'entrées (AnalogInput), sorties (Analog/Digital Output).Ce package permet de communiquer Matlab ou Simulink avec la carte Arduino via un câble USB. Elle consiste à précharger un programme dans la carte Arduino afin que celle-ci fonctionne en serveur[11].

Ce programme consiste d'écouter les requêtes envoyées via la liaison série(USB) et de répondre à ces requêtes en renvoyant l'état d'une entrée ou en modifiant l'état d'une sortie[18]. Ces mêmes entrées/sortie sont vues dans matlab comme des entrées logiques ou analogiques (utilisation du CAN) ou des sorties analogiques (mode PWM)[13].

Pré-chargement du programme dans la carte Arduino

1. Télécharger le package ArduinoIO
2. Décompresser à la racine de votre disque dur, exemples E :ardèuino.
3. Ouvrir le dossier décompressé.
4. Aller vers :ArduinoIO
5. Charger le fichier adiosrv.pde vers le logiciel Arduino.
6. Televerser.

"adiosrv" : est l'abréviation de : Analog and Digital Input and Output Server for MATLAB.

La carte Arduino est maintenant configuré pour être utiliser comme une carte d'interface Entrées/Sorties[22].

4.9 Arduino Target

4.9.1 Installation du package ArduinoIO

1. Lancer Matlab et placer vous dans le répertoire E :
2. Exécuter la commande : install-arduino
3. Fermer et relancer Matlab puis Simulink
4. Dans les bibliothèques se trouvent maintenant les blocs dans Arduino IO Library, représenté 4.21.

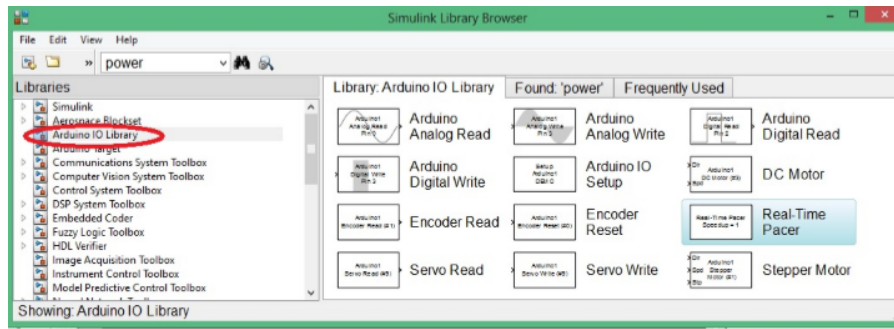


FIGURE 4.21 – Librairies de l'ArduinoIO

4.9.2 Exploitation de la bibliothèque ArduinoIO sous Simulink

Les blocs nécessaires pour notre objectif d'asservissement sont les suivants : représente 4.22



FIGURE 4.22 – Blocs d'ArduinoIO nécessaires pour la commande

Real-Time Pacer : Ce bloc permet de ralentir le temps de simulation de sorte qu'il synchronise avec le temps réel écoulé. Le coefficient de ralentissement est contrôlable par l'intermédiaire du paramètre Speedup.

Arduino IO Setup : Pour configurer sur quel port la carte Arduino est connectée. Pour cela il suffit de voir dans Gestionnaire des périphériques.

Arduino Analog Read : Pour configurer à partir de quel pin on va acquérir les données du capteur.

Arduino Analog Write : Pour configurer à partir de quel pin on va envoyer la commande en PWM vers l'actionneur.

4.9.3 Exploitation du package ArduinoIO sous Matlab

Le package ArduinoIO offre une panoplie de commandes permettant d'écrire un programme sous Matlab (M-file). Pour accéder à ces commandes il faut créer un objet arduino dans l'espace de travail et spécifier le port sur lequel la carte arduino est connecté avec la commande : `a = arduino('port')`; Parmi les commandes qui sont accessibles on retrouve :

- 1- `pinMode()` Exemple : `a.PinMode(11, 'output')` // configurer la pin 11 comme sortie.
- 2- `digitalRead()` Exemple : `val=a.digitalRead(4)` //lecture de l'état de la pin 4 digital-
`Write()` Exemple : `a.digitalWrite(13,0)`; mettre la pin 13 à l'état bas 0V
- 3- `analogRead()` Exemple : `val=a.analogRead(0)` //lecture de la pin 0 de l'ADC .
- 4- `analogWrite()` Exemple : `a.analogWrite(3,10)`; //envoyer sur la pin 10 un signal pwm de rapport cyclique 10/255 [13].

4.10 Arduino Target

Embedded Coder Support Package for Arduino permet de créer des applications Simulink qui vont fonctionner de façon autonome sur la carte Arduino. on dit que la carte Arduino est devenue une cible (Target) et elle peut fonctionner d'une façon autonome (sans avoir recours à Matlab/Simulink).

4.11 les bibliothèques utilisent

Dans notre projet on utilise plusieurs bibliothèques pour faire la communication hardware avec software et on explique chaque bibliothèque brièvement. Donc les bibliothèques utilisées sont :

4.11.1 compilateur de C++

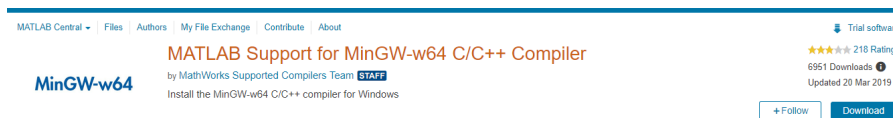


FIGURE 4.23 – Bibliothèque utilisée en Matlab 2018 pour la compilation

Ce compilateur permet de générer ou traduit code matlab ou Simulink à c++, ce dernier envoyer vers IDE pour l'exécutable dans hardware, le compilateur qui ce faire est le Mingw-64. C'est une bibliothèque on peut télécharger comme suivant :

- ouvrir matlab.
- cliquer sur « add on ».
- chercher sur « Mingw-w64 ».
- cliquer sur « download and install ».
- finalement cliquer « open folder » aller directement vers matlab ; représente 4.23.

Remarque 01 : pour les bibliothèques qui on utilisera il faut suivi sur même étapes.

Remarque 02 : l'extension de code matlab est (.m), L'extension de code Simulink est (.slx). Ces codes il faut générer par extension (.c) et (.cpp) finalement (.h) pour l'exécution.

4.11.2 ArduinoIO

Capacités et caractéristiques : Avec MATLAB Support Package for Arduino Hardware, vous pouvez utiliser MATLAB pour communiquer de manière interactive avec une carte Arduino. Le package vous permet d'effectuer des tâches telles que acquérir des données de capteurs analogiques et numériques à partir de votre carte Arduino. Commande d'autres appareils avec sorties numériques et PWM Entraînement de moteurs CC, servomoteurs et moteurs pas à pas (supporte également Adafruit Motor Shield) Accéder aux périphériques et capteurs connectés par I2C ou SPI Communiquez avec une carte Arduino via un câble USB ou sans fil via Wi-Fi. Construire des add-ons personnalisés pour s'interfacer avec des bibliothèques matérielles et logicielles supplémentaires[23].

4.11.3 Embedded Coder Target for Arduino

Cette soumission d'échange de fichiers est une cible personnalisée Embedded Coder pour Arduino. Il utilise la plate-forme Arduino comme matériel d'exemple, mais les exemples qu'il montre peuvent être utilisés pour apprendre et ensuite être appliqués sur n'importe quelle cible personnalisée. D'une manière appliquée - il montre les capacités et les étapes qui exploitent de nombreuses fonctionnalités de Embedded Coder et la création de cible personnalisées[23].

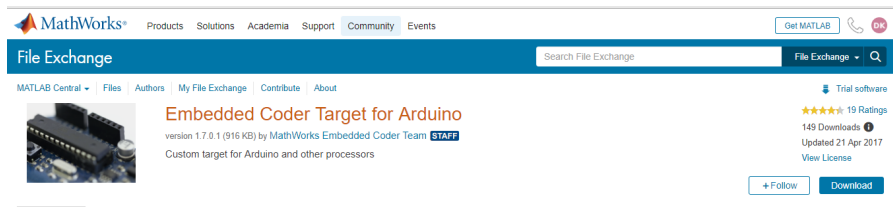


FIGURE 4.24 – Codeur embarqué cible Arduino

4.11.4 RASPLIB

Une boîte à outils Simulink Arduino avec des blocs pour : codeur en quadrature, compas/magnétomètre, baromètre, température, accéléromètre, gyroscope, ultrasons, amplificateur de pilotage (pilote moteur DC), et outils simples de communication série et de traçage. Capteurs I2C supportés : MPU6050, MPU9250, HMC5883, BMP180, BMP280, MS5611, BMI160 (communément trouvé sur GY-521, GY-271, GY-91, GY-87), HCSR04, VL53L0X.



FIGURE 4.25 – Bibliothèque utilisée en Hardware

4.11.5 Afficheur LCD

représente4.26.

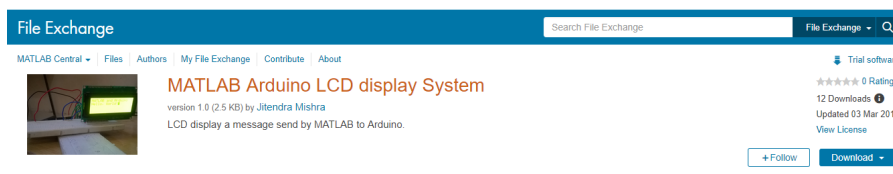


FIGURE 4.26 – LCD Hardware

4.11.6 MPU6050



FIGURE 4.27 – MPU6050

Le module capteur MPU6050 est un dispositif complet de suivi de mouvement à 6 axes. Il combine un gyroscope 3 axes, un accéléromètre 3 axes et un processeur de mouvement numérique, le tout dans un petit emballage. De plus, il est doté d'une fonction supplémentaire de capteur de température sur puce. Il dispose d'une interface bus I2C (protocol de communication) pour communiquer avec les micro contrôleurs.

Il dispose d'un bus auxiliaire I2C pour communiquer avec d'autres capteurs comme un magnétomètre 3 axes, un capteur de pression, etc.

Si le magnétomètre 3 axes est connecté au bus auxiliaire I2C, MPU6050 peut fournir une sortie Motion Fusion 9 axes complète.

Voyons voir MPU6050 à l'intérieur des capteurs.

Mise en Ouvre de l'accéléromètre gyroscope numérique MPU 6050 sur Arduino MEGA 2560 Initialisation et exploitation avec Stateflow représenteci-dessus :

Gyroscope 3 axes

Le MPU6050 est un gyroscope à 3 axes avec technologie MEMS (Micro Electro Mechanical System). Il est utilisé pour détecter la vitesse de rotation le long des axes X, Y, Z comme indiqué sur la figure ci-dessous. - Lorsque les gyroscopes tournent autour de l'un des axes de détection, l'effet de Coriolis provoque une vibration qui est détectée par un MEM dans MPU6050.

- Le signal résultant est amplifié, démodulé et filtré pour produire une tension proportionnelle à la vitesse angulaire.

- Cette tension est numérisée en utilisant l'ADC 16 bits pour échantillonner chaque axe.
- La plage de sortie pleine échelle est de +/- 250, +/- 500, +/- 1000, +/- 2000.
- Il mesure la vitesse angulaire le long de chaque axe en degrés par seconde unité.

Accéléromètre 3 axes

Le MPU6050 est un accéléromètre 3 axes avec technologie MEM (Micro Electro Mechanical). Il sert à détecter l'angle d'inclinaison ou d'inclinaison le long des axes X, Y et Z comme indiqué sur la figure ci-dessous. - L'accélération le long des axes fait dévier la masse mobile.

- Ce déplacement de la plaque mobile (masse) déséquilibre le condensateur différentiel, ce qui entraîne la sortie du capteur. L'amplitude de sortie est proportionnelle à l'accélération.

- L'ADC 16 bits est utilisé pour obtenir une sortie numérisée.
- La gamme complète d'accélération est de +/- 2g, +/- 4g, +/- 8g, +/- 16g.
- Elle est mesurée en g (force de gravité).
- Lorsque l'appareil est placé sur une surface plane, il mesure 0g sur les axes X et Y et +1g sur l'axe Z.

DMP (Digital Motion Processor) Le processeur de mouvement numérique (DMP) intégré est utilisé pour calculer les algorithmes de traitement du mouvement. Il prend les données du gyroscope, de l'accéléromètre et d'autres capteurs tiers tels que le magnétomètre et les

traite. Il fournit des données de mouvement comme le roulis, le tangage, les angles de lacet, le sens du paysage et du portrait, etc. Il minimise les processus de l'hôte dans le calcul des données de mouvement. Les données résultantes peuvent être lues à partir des registres DMP ; la bibliothèque utilise sous MATLAB en figure 4.27[24].

4.11.7 Driver moteur

Ce set contient les blocs de commande moteur pour 3 Arduino Motor Shields :

- 1) Adafruit Motor Shield V1.x (drivers moteurs pas à pas et DC)
- 2) Adafruit Motor Shield V2 (drivers moteurs DC et pas à pas)
- 3) Arduino Motor Shield R3 (drivers moteur DC pour ports A et B) la bibliothèque qui est utilisée dans la figure 4.28.

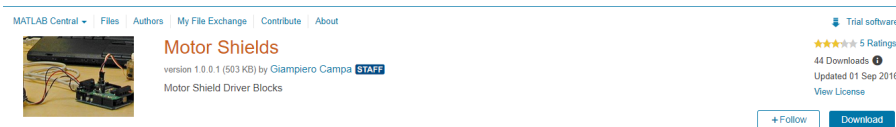


FIGURE 4.28 – Driver du moteur

4.11.8 Bluetooth

ce module pour la communication entre software et hardware sans fils ; représenté 4.29

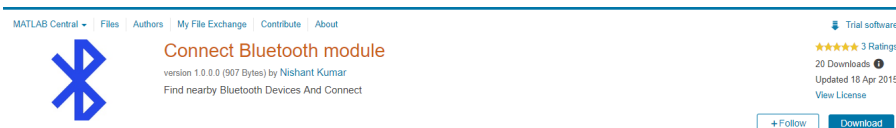


FIGURE 4.29 – Module de communication bluetooth

4.12 traitement des données

4.12.1 présentation de ADC

Le convertisseur analogique-numérique (ADC) embarqué d'Arduino convertit un signal analogique en une valeur numérique.

La carte Arduino Mega 2560 dispose de 16 entrées analogiques notées A0, A1,...A15 mais d'un seul convertisseur analogique/numérique, la durée d'une conversion est de l'ordre de 10^{-6} s. Il a une résolution de 10 bits. La donnée numérique qu'il fournit après conversion est donc comprise entre 0 et 1024. Il n'est pas nécessaire d'initialiser ces entrées analogiques qui n'ont que cette seule fonction. La syntaxe de l'instruction permettant d'acquérir l'entrée analogique est la suivante : **analogRead(pin)** ; pin : la pin sur laquelle on souhaite acquérir le signal analogique ; représenté 4.30.

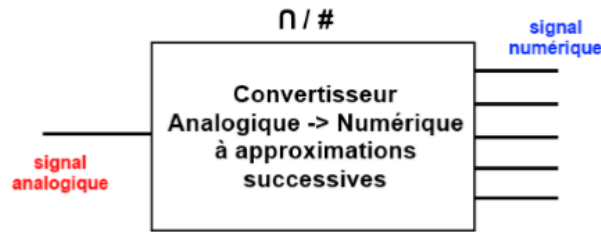


FIGURE 4.30 – Convertisseur Analogique Numérique

4.12.2 Acquisition des donnée

4.12.2.1 Le gyroscope MPU6050

Le MPU6050 communique avec l'Arduino via le protocole I2C dans lequel le MPU6050 est connecté à l'Arduino. Si le module MPU6050 possède une broche 5V, il peut être connecté à la broche 5V de l'Arduino. Si ce n'est pas le cas, il doit être raccordé à la broche 3,3V. Et la broche GND de l'Arduino est connectée à la GND du MPU6050. Le capteur MPU-6050 combine un gyroscope et un accéléromètre. Un accéléromètre fonctionne sur le principe de l'effet piézoélectrique. Les gyroscopes sont utilisés pour mesurer rapidement la vitesse angulaire lors des virages d'un objet. La rotation est généralement mesurée par rapport à l'un des trois axes : lacet, tangage et roulis.

- Lacet : Il mesure la rotation autour d'autres axes par une orientation de montage appropriée.

- Pitch : Il fournit la position en degrés.

- Roulis : En montant sur son côté, l'axe de lacet devient l'axe de roulis.

Branchement avec la carte arduino MEGA 2560 le module MPU-6050 a 8 broches,

INT : Interruption de la broche de sortie numérique.

AD0 : Broche LSB de l'adresse esclave I2C. C'est le 0ème bit de l'adresse esclave 7 bits de l'appareil. Si elle est connectée au V_{cc} , elle est lue comme une logique un et l'adresse de l'esclave change.

XCL : Broche de l'horloge auxiliaire série. Cette broche est utilisée pour connecter d'autres capteurs compatibles avec l'interface I2C SCL à la broche MPU-6050.

XDA : Broche de données série auxiliaire. Cette broche est utilisée pour connecter d'autres capteurs compatibles avec l'interface I2C SDA à MPU-6050.

SCL : Serial Clock pin. Connecter cette broche à la broche SCL du microcontrôleur.

SDA : Broche de données série. Connecter cette broche à la broche SDA du microcontrôleur.

GND : Broche de terre. Connectez cette broche à la terre.

VCC : Broche d'alimentation.

le model Simulink

représente4.31

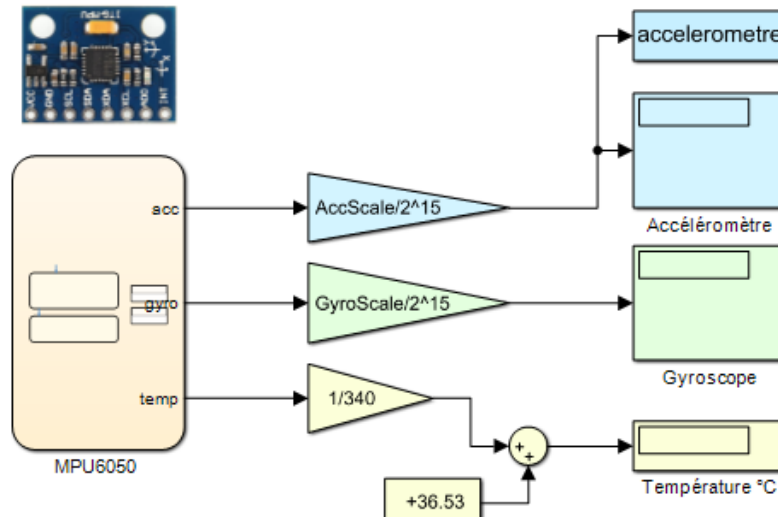


FIGURE 4.31 – Modèle Similink sous MATLAB

4.12.2.2 Encodeur incremental

présentation du module Un codeur est un dispositif qui peut être utilisé pour déterminer la position. La plupart des codeurs utilisent des capteurs optiques pour sortir un train d’impulsions qui peut être décodé pour déterminer la position et direction[24].

Principe de fonctionnement : Si l’arbre du moteur de sortie tourne de 1 tour, la roue codeuse tourne de 24 tours - ce qui signifie que pour un tour de l’arbre de sortie du moteur, il y aura $24 \times 12 = 288$ des impulsions du codeur. L’encodeur dispose de deux canaux A et B. Ces canaux sont le codeur. avec un canal décalé de 90 degrés. En surveillant ces deux canaux, la position et la direction de l’arbre du moteur peuvent être déterminées. Si quadrature le décodage est utilisé, la résolution peut être multipliée par 4. Le bloc codeur fourni avec ce laboratoire effectue le décodage en quadrature, de sorte que le total de l’Encoder la résolution est :

résolution = $24 \times 15 \times 4 = 1440$ voire le modèle Simulink dans la figure 4.32.

le modèle Simulink :

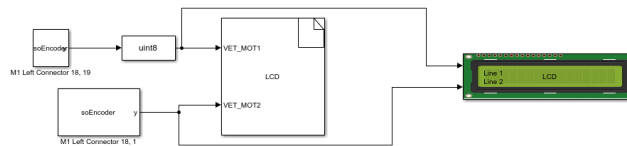


FIGURE 4.32 – Modèle Simulink d’un encodeur

4.12.2.3 Capteur de distance Ultrason HC-SR04

présentation du module Ce module dispose de 4 pins de sortie : VCC , TRIG, ECHO, GND . Les caractéristiques techniques de ce module sont les suivantes :

- alimentation : 5V DC
- Courant de repos : <2mA
- Angle de mesure : < 15°

- Gamme de distance : 2cm – 500 cm
- résolution : 0.3 cm

Branchement avec la carte arduino MEGA 2560

Le processus de mesure de distance est le suivant : donner la pin 'TRIG' une impulsion de niveau haut (5V) durant au moins $10^{-5}s$ et le module démarre sa lecture ; à la fin de la mesure, s'il détecte un objet devant lui, la pin 'ECHO' passe au niveau haut (5V). Et , la distance où se situe obstacle est proportionnelle à la durée de cette impulsion et le branchement sur la carte arduino comme suivant :

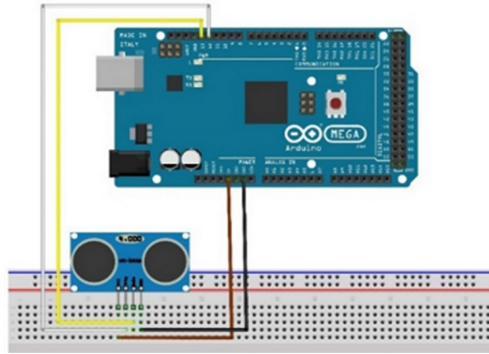


FIGURE 4.33 – Capteur ultra son avec Arduino méga 2560

le modèle Simulink :

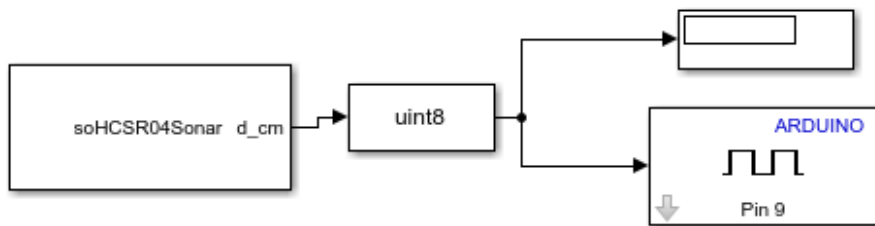


FIGURE 4.34 – Modèle Simulink d'un capteur ultra son

4.12.3 Envoie des données

Dans les contrôleurs numériques qui est utilisé actuellement existe deux types de signaux qui sont envoyés vers l'extérieures, ce sont :

- 1) Signal analogique .
- 2) Signal numérique.

Pour les signaux de sorties analogiques (mode PWM), la carte Arduino MEGA 2560 dispose de 54 pin qui peuvent être utilisées en mode PWM (18 pin), c'est-à-dire en modulation de largeur d'impulsion. Ce sont des signaux logiques binaires de fréquence constante (500Hz) mais de rapport cyclique variable.

Lorsqu'un moteur ou une lampe est alimenté par ce type de tension, tout se passe comme si il était alimenté par une tension continue ajustable entre 0V (rapport cyclique= 0) et 5V (rapport cyclique=255). Ces sorties doivent être initialisées comme des sorties digitales.

La syntaxe de l'instruction permettant de générer le signal PWM est la suivante : **analogWrite(pin, valeur) ;**

- pin : la pin sur la quelle on souhaite envoyer le signal.
- valeur : le rapport cyclique entre 0 et 255.

4.12.3.1 Commande PWM

Lorsqu'un moteur ou une lampe est alimenté par ce type de tension, tout se passe comme si il était alimenté par une tension continue ajustable entre 0V (rapport cyclique= 0) et 5V (rapport cyclique=255). Ces sorties doivent être initialisées comme des sorties digitales. L'équation suivante relie la tension d'entrée à la valeur numérique convertie :

$$\frac{v_{in}}{v_{ref}} \tag{4.1}$$

v_{in} Est la tension d'entrée - typiquement d'un capteur.

v_{ref} Est la tension de référence (généralement 5v ou 3,3v).

dans notre cas on utilisera un signal PWM pour commander la vitesse de rotation de notre robot par un driver qui est appelé 'l293D', ce dernier utilise pour commander la vitesse et le sens de rotation d'un moteur à courant continu.

4.13 Détermination de l'angle à partir des données d'accéléromètre et de gyroscope

Dans ce projet, les données du gyroscope ont été utilisées pour suivre la rotation angulaire, mais l'angle mesuré peut «dériver», même lorsque la planche ne bouge pas. Dans ce projet, l'accéléromètre sera utilisé pour calculer l'angle, et les résultats du gyroscope et de l'accéléromètre seront combinés dans un filtre complémentaire pour obtenir les avantages des deux[19].

4.13.1 Calculer l'angle de position angulaire avec les données du gyroscope

Tout d'abord, on crée un nouveau programme de mesure angulaire à l'aide des données gyroscopiques. On crée le diagramme Simulink dans la figure 4.32.

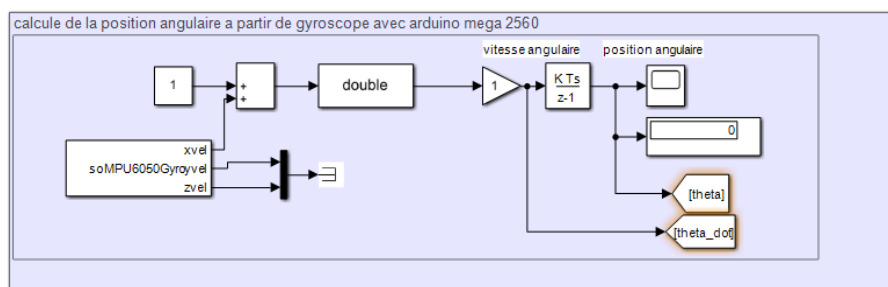


FIGURE 4.35 – Modèle Simulink de position angulaire à l'aide d'un gyroscope

-On utilise le bloc du gyroscope de notre système matériel. nous pouvons voir dans la figure 4.32 que l'entrée du pilote du gyroscope est en cours d'intégration afin de produire la sortie finale de Scope. Cela signifie également que si le matériel ne lit pas exactement zéro lorsqu'il est au repos, les valeurs faibles entraîneront une augmentation ou une diminution régulière de la lecture. C'est le biais du gyroscope. Un terme de biais -C- est utilisé pour supprimer avant

l'intégration. Vous devez manuellement «calibrer» le système en trouvant expérimentalement une valeur pour éliminer ce biais. Même après avoir éliminé le biais, le biais initial changera avec le temps. Ceci est connu sous le nom de «dérive» du gyroscope et il faudra éventuellement faire quelque chose pour le corriger pendant de plus longues périodes (minutes). Le bloc Gyroscope est un bloc spécial. Double-cliquez sur le bloc Gyroscope ; représente4.32.

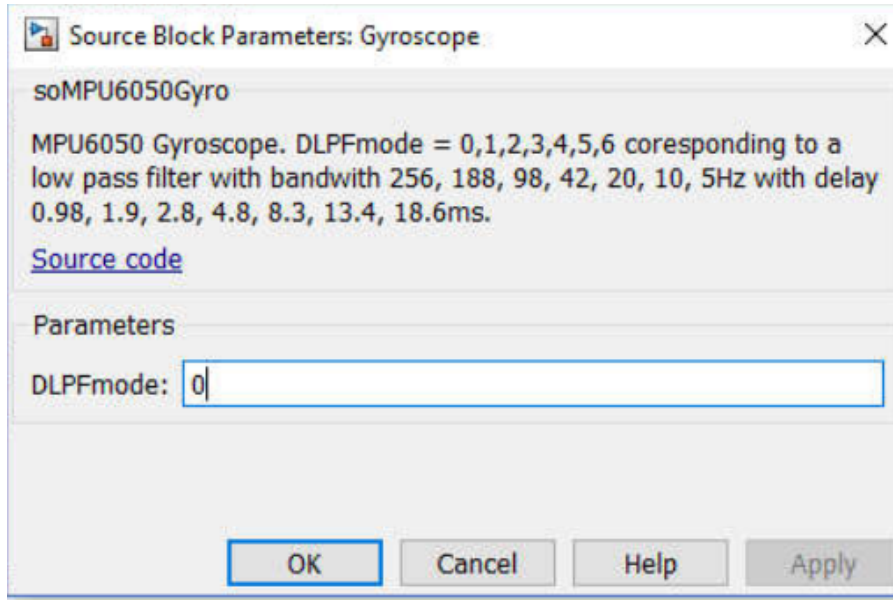


FIGURE 4.36 – Bloc de paramètres d'un gyroscope

4.13.2 Calculer de la position angulaire avec les données de l'accéléromètre

Dans cette partie, l'angle de la carte est calculé à l'aide des données de l'accéléromètre en observant composantes de la force gravitationnelle constante (qui est toujours "vers le bas") sur chacun des accéléromètres axes.

Les données de 2 axes de l'accéléromètre (Y_{acc} et Z_{acc}) sont utilisées pour calculer les valeurs de position angulaires. L'accélération vers le bas depuis la Terre sera toujours assez constante : si la planche mesure un composant Z de $9,81 \text{ m / s}^2$ mais rien dans le composant Y, il sait qu'il est couché à plat. Si l'accélération Z diminue, mais l'accéléromètre Y mesure une partie de cette vers le bas, nous pouvons calculer l'angle à l'aide d'un bloc arctangent. Créez le diagramme Simulink qui represent dans la figure 4.33.

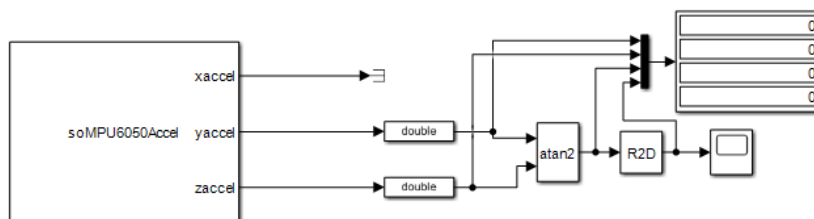


FIGURE 4.37 – Position angulaire à l'aide d'un accéléromètre

4.13.3 Création du filtre complémentaire

Les deux circuits de mesure précédents peuvent être combinés dans un système de mesure utiliser les caractéristiques des deux capteurs : le gyroscope peut mesurer des changements rapides de rotation, mais a une erreur d'état stable. L'accéléromètre peut obtenir avec précision l'état d'équilibre angle[23].

Un procédé simple pour combiner ces deux mesures est appelé filtre complémentaire. On crée les diagrammes Simulink dans la figure 4.35.

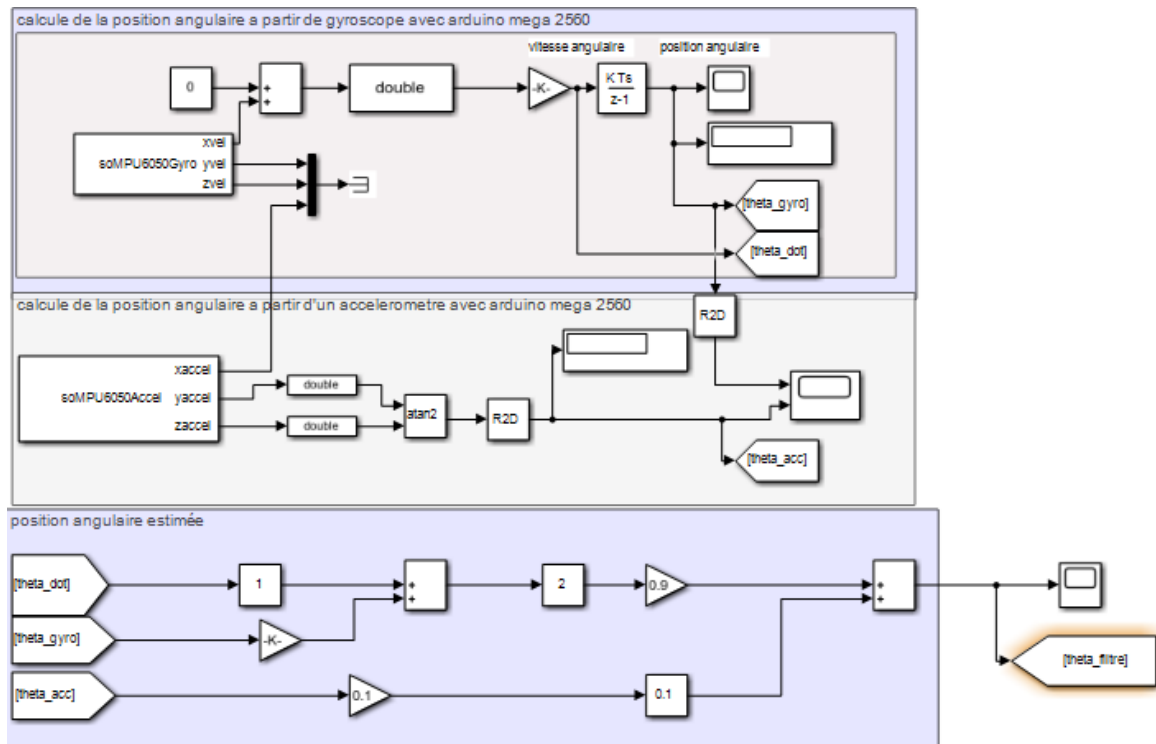


FIGURE 4.38 – Modèle Simulink d'un angle estimé

Remarque : les blocs «GoTo» permettent de connecter des signaux sans avoir les «fils» dessinés sur le diagramme Simulink. Le gyroscope n'a besoin que d'une petite correction pour compenser l'accumulation de dérive. Donc, la plupart de l'angle est mesuré à partir du gyroscope.

Maintenant, sélectionnez les composants du diagramme détaillé ci-dessus, puis cliquez avec le bouton droit de la souris «Créer un sous-système à partir de la sélection». Cela créera un bloc pouvant être facilement inséré dans le code pour d'autres applications.

Utilisation du filtre complémentaire pour le contrôle de position Maintenant, nous allons revoir un diagramme de contrôle de position et appliquer le filtre complémentaire que vous avez choisi. viens de créer. Dans ce cas, nous mesurerons la position, puis sortirons ces données vers un contrôleur PID et un pilote de moteur. Que pensez-vous que le système résultant va accomplir? On crée le diagramme Simulink pour le contrôle de position angulaire en boucle fermée, représenté 4.35.

4.13.4 Contrôle de la position linéaire à l'aide d'un encodeur incrémental

Pour calculer la position et la vitesse linéaire de notre robot on utilisera un capteur de position appelé encodeur, ce dernier très facile d'utiliser dans le modèle Simulink, représente 4.36 : Modèle Simulink

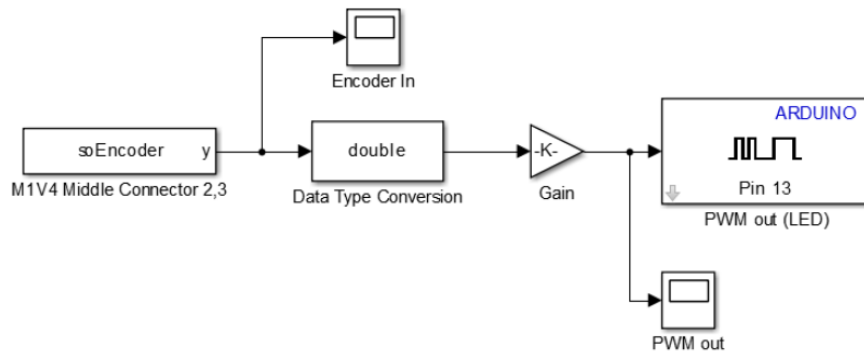


FIGURE 4.39 – Position linéaire en temps réel

Il est plus facile de démarrer avec le modèle Simulink configuré pour effectuer du matériel. Dossiers expérimentaux Nos matériaux doivent être utilisés à cet effet. Installez RASPLib, cliquez avec le bouton droit de la souris sur le package de support de Rensselaer Arduino pour ouvrir la bibliothèque, sous Démonstrations, puis ouvrez MinSegShield M1V4

4.13.5 Capteur de distance Ultrason HC-SR04 :

Ce module dispose de 4 pins de sortie : VCC , TRIG, ECHO, GND .

Les caractéristiques techniques de ce module sont les suivantes :

- alimentation : 5V DC
- Courant de repos : <2mA
- Angle de mesure : 15°
- Gamme de distance : 2cm – 500 cm
- résolution : 0.3 cm qui représente dans la figure 4.37



FIGURE 4.40 – Capteur ultra sonic

Le processus de mesure de distance est le suivant : donner la pin "TRIG" une impulsion de niveau haut (5V) durant au moins $10^{-6}s$ et le module démarre sa lecture ; à la fin de la mesure, s'il détecte un objet devant lui, la pin 'ECHO' passe au niveau haut (5V). et le branchement sur la carte arduino qui représente dans la figure 4.38 :

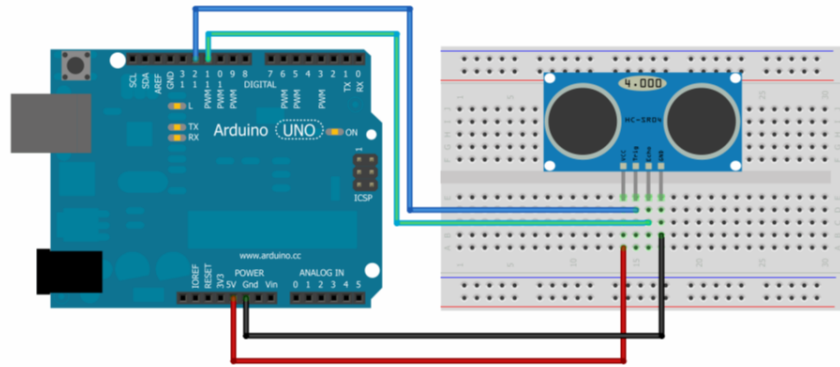


FIGURE 4.41 – Branchement du HC-SR04 avec la carte Arduino UNO

4.13.6 commande LQR

La stabilité des systèmes non linéaires est très difficile commander pour cela on utilisera linéarisation des systèmes non linéaire, ces dernier qui donne un commande appelle LQR, on applique cette commande sur notre robot et on envoyés ces signaux sur une carte arduino pour commander le robot.

Cette figure représente un modèle Simulink de la commande LQR, on trouve deux capteurs dans cette modèle :

- 1) Capteur de position : ce capteur permet de donner les valeurs de position linéaire et la vitesse linéaire.
- 2) MPU6050 : ce capteur permet de trouver les rotations angulaires sur trois axes (roulis, tangage, lacet) c.-à-d. les positions angulaires plus les vitesses angulaires

4.13.7 commande PID

Dans ce sous partie on utilise la commande précédent avec un autre régulateur « PID »

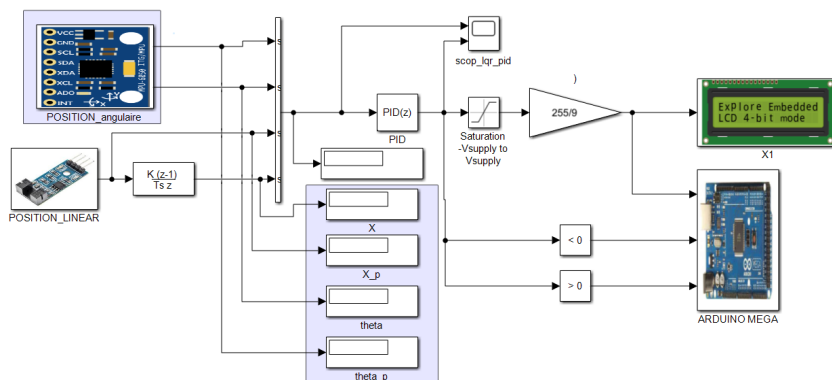


FIGURE 4.42 – Commande LQR avec PID

4.14 Présentation du robot sous Solidwork

Dans notre projet on utilisera logicielles de conception appelle « solidwork » ce dernier qui permettent de concept les robots après la réalisation pour gagner le temps et démunie le cout

de réalisation, et on peut communiquer avec MATLAB pour observer l'allure de notre robot, représenté 4.43 :

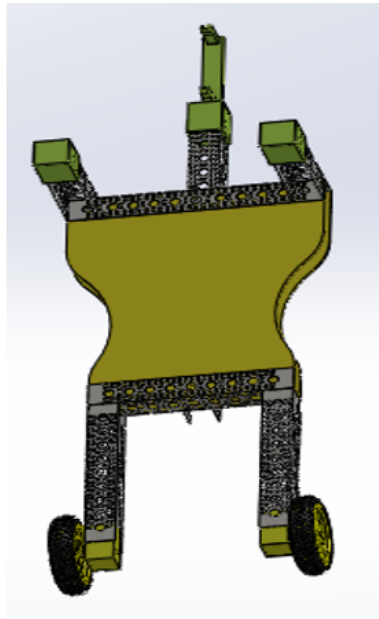


FIGURE 4.43 – Robot auto balancier sous Solidwork

4.14.1 l'interface de commande

Cette interface permet de contrôler notre robot en temps réel, elle contient des leds pour avec les quelles on peut s'avoir la position du robot par rapport au vertical, et un slider pour varier la position et un scope pour visualiser le comportement du système.

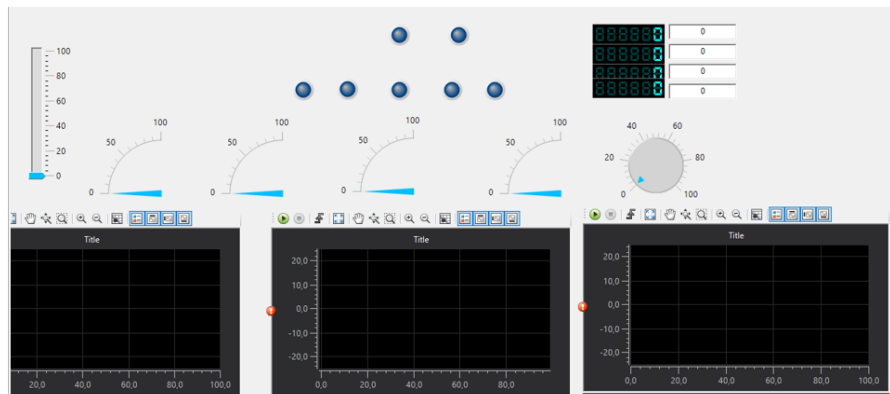


FIGURE 4.44 – L'interface de commande en temps réel

4.15 Réalisation du robot

4.15.1 Les étapes de réalisation

1) **communication MPU6050 avec MATLAB** Dans le première étape on teste la communication entre le gyroscope/accéléromètre et MATLAB .

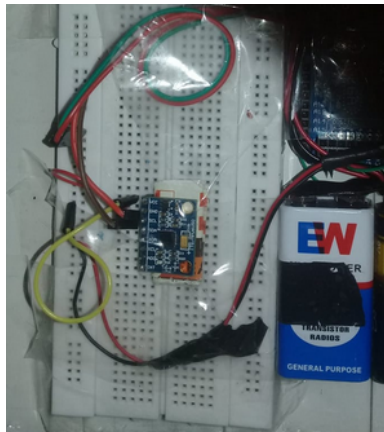


FIGURE 4.45 – Test de la communication entre MPU6050 et MATLAB

2) Raccordement les deux roues



FIGURE 4.46 – Raccordement des deux roues avec le chassie

3) **Raccordement les roues avec l'arduino** Dans cette étape on assemble le robot et on fait la premier test.



FIGURE 4.47 – Assemblage du Robot

Robot final (version 1)

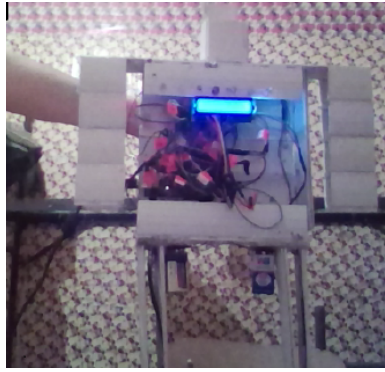


FIGURE 4.48 – Robot final version 1

Robot final (version 2)

Nous avons réalisé ce robot dans laboratoire de recherche d'Énergie et des Systèmes Intelligent (LESI) à l'université DJILALI BOUNAAMA KHEMIS MILIANA.



FIGURE 4.49 – Robot final version 2

4.16 Présentation du robot

Le robot a la forme d'un humanoïde avec deux roues, de 56cm d'hauteur et qui ne dépasse pas 1kg de poids. Il est constitué d'un capteur de position MH-sensor-series « Encodeur » et des leds. Le capteur est fixé sur l'arbre du moteur. Un module de communication entre MATLAB et le robot est utilisé pour nous permettre la communication à distance avec notre système, ce module est le « H-05 » module Bluetooth. Le driver «l298N» est L'élément principale pour commander le sens de rotation et la vitesse de rotation des roues du robot, il est connecté avec l'arduino méga 2560 et le moteur.

4.17 Conclusion

Dans ce dernier chapitre nous avons utilisé deux logiciels très importants dans le domaine de la robotique pour communiquer entre le software et le hardware, le logiciel Solidwork, qui est très puissant dans la conception des robots et MATLAB pour commander les systèmes en temps réel via des cartes d'acquisitions. nous avons rencontré pas mal de contraintes lors de la réalisation de notre robots.

Conclusion Générale

Notre travail a mis en évidence l'un des axes principaux de la recherche dans la robotique mobile dans l'environnement naturel, dans un souci d'exploiter nos résultats pour la réalisation des systèmes robotisés pour l'aide à la marche des personnes handicapées.

Nous avons traité d'une façon profonde le sujet étudié, d'où nous avons commencé à donner un état de l'art sur les systèmes sous-actionnés, ou nous avons focalisé notre intérêt au robot auto balancier à deux roues. Une modélisation détaillée du système a été présentée. Après une simulation de deux lois de commande dans le cas continu et dans le cas discret ont été réalisées sous l'environnement MATLAB/Simulink, où nous avons trouvé des résultats satisfaisants, où les grandeurs de sortie convergent.

Dans la dernière partie qui nous a pris beaucoup de temps et d'efforts, nous avons réalisé notre prototype de robot auto balancier où nous avons utilisé les moyens existants. Au début, nous avons obtenu un comportement encourageant de notre système, après et à cause de la qualité des matériels nous n'avons pas pu avoir des résultats meilleurs mais nous pouvons surmonter tout ça par des moteurs plus puissants, et en prenant en considération les frottements qui affectent le système.

Entant que sortant de master en automatique, ce projet de fin d'étude m'a permis d'approfondir mes connaissances théoriques et d'améliorer mon niveau du côté pratique grâce à la réalisation de notre robot.

On note que le dispositif de commande électronique est opérationnel dans sa première version. Il pourra servir d'un robot mobile sous actionnée de tests au profit des futurs candidats de Master.

En perspective, nous recommandons la prise en compte des points suivants :

- effectuer des améliorations de balancement avec contrôle optimale de trajectoire.
- Effectuer des améliorations en intégrant d'autres fonctionnalités de détection et le traitement et vision.
- Améliorer la technique de commande sans fil avec MATLAB (pour rendre le contrôleur d'une commande par (Pc), et commande autonome).
- effectuer des simulations sous SolidWork avec MATLAB pour diminuer le coût de réalisation.

Annexe A

Programmation de la commande linéaire

Le code Matlab pour obtenir les valeurs du gain K par placement des pôles et LQR :

```
1 %This function creates the state space for a two-wheeled open loop balancer
2 % Parametres du moteur
3 R=3; % resistance d'un moteur
4 Km=0.006123; % couple moteur
5 Ke=0.0069203; % couple a appliquer
6 %les parametres du pendule
7 Mp=0.3; % la masse du pendule
8 l=0.12; % la longueur du pendule
9 Ip=0.0041; % moment d'inertie du pendule
10 % les parametres des roues
11 Mw=0.03; % la masse des roues
12 Iw=0.000039; % moment d'inertie des roues
13 r=0.035; % le rayon de roues
14 %*****
15 g=9.81; %la gravite
16 %*****
17 beta = (2*Mw+(2*Iw/r^2)+Mp)
18 alpha = (Ip*beta + 2*Mp*l^2*(Mw + Iw/r^2))
19 % *****
20 % declaration des matrices d'etat
21 A= [0 1 0 0;0 (2*Km*Ke*(Mp*l*r-Ip-Mp*l^2))/(R*r^2*alpha) (Mp^2*g*l^2)/alpha 0;
22 0 0 0 1; 0 (2*Km*Ke*(r*beta - Mp*l))/(R*r^2*alpha) (Mp*g*l*beta)/alpha 0]
23 B = [ 0;
24 (2*Km*(Ip + Mp*l^2 - Mp*l*r))/(R*r*alpha);
25 0;
26 (2*Km*(Mp*l-r*beta))/(R*r*alpha)]
27 C = [1 0 0 0;0 0 1 0]
28 D = [0;
29 0]
30 a22=(2*Km*Ke*(Mp*l*r-Ip-Mp*l^2))/(R*r^2*alpha)
31 a23= (Mp^2*g*l^2)/alpha
32 a42=(2*Km*Ke*(r*beta - Mp*l))/(R*r^2*alpha)
33 a43=(Mp*g*l*beta)/alpha
34 b12=(2*Km*(Ip + Mp*l^2 - Mp*l*r))/(R*r*alpha)
35 b14=(2*Km*(Mp*l-r*beta))/(R*r*alpha)
36
37 % Commande par placement de pole
38 %*****
39 % poles=[-3;-3;-3;-3]
40 % poles= [-10 -10 -20 -30]
41 % poles= [-10 -50 -1 -30] bien
42 poles= [-10 -60 -1 -30]
43 K_place=acker(A,B,poles)
44 ylabel('Position[m], Angle[rad]')
45 xlabel('Temps[s]')
46 legend(' position', 'angle de rotation')
47 % Commande lineaire quadratique
48 a=5,b=10
49 % a =500;b =5000; good
50 Q=[1 0 0 0;0 a 0 0;0 0 1 0;0 0 0 0 b]
```

```

51 R=0.1
52 K_lqr=lqr(A,B,Q,R)
53 % ylabel('Position[m], Angle[rad]')
54 % xlabel('Temps[s]')
55 % legend(' position','angle de rotation')
56 % x = 1000; y =10000;
57 % Q = [x 0 0 0;
58 % 0 1 0 0;
59 % 0 0 y 0;
60 % 0 0 0 1];
61 % R =0.01;
62 % K_lqr=lqr(A,B,Q,R)

```

Modèle Simulink de la commande LQR

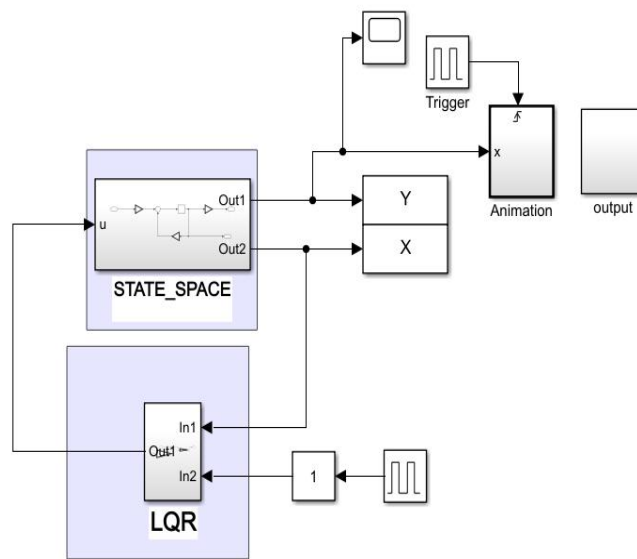


FIGURE 50 – Modèle simulink de la commande LQR d'un robot auto balancier

Le régulateur

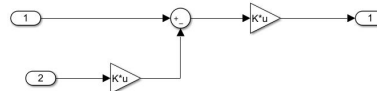


FIGURE 51 – Régulateur utilisé avec le système

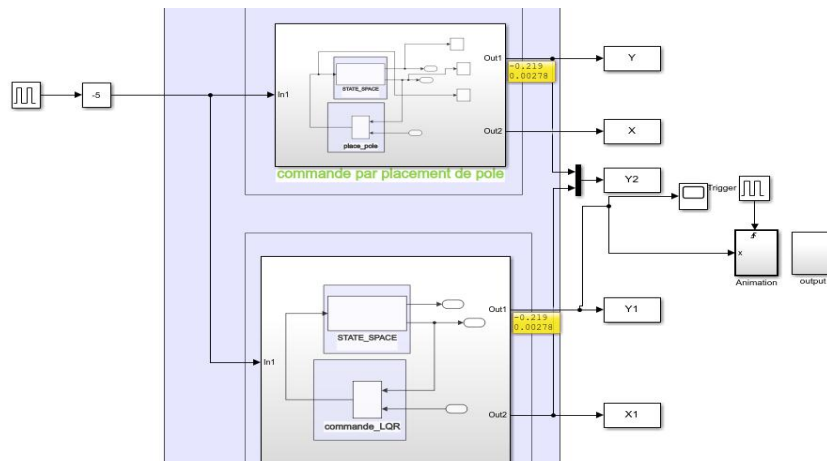


FIGURE 52 – Modèle Simulink du système

L'animation sous Matlab

```

1 function call_plot2(q)
2 %#codegen
3 % Define the extrinsic functions
4
5 coder.extrinsic('pause')
6
7 if ~ishandle(1) %Initialize Figure
8     figure(1);
9 end
10
11 clf
12
13 PlotS2A(q);
14 pause(0.02)
15 end
16
17 function PlotS2A(q)
18
19 q1 = q(1);
20 q2 = q(2);
21
22 r = 0.4; % Wheel radius
23 l1 = 2;
24 p0 = [q1;0;0]; % Base
25 Rotz_q2 = RotZ(q2);
26 p2 = Rotz_q2*[0;1;0] * l1 + p0; % Tip of the Arm / Head
27 AngleWheel = -q1/r;
28 RotWheel = RotZ(AngleWheel);
29
30 %% Arm
31 line([p0(1);p2(1)], [p0(2);p2(2)], 'LineWidth', 10, 'Color', [1 0.4 0.1]);
32 hold on
33
34 %% Cart
35
36 rectangle('Position', [p0(1)-r, p0(2)-0.2, 2*r, 2*r], ...
37           'Curvature', [1 1], 'FaceColor', [0.4 0.5 0.7])
38 % Wheel
39 p_wheelCenter = p0+[0;r-0.2;0];
40 p1_wheel = p_wheelCenter + RotWheel*[0;r*0.95;0];
41 p2_wheel = p_wheelCenter + RotWheel*RotZ(120*pi/180)*[0;r*0.95;0];
42 p3_wheel = p_wheelCenter + RotWheel*RotZ(-120*pi/180)*[0;r*0.95;0];
43
44 plot(p_wheelCenter(1), p_wheelCenter(2), 'o', 'LineWidth', 7, 'Color', [0.3 0.2 0.7]*0.3)
45 line([p_wheelCenter(1);p1_wheel(1)], [p_wheelCenter(2);p1_wheel(2)], ...
46      'LineWidth', 5, 'Color', [0.4 0.5 0.7]*0.3);
47 line([p_wheelCenter(1);p2_wheel(1)], [p_wheelCenter(2);p2_wheel(2)], ...
48      'LineWidth', 5, 'Color', [0.4 0.5 0.7]*0.3);
49 line([p_wheelCenter(1);p3_wheel(1)], [p_wheelCenter(2);p3_wheel(2)], ...
50      'LineWidth', 5, 'Color', [0.4 0.5 0.7]*0.3);

```

```
51 %% Head
52 p_head_left = p2-Rotz_q2*[0.2;0;0];
53 p_head_right = p2+Rotz_q2*[0.2;0;0];
54 plot(p_head_right(1),p_head_right(2),'*','LineWidth',5, 'Color',[1 0.5 0.1])
55 line([p_head_left(1);p_head_right(1)], [p_head_left(2);p_head_right(2)],...
56      'LineWidth',10, 'Color',[0.4 0.4 0.5]*.3);
57 hold on
58
59 %% Ground
60 rectangle('Position',[-4,-0.2-0.1,8,0.1],'FaceColor',[0.6 0.6 0.7]*0.36)
61 %% Axis
62 % grid on
63 axis([-4 4 -3 3])
64 xlabel('X')
65 ylabel('Y')
66
67
68 end
69
70 function R = RotZ(qz)
71     R = [cos(qz) -sin(qz) 0;sin(qz) cos(qz) 0; 0 0 1];
72 end
```

Annexe B

Bloqueur d'ordre zéros

Dans le schéma ci-dessus présente le système de commande numérique, nous voyons que le système contient à la fois des parties discrètes et continues. Typiquement, le système contrôlé génère et répond à des signaux en temps continu, tandis que l'algorithme de contrôle peut être implémenté sur un ordinateur numérique. Lors de la conception d'un système de commande numérique, nous devons d'abord trouver l'équivalent discret de la partie continue du système.

Cette technique de contrôle est représentée par la figure ci-dessous

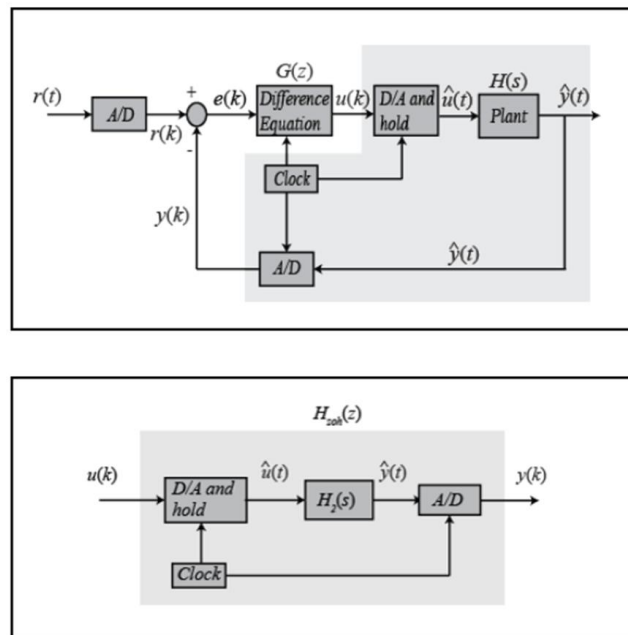


FIGURE 53 – Schéma synoptique de la commande numérique

L'horloge connectée aux convertisseurs N/A et A/N fournit une impulsion toutes les T secondes et chaque N/A et A/N envoie un signal uniquement lorsque l'impulsion arrive. Le but d'avoir cette impulsion est d'exiger que $H_{zoh}(z)$ n'agisse que sur des échantillons d'entrée périodiques $u(k)$, et produise des sorties périodiques $y(k)$ seulement à des intervalles de temps discrets ; ainsi, $H_{zoh}(z)$ peut être réalisé comme une fonction distincte.

Méthodologie

La méthodologie de conception de la commande est la suivante. Nous voulons trouver une fonction discrète $H_{zoh}(z)$ de sorte que pour une entrée constante par morceaux dans le système continu $H(s)$, la sortie échantillonnée du système continu égale la sortie discrète. Supposons que le signal $u(k)$ représente un échantillon du signal d'entrée. Il existe des techniques pour prendre cet échantillon $u(k)$ et pour le tenir pour produire un signal continu $\hat{u}(t)$. Le schéma ci-dessous montre un exemple où le signal continu $\hat{u}(t)$ est maintenu constant à chaque échantillon $u(k)$ sur l'intervalle kT à $(k+1)T$. Cette opération de maintien de la constante $\hat{u}(t)$ sur la période d'échantillonnage s'appelle une retenue d'ordre zéro.

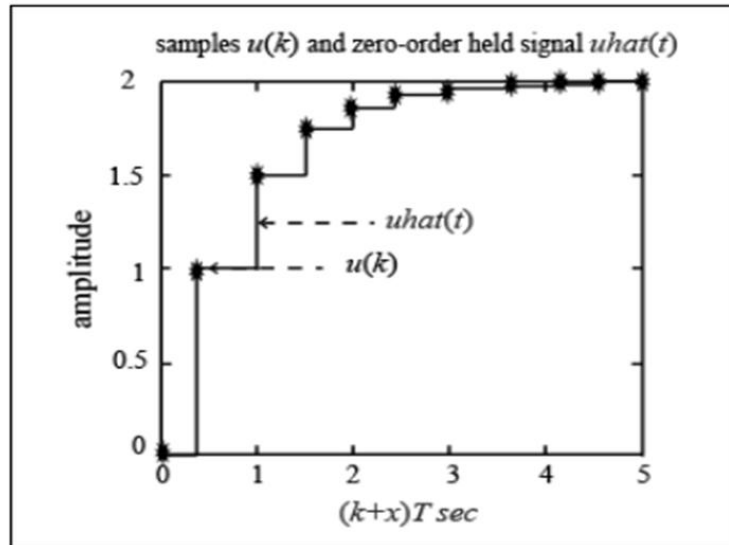


FIGURE 54 – Discrétisation d'un système continu

Le signal retenu $\hat{u}(t)$ est alors passé à $H_2(s)$ et l'A/D produit la sortie $y(k)$ qui sera le même signal pièce par pièce que si le signal discret $u(k)$ avait été passé par $H_{zoh}(z)$ pour produire la sortie $y(k)$.

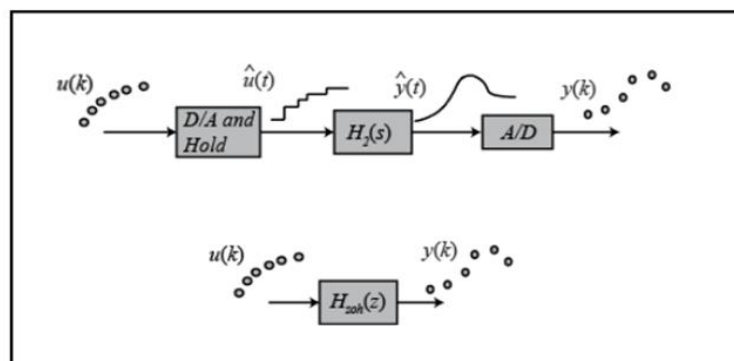


FIGURE 55 – L'état des systèmes numériques

Nous allons maintenant redessiner le schéma, en remplaçant la partie continue du système par $H_{zoh}(z)$.

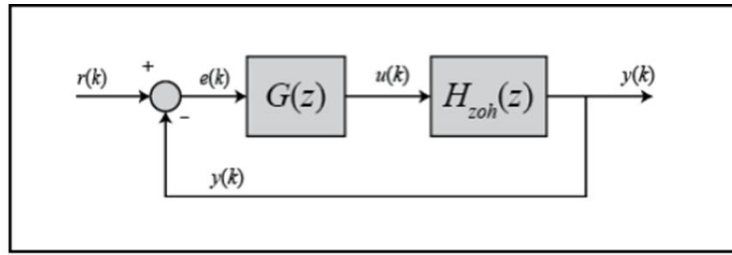


FIGURE 56 – Commande numérique en boucle fermée

Nous pouvons maintenant concevoir un système de contrôle numérique.

Représentation d'état discrète

La représentation d'état discrète d'un système linéaire à coefficients constants, d'ordre n , possédant r entrées et m sorties s'écrit :

$$\begin{cases} X_{(i+1)} = AX_{(i)} + BU_{(i)} \\ Y_{(i+1)} = CX_{(i)} + DU_{(i)} \end{cases} \quad (2)$$

avec :

i l'instant d'échantillonnage

Le diagramme structurel de ce système multivariable est alors :

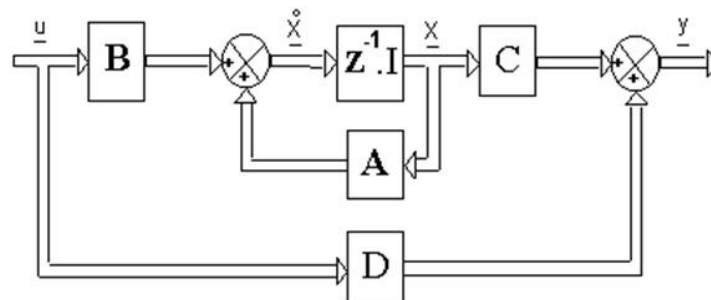


FIGURE 57 – Schéma fonctionnel d'une commande numérique

Commande Linéaire Quadratique

La commande linéaire quadratique est issue de l'application des théorèmes de stabilité de Lyapunov utilisés sur les systèmes linéaires. Soit le système multivariable à r entrées, m sorties et d'ordre n dont la représentation d'état discrète est la suivante :

$$\begin{cases} X_{(i+1)} = AX_{(i)} + BU_{(i)} + f \\ Y_{(i+1)} = CX_{(i)} + DU_{(i)} \end{cases} \quad (3)$$

avec f le vecteur des perturbations.

On recherche la commande $u(i)$ permettant de minimiser, sur un horizon N considéré infini, le critère quadratique suivant :

$$J = \sum_{i=0}^{N-1} (e_{(i)}^T Q e_{(i)} + u_{(i)}^T R u_{(i)}) \quad (4)$$

Avec :

$e(i)$: le vecteur d'écart entre les consignes z et les sorties $y(i)$

La solution optimale est de la forme :

$$\begin{cases} u(i) = \mu(i) - L(i) \cdot X \\ L(i) = [R + B^T \cdot K_{(i+1)} \cdot B]^{-1} \cdot B^T \cdot K_{(i+1)} \cdot A \\ \mu(i) = -[R + B^T \cdot K_{(i+1)} \cdot B]^{-1} \cdot B^T \cdot [g_{(i+1)} + K_{(i+1)} \cdot f] \\ K(i) = L_{(i)}^T R L_{(i)} + [A - B L_{(i)}]^T \cdot K_{(i+1)} [A - B L_{(i)} + C^T \cdot Q \cdot C \\ G(i) = [A - B L_{(i)}] \cdot [g_{(i+1)} + K_{(i)} \cdot f] - C^T \cdot Q \cdot Z \end{cases}$$

Ces équations de récurrences sont calculées en temps inverse, à partir des valeurs finales $K(N) = 0$ et $g(N) = 0$. Les valeurs de la matrice L et du vecteur m dépendent directement du choix des matrices Q et R qui agissent sur la rapidité et la robustesse de la commande. Une première évaluation de ces matrices peut être effectuée avec la règle de Bryson /De Larminat 93/ qui donne :

$$r(i) = [1/\text{sup}(u(i))]^2, s(i) = [1/\text{sup}(y(i))]^2 \text{ avec } Q = C^T \cdot S \cdot C \quad (5)$$

Le choix des matrices de pondération est ensuite affiné par essais et erreurs en simulation. La commande par retour d'état suppose que le vecteur d'état $X(i)$ soit accessible. En règle générale seule les entrées et sorties du système sont disponibles ; le vecteur d'état peut être alors remplacé par son estimation $\hat{X}(i)$ sans modifier les propriétés de la commande. L'estimation est obtenue grâce à un reconstructeur d'état ou observateur qui est un modèle déterministe du système à commander.

Reconstructeur d'état

Le rôle de l'observateur est de fournir une estimation de l'état réel du système à l'instant $i + 1$, à partir des informations présentes à l'instant i . Une représentation du système et de son observateur est donné par le diagramme structurel suivant :

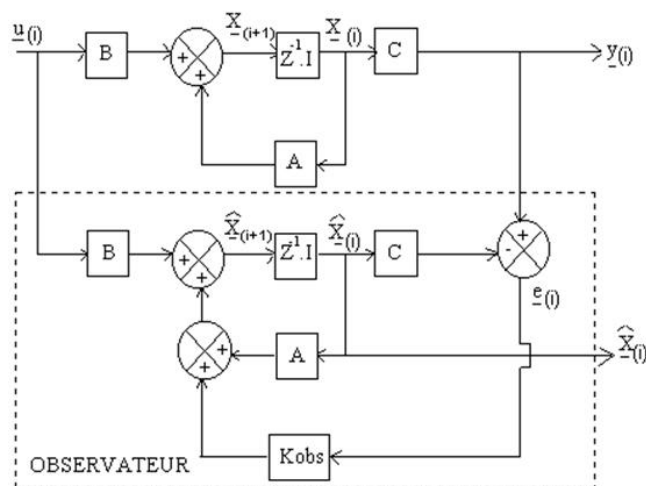
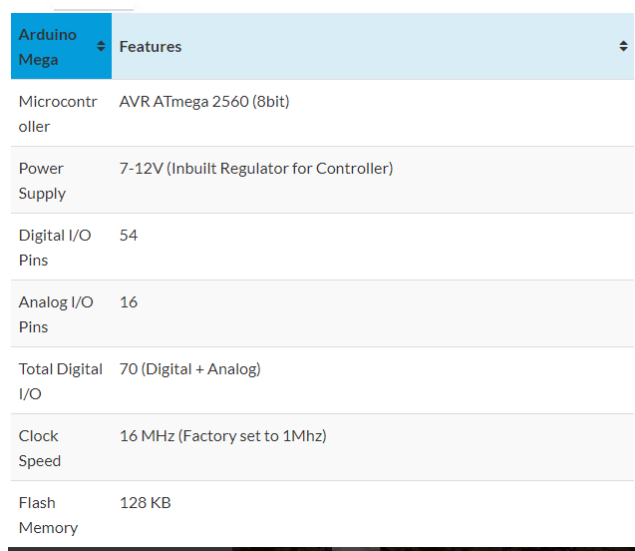


FIGURE 58 – Modèle Simulink d'un Observateur de sortie

Annexe C

Caractéristique de la carte arduino MEGA 2560



Arduino Mega Features	
Microcontroller	AVR ATmega 2560 (8bit)
Power Supply	7-12V (Inbuilt Regulator for Controller)
Digital I/O Pins	54
Analog I/O Pins	16
Total Digital I/O	70 (Digital + Analog)
Clock Speed	16 MHz (Factory set to 1Mhz)
Flash Memory	128 KB

FIGURE 59 – Caractéristique de l’arduino méga 2560

Application du capteur accéléromètres et gyroscopes

Les accéléromètres et les gyroscopes sont des capteurs de choix pour l’acquisition d’informations d’accélération et de rotation dans les drones, les téléphones portables, les automobiles, les avions et les dispositifs IoT portables. Cependant, les accéléromètres et les gyroscopes sont sujets à erreurs, notamment le bruit et la dérive (respectivement). Les concepteurs doivent donc utiliser de nouvelles approches pour obtenir une précision optimale.

Parmi ces approches figure la fusion de capteurs. Cet article évalue les accéléromètres et les gyroscopes indépendamment pour voir comment ces erreurs de bruit et de dérive se produisent. Il présente ensuite des exemples pour chaque type de capteur et explique l’utilisation des techniques de fusion de capteurs pour combiner les résultats de ces deux capteurs et réduire l’impact de ces erreurs. Choisir les bons capteurs

Un accéléromètre mesure toutes les forces linéaires appliquées sur un objet en millivolts/g (mV/g). Un objet mobile peut présenter un mouvement dynamique tel que l’accélération, ainsi que la gravité comme force statique continue. En fixant un accéléromètre à un objet, il est

possible de mesurer son accélération et la force gravitationnelle exercée sur l'objet. Toutefois, les accéléromètres ont tendance à montrer des erreurs de position au fil du temps[23].



FIGURE 60 – Un drone avec deux capteurs, accéléromètre tridimensionnel et gyroscope 3D

Le gyroscope fournit le taux de variation de la vitesse angulaire exercée sur un objet au fil du temps, en mV par degré par seconde (mV/deg/s). En fixant un gyroscope à un objet, le capteur peut facilement mesurer les changements angulaires de l'objet, mais tout comme les accéléromètres, les gyroscopes montrent des erreurs angulaires de plus en plus importantes au fil du temps[23].

De nombreux accéléromètres et gyroscopes sont fabriqués à l'aide de microsystèmes électromécaniques (MEMS). Le processus de production des capteurs MEMS combine des fonctions mécaniques et silicium dans le même substrat silicium du micromètre. Les principaux composants de ces dispositifs sont les éléments mécaniques, le mécanisme de détection et le circuit intégré spécifique à une application (ASIC)[23].

Accélérateurs de type MEMS

La construction d'un accéléromètre MEMS unique utilise des plaques silicium fixes et des ressorts mécaniques qui répondent aux forces externes

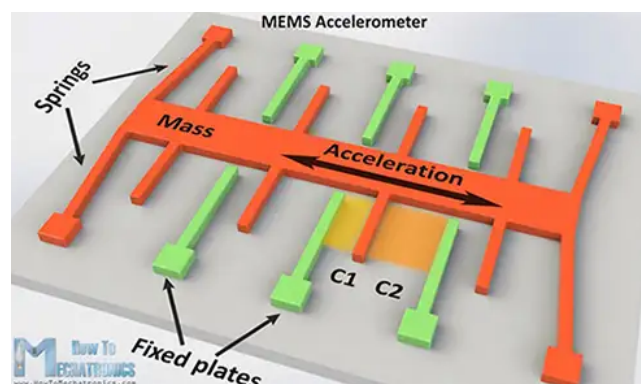


FIGURE 61 – Le modèle d'accéléromètre MEMS utilisé pour les systèmes mécaniques

Application d'Arduino avec Simulink, StateFlow

Mise en œuvre de l'accéléromètre gyroscope numérique MPU 6050

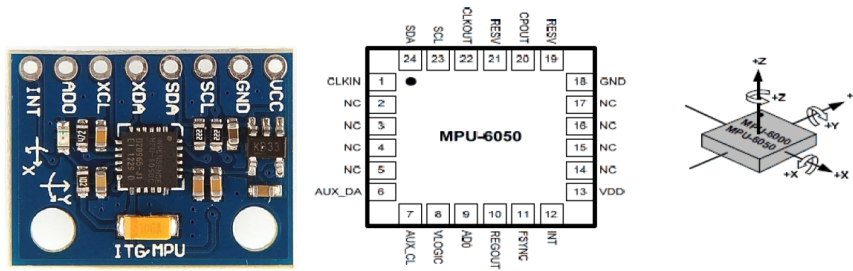


FIGURE 62 – mpu6050 réel

Schéma structurel de la carte du capteur

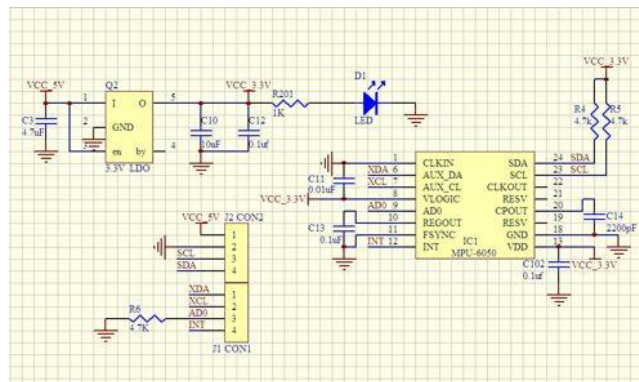


FIGURE 63 – Schéma structurel de la carte du capteur

Modèle de simulation *MPU6050.slx*

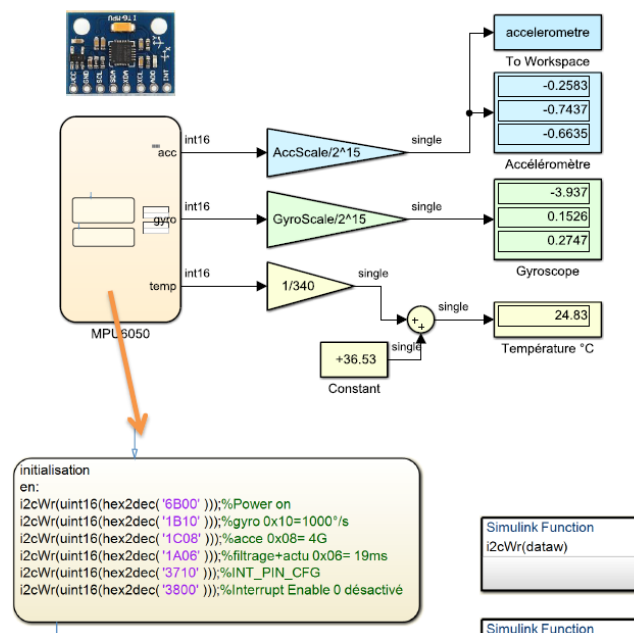


FIGURE 64 – Configuration d'un I2C

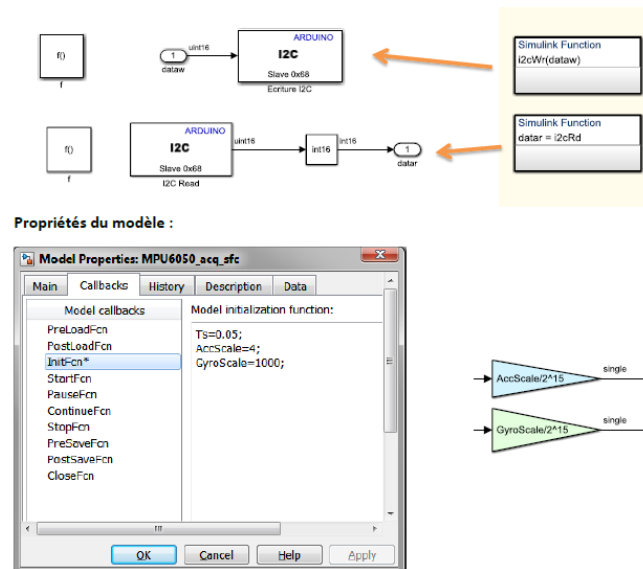


FIGURE 65 – Paramétrage du gyroscope

RASPLib installation Instructions 2015a

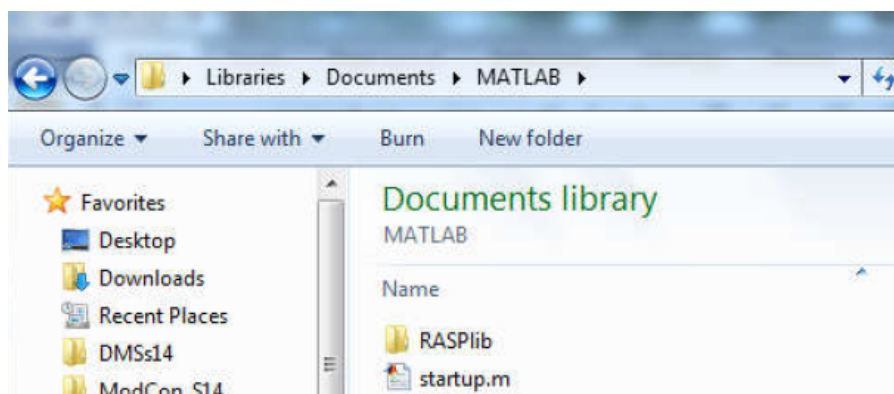


FIGURE 66 – Configuration du RASPLIB

o Ouvrez Matlab, ouvrez la bibliothèque Simulink, puis cliquez à droite sur "Rensselaer Arduino Support Package" et sélectionnez "Open Rensselaer Arduino Support Package".

o Si vous ne voyez pas "Rensselaer Arduino Support Package" voir les étapes à suivre dans la figure 56.

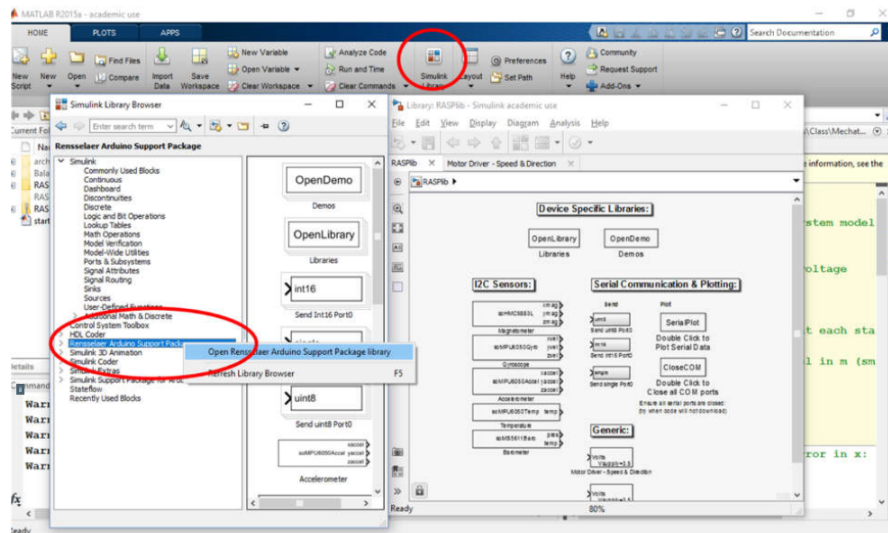


FIGURE 67 – Configuration du hardware

Simulink peut vous demander de mettre à jour les informations de la bibliothèque. Dans le volet de gauche, à gauche (Figure 67) cliquez avec le bouton droit de la souris et sélectionnez "Rafraîchir le navigateur de la bibliothèque". (Selon votre version, la fonction les blocs réels dans la bibliothèque peuvent sembler différents).

Modèle simulink



FIGURE 68 – Vue Globale de la Réalisation

Bibliographie

- [1] E. Ragnar and M. Per. Lqg control design for balancing an inverted pendulum mobile robot. *Intelligent Control and Automation*, 2011.
- [2] A. Guillaume. *Application de l'approche par fonctions transverses à la commande de véhicules non-holonomes manœuvrant*. Phd, L'ECOLE NATIONALE SUPERIEURE DES MINES DEPARIS - SOPHIA ANTIPOLIS, 2005.
- [3] H. BENARIBA. *Commande d'un robot mobile (Mobrob) sur deux roues*. Magister, Université de Tlemcen, 2013.
- [4] R. Samer. *Contribution à l'estimation et à la commande de systèmes mécaniques sous-actionnés*. Phd, Ecole Centrale de Lille, 2008.
- [5] A. KOUIDER and P. Abdelkrim. *Etude et réalisation expérimentales d'une commande destinée à contrôler un pendule inverse*. Master, UDBKM, 2017.
- [6] L. AKSAS. *Commande floue d'un pendule inversé-application sur un banc d'essais*. Magister, Université Abderrahmane MIRA BEJAIA, 2014.
- [7] D. SADAOUI. *Contribution à l'étude et au contrôle d'attitude des satellites*. Phd, Université de Constantine, 2012.
- [8] A. Khemissat. *Plateforme de prototypage rapide pour la robotique mobile : Application à un robot Auto-balancé*. Master, UMBB, Boumerdes, 2016.
- [9] F. Mudry. *Modélisation et régulation d'un pendule inversé*. Phd, d'Automatisation industrielle de l'Ecole d'Ingénieurs du Canton de Vaud, Suisse, 2003.
- [10] Ferhat Lahouazi. *Mise on œuvre d'une stratégie de commande nuer floue Application*. Magister, Mouloud mammeri Tizi-ouzou, 2011.
- [11] A. Mikael and K. Jonas. *Design, construction and verification of a self-balancing vehicle*. Phd, Department of Signals and Systems Chalmers University of Technology Goteborg, Sweden, 2012.
- [12] Mathwork. "robot auto-balancé : Segway". <http://public.iutenligne.net/etudes-etrealisations/nardi/Segway/Principe/index.html>, 2016.
- [13] N. CHELLY and A. CHARED. *Commande d'un système thermique à l'aide de la carte Arduino UNO*. 2014.
- [14] L. JIN and H. KAO. Gait balance and acceleration of a biped robot based on q-learning. *IEEE Access*, 2016.
- [15] G. Felix, D. Aldo, and C. Silvio. Joe : A mobile, inverted pendulum. *IEEE TRANSACTIONS ON INDUSTRIAL ELECTRONICS*,, 2002.
- [16] R. Ooi. *Balancing a two-wheeled autonomous robot*. Phd, University of Western Australia, 2003.

- [17] B. Merabet. *NOTES DE COURS : TECHNIQUES DE COMMANDE AVANCÉE*. PolycoPie, Centre universitaire de Rélizane Ahmed Zabana Faculté Des sciences et de la Technologie ST Département d'Electrotechnique, 2015.
- [18] P. Nicolas and R. Pierre. *Automatique Dynamique et contrôle des systèmes*. Phd, MINES ParisTech, 2011.
- [19] Edouard Laroche. *Commande Optimale*. Phd, École Nationale Supérieure de Physique de Strasbourg 3A - Option ISAV Université Louis Pasteur de Strasbourg, 2007.
- [20] A. SUHARDI. *TWO-WHEELED BALANCING ROBOT CONTROLLER DESIGNED USING PID*. Phd, Faculty of Electrical and Electronic Engineering Universiti Tun Hussein Onn Malaysia, 2015.
- [21] M. ELTAIEF and A. CHOUCHENE. *ATELIER CONCEPTION 1(CAO)*. 2015.
- [22] Mathwork. “apply sensor fusion to accelerometers and gyroscopes”. <https://www.digikey.fr/fr/articles/techzone/2018/jan/apply-sensor-fusion-to-accelerometers-and-gyroscopes>, 2017.
- [23] Mathworks. “matlab”. <https://whatis.techtarget.com/definition/MATLAB>, 2018.
- [24] F. Alberto, C. Eugene, and L. Beom. Usability and validation of the smarter balance system : An unsupervised dynamic balance exercises system for individuals with parkinson’s disease. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 2018.