



الجمهورية الجزائرية الديمقراطية الشعبية
République Algérienne Démocratique et Populaire
وزارة التعليم العالي والبحث العلمي
Ministère de L'Enseignement Supérieure et de la Recherche Scientifique

جامعة وهران 2 محمد بن أحمد
Université d'Oran 2 Mohamed Ben Ahmed
معهد الصيانة و الأمن الصناعي
Institut de Maintenance et de Sécurité Industrielle

Département de Maintenance en Instrumentation

MÉMOIRE

Pour l'obtention du diplôme de Master

Filière : Génie Industriel

Spécialité : Maintenance en automatisme et instrumentation

Thème

Réalisation d'un Robot Auto-Équilibrant

Présenté et soutenu publiquement par :

HAFNAOUI MOHAMED ADNANE

BOUSSOURI AICHA

Devant le jury composé de :

Nom et Prénom	Grade	Etablissement	Qualité
BENFEKIR Abderrahim	MCB	IMSI-UNIV. D'ORAN 2	Président
MEKKI IBRAHIM EL-Khalil	MBK	IMSI-UNIV. D'ORAN 2	Encadreur
ADDA NEGGAZ Samir	MAA	IMSI-UNIV. D'ORAN 2	Examineur

Année 2024/2025

Remerciements

On remercie Allah le tout puissant de nous avoir donné la santé et la volonté d'entamer et de terminer ce mémoire.

Tout d'abord, nous remercions vivement Monsieur MEKKI IBRAHIM EL-Khalil, notre directeur de mémoire, pour ses conseils avisés, sa disponibilité, et son soutien constant tout au long de ce travail. Ses remarques pertinentes et son exigence scientifique ont grandement contribué à la qualité de ce mémoire.

Nous souhaitons également remercier l'ensemble des enseignants de l'IMSI-Oran2 pour la formation de qualité qu'ils nous ont dispensée durant ces années d'études. Leurs enseignements ont été une source inestimable de savoir et d'inspiration.

Dédicaces

Je dédie ce modeste travail accompagné d'un profond amour :

À mon cher père, qui m'a toujours soutenu sans faille. Ton soutien constant, tes encouragements et ta compagnie tout au long de ce parcours ont été une source persévérante d'inspiration et de motivation.

Ma chère mère, ton amour inconditionnel et tes prières m'ont toujours soutenus, et je t'offre ces mots du plus profond de mon cœur.

Mon frère Mouad et ma sœur Manel, qui m'ont accompagné dans les bons comme dans les mauvais moments, vous remercient pour votre amour et votre compagnie, qui ont facilité ce parcours.

À mes amis Naïm et Imad, deux compagnons inoubliables, merci pour les moments et les souvenirs partagés durant nos années d'université.

Ce mémoire est le fruit de vos encouragements et de votre soutien. Je vous le dédie à tous avec amour et gratitude.

<< Hafnaoui Mohamed Adnane >>

Dédicaces

Je dédie ce modeste travail accompagné d'un profond amour :

À celle qui m'a arrosé de tendresse et d'espoirs, à la source d'amour Incessible, à la mère des sentiments fragiles qui ma bénie par ses prières.

À mon support dans ma vie, qui m'a appris à me diriger Vers la gloire « mon père »

À mes chères sœurs et mon frère « ALAA, IBTIHAL ET ZAIID »

À ma chère tante, ma supportrice. À toutes les personnes de ma grande famille

À ma meilleure amie « OMICH » et tous mes amis

Je tiens à exprimer ma profonde gratitude envers toutes les personnes qui ont contribué, de près ou de loin, à l'aboutissement de ce mémoire de fin d'études.

<< Boussouri Aicha >>

Résumé:

Le projet vise à développer un robot auto-équilibré intégrant des fonctionnalités avancées telles que la détection d'obstacles, la prédiction de chute, et la communication sans fil. Utilisant un ESP32 comme unité de commande, un MPU6050 pour la mesure de l'inclinaison et un moteur à courant continu avec encodeur pour un contrôle précis, le robot utilise un algorithme PID pour le maintien de l'équilibre en temps réel et un filtre de Kalman pour une précision de mesure accrue. Une intelligence artificielle intégrée permet une détection et une évitement intelligents des obstacles en temps réel. La mise en œuvre réussie de ces technologies, combinée à la localisation par GPS, démontre le potentiel de création de robots mobiles intelligents et autonomes, capables de naviguer et d'interagir efficacement avec leur environnement.

ملخص:

يهدف المشروع إلى تطوير روبوت ذاتي التوازن مزود بميزات متقدمة، مثل اكتشاف العوائق، التنبؤ بالسقوط والاتصال اللاسلكي. يستخدم الروبوت وحدة التحكم ESP32، ومستشعر MPU6050 لقياس الميل، بالإضافة إلى محركات تيار مستمر مزودة بمشفرات (encoders) للتحكم الدقيق في الحركة. يعتمد النظام على خوارزمية PID للحفاظ على التوازن في الوقت الفعلي، وعلى مرشح Kalman لتحسين دقة القياس، مع دمج الذكاء الاصطناعي لتفادي العوائق في الوقت الفعلي. يثبت التنفيذ الناجح لهذه التقنيات، مع الاستفادة من تحديد المواقع عبر GPS، إمكانية تطوير روبوتات ذكية مستقلة قادرة على التنقل والتفاعل بفعالية مع بيئتها.

Abstract:

The aim of this project is to develop a self-balancing robot with advanced features such as obstacle detection, fall prediction, and wireless communication. The robot uses an ESP32 as a control unit, an MPU6050 for tilt measurement, and DC motors with encoders for precise control. It employs a PID algorithm for real-time balancing and a Kalman filter for improved measurement accuracy. Integrated artificial intelligence enables real-time intelligent obstacle detection and avoidance. The successful implementation of these technologies, combined with GPS localization, demonstrates the potential for developing intelligent autonomous mobile robots capable of navigating and interacting effectively with their environment.

Sommaire

Remerciements.....	I
Dédicaces.....	II
Résumé.....	III
Sommaire.....	IV
Listes des figures.....	VIII
Listes des tableaux.....	X
Liste des abréviations.....	XI
Introduction générale	1
Chapitre I : Revue de Littérature	2
1.1.1 Introduction.....	3
1.1.2 Modélisation mathématique du pendule inversé.....	3
1.1.2.1 Cas 1 : Pendule inversé simple (sans chariot).....	3
1.1.2.2 Cas 2 : Pendule inversé monté sur un chariot.....	4
1.1.3 Application au robot auto-équilibré.....	5
1.2. Applications des robots auto-équilibrés.....	5
1.2.1 Transport personnel (Segway).....	6
1.2.1.1 Concept et historique.....	6
1.2.1.2 Applications actuelles.....	6
1.2.2 Gita cargo robot.....	7
1.2.2.1 Avancées et défis.....	7
1.2.3 Automatisation industrielle (Handle).....	8
1.2.3.1 Contexte et rôle.....	8
1.2.3.2 Cas d'utilisation.....	8
1.2.3.3 Perspectives et défis.....	8
1.2.4 surveillance, éducation, assistance médicale et assistance agricole.....	9
1.2.4.1 Applications diversifiés.....	9
1.2.4.2 Surveillance.....	9
1.2.4.3 Éducation.....	9
1.2.4.4 Réhabilitation médicale.....	9
1.2.4.5 Assistance Agricole.....	9
1.2.4.6 Défis et perspectives.....	10
Chapitre II : Modélisation mathématique	11
2.1 Principe de fonctionnement.....	12
2.2 Modèle linéaire d'un moteur à courant continu.....	12

2.3	Modèle dynamique pour un pendule inversé à deux roues.....	18
2.4	Modélisation dynamique du châssis du robot (pendule inversé).....	20
2.5	Équations de mouvement non linéaires.....	21
2.6	Linéarisation du modèle.....	21
2.7	Représentation en espace d'état.....	22
2.8	Analyse et déduction de la fonction de transfert.....	22
2.9	Calcul de la matrice d'état.....	25
2.10	Matrices en espace d'état du système d'équilibrage à deux roues.....	26
2.11	Modèle en espace d'état simplifié.....	27
2.12	Commandabilité.....	28
2.13	Résumé.....	28
	Chapitre III : Conception et construction du robot.....	29
3.1	Architecture logicielle.....	30
3.1.1	Présentation de l'architecture logicielle.....	30
3.1.2	Définition de l'IDE Arduino.....	30
3.1.3	Langage utilisé.....	30
3.1.4	Comparaison C++ (Arduino IDE) vs Python (MicroPython) sur ESP32	30
3.2	Les principaux composants du logiciel.....	31
3.2.1	Obtenir des données à partir des capteurs.....	31
3.2.2	Mise en œuvre de l'algorithme de contrôle.....	32
3.2.3	Communication Bluetooth avec l'application.....	32
3.2.4	Système d'alarme par buzzer.....	33
3.3	Partie électrique.....	34
3.3.1	ESP32 microcontrôleur.....	34
3.3.2	TB1266FNG Pont en H.....	35
3.3.3	Gyroscope MPU6050 (et Accéléromètre Intégré).....	36
3.3.4	Capteur ultrasonique.....	36
3.3.5	Moteurs avec encoders et roues.....	37
3.3.6	Module GPS GY-NEO6MV2.....	38
3.3.7	Batteries 18650.....	39
3.3.8	Importance des condensateurs dans le projet.....	40
3.4	Conception du PCB.....	41
3.4.1	Étapes pour créer un PCB à l'aide d'EasyEDA.....	41
3.5.1	Architecture de l'Application.....	49
3.6	Utilisation d'un réseau de neurones pour la navigation du robot.....	49
3.6.1	Historique des réseaux de neurones.....	49
3.6.2	Définition d'un réseau de neurones.....	51

3.6.3 Étapes de développement du modèle d'intelligence artificielle.....	51
3.7 Partie mécanique.....	54
3.7.1 Conception.....	54
3.7.2 Modèle 3D du robot à auto-équilibrage.....	55
3.7.2.1 Châssis.....	55
3.7.2.2 Roues.....	56
3.7.2.3 Supports.....	56
3.7.2.4 Robot complet.....	57
3.8 Assemblage.....	57
3.9 Conclusion.....	58
Chapitre IV : Théorie de Contrôle.....	60
4.1 Modélisation, analyse et contrôle d'un robot auto-équilibré.....	61
4.2 Système en boucle fermée.....	61
4.3 Aperçu du contrôleur PID.....	62
4.4 Algorithme du contrôleur PID.....	62
4.5 Influence des actions proportionnelle, intégrale et dérivée.....	63
4.5.1 Contrôleur proportionnel (P).....	63
4.5.2 Contrôleur proportionnel intégral (PI).....	64
4.5.3 Contrôleur proportionnel-intégral-dérivé (PID).....	64
4.6 Pourquoi un contrôleur PID est nécessaire.....	66
4.6.1 Démonstration de l'instabilité du système.....	66
4.7 Pourquoi utiliser un filtre de Kalman.....	66
4.8 Modélisation Symbolique et Analyse de Stabilité.....	67
4.8.1 Analyse Numérique de l'Instabilité en Boucle Ouverte.....	67
4.8.2 Analyse des réponses dynamiques multi-sorties - étude des réponses à des perturbations et entrées.....	68
4.8.3 Estimation de l'angle avec un filtre de Kalman.....	68
4.8.4 Stabilisation avec un Contrôleur PID.....	69
4.9 Analyse des forces du robot auto-équilibré.....	70
4.9.1 Équation d'équilibre des couples.....	70
4.9.2 Forces agissant sur le robot.....	71
4.9.3 Rôle du moteur à courant continu avec réducteur.....	72
4.10 Contrôle par régulateur PD.....	72
4.11 Implémentation du contrôle moteur.....	73
4.12 Importance des engrenages.....	73
4.13 Problèmes fréquents et solutions.....	74
4.14 Résumé.....	74

4.15 Justification du choix ($K_i = 0$) dans la commande.....	74
4.15.1 Réactivité et stabilité immédiate.....	74
4.15.2 Inutilité de la correction de l'erreur statique.....	74
4.15.3 Risque de "windup" de l'intégrateur.....	75
4.15.4 Sensibilité accrue au bruit des capteurs.....	75
4.15.5 Propriétés du modèle dynamique.....	75
4.15.6 Bilan technique du choix ($K_i = 0$).....	75
4.16 Analyse Critique des Modèles Mathématiques pour Robots Autostabilisants : Limites et Perspectives d'Amélioration.....	76
4.16.1 Introduction.....	76
4.16.2 Principales limitations des modèles théoriques.....	76
4.16.3 Validation expérimentale et ajustements nécessaires.....	77
4.17 Justification de l'intégration des effets réalistes.....	78
4.18 Résultats graphiques.....	79
4.19 Construction de la Simulation dans Simulink.....	82
4.19.1 Utilisation d'un Contrôleur PID.....	83
4.20 Régulateur Linéaire Quadratique (LQR).....	89
4.20.1 Application à un robot auto-équilibrant.....	90
4.21 Visualisation des résultats	91
4.22 Comparaison entre PID et LQR.....	91
4.22.1 Facilité d'implémentation sur ESP32.....	92
4.23 Conclusion	93
Chapitre V : Tests et Résultats.....	93
5.1. Étalonnage.....	95
5.1.1. MPU6050.....	95
5.1.2. Test du filtre de Kalman pour la sélection des axes.....	96
5.2. Optimisation et Tests.....	96
5.2.1. Optimisation.....	97
5.2.2. Tests.....	97
5.3. Conclusion.....	98
CONCLUSION GENERALE	100
Références	101
Annexe	106
Figure a.1 : Broche de puce ESP32.....	106
Figure a.2 : Broche de puce NEO-6.....	106
Figure a.3 : Schéma du circuit intégré Régulateur LM2596.....	107
Figure a.4 : Schéma du circuit intégré TB6612FNG.....	107

Figure a.5 : Schéma du circuit intégré MPU6050.....	108
Figure a.6 : Structure de HC-SR04.....	108

Liste des figures

Figure I.1 : une illustration du système pendule-chariot avec (θ , m , M , L , F , x)	4
Figure I.2 : Le Segway Personal Transporter (PT).....	6
Figure I.3 : Robot Gita, sphérique à deux roues	7
Figure I.4 : travail d'un robot auto-équilibré Handle dans un entrepôt automatisé.....	8
Figure I.5 : robot auto-équilibré à deux roues dans l'assistance agricole.....	10
Figure II.1 : Schéma équivalent d'un moteur à courant continu.....	13
Figure II.2 : Diagramme du corps libre du châssis	20
Figure III.1 : Architecture générale du robot.....	32
Figure III.2 : Interface de communication Bluetooth +Wifi entre le robot et l'application externe.....	33
Figure III.3 : Schéma de l'ESP32 et ses principales interfaces.....	35
Figure III.4 : Configuration du pont en H TB1266FNG pour le contrôle des moteurs... ..	35
Figure III.5 : Module gyroscope et accéléromètre MPU6050 avec ses axes de mesure..	36
Figure III.6 : Capteur ultrasonique HC-SR04 et son principe de mesure de distance.....	37
Figure III.7 : Moteur à courant continu avec encodeur incrémental et ses caractéristiques.....	37
Figure III.8 : Module GPS GY-NEO6MV2.....	39
Figure III.9 : Batteries 1865.....	39
Figure III. 10 : Interface de conception de PCB avec EasyEDA.....	40
Figure III.11 : Exemple de saisie de schéma électronique dans EasyEDA	41
Figure III.12 : Exemple de câblage de schéma électronique dans CirKit.....	42
Figure III.13 : Disposition des composants sur le PCB avant routage	42
Figure III.14 : Exportation des fichiers Gerber pour fabrication du PCB.....	43
Figure III.15 : PCBs Fabriqués (PCB GPS / PCB du contrôle).....	44
Figure III.16 : Capture d'écran de localisation du robot sur l'application	47
Figure III.17 : Interface de conception de l'application avec Android Studio	48
Figure III.18 : L'interface de l'application (Control Panel).....	49
Figure III.19 : Comparaison entre les neurones humains et les réseaux de neurones	50
Figure III.20 : Architecture d'un réseau de neurones profond avec couche d'entrée, couches cachées multiples et couche de sortie	51
Figure III.21 : Histogramme de la distribution des distances par action	52
Figure III.22 : Matrice de confusion finale illustrant le précision	53
Figure III.23 : Résultats de la précision et la perte.....	54
Figure III.24 : Évolution de la précision d'entraînement et de validation au fil des époques.....	54
Figure III.25 : Évolution de la perte d'entraînement et de validation au fil des époques.....	54

Figure III.26 :SolidWorks.....	55
Figure III.27 : Modèle 3D du châssis.....	56
Figure III.28 : Modèle 3D des roues.....	56
Figure III.29 : Modèle 3D des supports.....	57
Figure III.30 : Modèle 3D complet du robot à auto-équilibrage.....	57
Figure III.31 : Tous les composants et pièces avant assemblage.....	58
Figure III.32 : Robot Assemblé.....	58
Figure IV.1 : Schéma d'un système en boucle fermée avec capteurs, contrôleur et actionneur.....	61
Figure IV.2 : Réponse d'un contrôleur proportionnel (P) à une entrée de consigne.....	63
Figure IV.3 : Réponse d'un contrôleur proportionnel-intégral (PI) à une entrée de consigne.....	64
Figure IV.4 : Réponse d'un contrôleur proportionnel-intégral-dérivé (PID) à une entrée de consigne.....	65
Figure IV.5 : Réponse à un échelon (Position + Angle)	67
Figure IV.6 : Réponse Impulsionnelle du Position et Angle du système.....	68
Figure IV.7 :Estimation de l'angle et l'accéléromètre avec le filtre de Kalman.....	69
Figure IV.8 :Réponse à l'échelon d'angle en Boucle Fermé.....	69
Figure IV.9 : Schéma d'un pendule simple avec force appliquée.....	70
Figure IV.10 : Schéma d'un pendule avec forces composantes et accélération.....	71
Figure IV.11 : Angle d'inclinaison du robot en fonction du temps.....	79
Figure IV.12 : Position du robot en fonction du temps.....	79
Figure IV.13 : Couple de contrôle appliqué au robot.....	80
Figure IV.14 : Portrait de phase du robot.....	80
Figure IV.15 : Perturbation externe appliquée au robot.....	81
Figure IV.16: Diagramme de Bode du système en boucle fermée.....	81
Figure IV.17: Réponse à un échelon du système en boucle fermée.....	82
Figure IV.18 : Architecture générale du système dans Simulink.....	84
Figure IV.19 : Architecture générale du système dans simulink avec les sous-systèmes	84
Figure IV.20 : le modèle 3D du robot dans la fenêtre du Simulation.....	85
FigureIV.21: Blocs Solver Configuration.....	85
FigureIV.22 :WorldFrame.....	86
FigureIV.23 : Mechanism Configuration.....	86
FigureIV.24 : Revolute Joint.....	86
Figure IV.25 : Prismatic Joint.....	87
Figure IV.26 : Fenêtre du réglage manuel du PID.....	88
Figure IV.27 : Slider du PID Tuner.....	89
Figure IV.28 : Fenêtre de la réponse de système au PID Tuner.....	89
Figure IV.29 : Fenêtre de la réponse de l'angle au LQR.....	92
Figure IV.30 : Fenêtre de la réponse de la position au LQR.....	92
Figure V.1 : Test du filtre de Kalman pour la sélection des axes.....	96
Figure V.2 : Interface d'application l'ajustement des paramètres PID.....	97
FigureV.3 : Robot auto-équilibré sur moquette.....	98

Liste des tableaux

Tableau III.1: Comparaison C++ (Arduino IDE) vs Python (MicroPython) sur ESP32	31
Tableau III.2 : Caractéristiques techniques du moteur à courant continu.....	38
Tableau IV.1: Effets des coefficients du régulateur PID sur la réponse du système.....	65
Tableau IV.2 : Bilan technique du choix $K_i=0$	76
Tableau IV.3 : Comparaison entre PID et LQR.....	93

Abréviations

NN: Neural Network

PCB: Printed Circuit Board

PWM: Pulse Width Modulation

COM: Center of Mass

GPS : Global Positioning System

AI: Artificial Intelligence (Intelligence Artificielle)

IoT: Internet of Things (Internet des Objets)

RNA: Recurrent Neural Network (Réseau Neuronal Récurrent)

DMP: Digital Motion Processor (Processeur de Mouvement Numérique)

IMU: Inertial Measurement Unit (Unité de Mesure Inertielle)

DRC: Design Rule Check (Vérification des Règles de Conception)

ERC: Electrical Rule Check (Vérification des Règles Électriques)

VC: Voltage Common (Tension Commune)

VM: Voltage Motor (Tension Moteur)

VCC: Voltage Common Collector (Tension Collecteur Commun)

EEPROM: Electrically Erasable Programmable Read-Only Memory (Mémoire Morte Programmable et Effaçable Électriquement)

RAM: Random Access Memory (Mémoire à Accès Aléatoire)

SPI: Serial Peripheral Interface (Interface Périphérique Série)

UART: Universal Asynchronous Receiver-Transmitter (Récepteur-Émetteur Asynchrone Universel)

WIFI: Wireless Fidelity (Fidélité Sans Fil)

BLE: Bluetooth Low Energy (Bluetooth à Basse Consommation)

UUID: Universally Unique Identifier (Identifiant Universellement Unique)

NMEA: National Marine Electronics Association (Association Nationale de l'Électronique Marine)

API: Application Programming Interface (Interface de Programmation d'Application)

CAO: Computer-Aided Design (Conception Assistée par Ordinateur)

3D: 3-Dimensions (Trois Dimensions)

2D: 2-Dimensional (Deux Dimensions)

STEM: Science, Technology, Engineering, and Mathematics (Science, Technologie, Ingénierie et Mathématiques)

MIMO: Multiple Input Multiple Output (Entrées Multiples Sorties Multiples)

CC (DC): Continuous Current (Direct Current) (Courant Continu)

AMPR: Autonomous Mobile Robot (Robot Mobile Autonome)

MVC: Model View Controller (Modèle Vue Contrôleur)

Introduction Générale

Les robots mobiles, de plus en plus présents dans des domaines tels que l'industrie, la médecine, la maison et l'éducation, jouent un rôle essentiel dans l'automatisation de tâches complexes, de manière autonome ou semi-autonome, réduisant ainsi la charge de travail humaine et augmentant l'efficacité. Parmi eux, les robots auto-équilibrés, combinant mécanique, électronique et intelligence artificielle, représentent un défi technique majeur en raison de la nécessité d'un contrôle en temps réel, du traitement de données multi capteurs et de la gestion de la stabilité dynamique. Ces systèmes sont utilisés dans les véhicules électriques, assistants personnels mobiles et transport autonome. L'objectif principal est de concevoir un tel robot utilisant un ESP32, un capteur MPU6050 pour mesurer l'angle d'inclinaison et un moteur à courant continu avec encodeur pour un contrôle précis, tout en intégrant des capteurs à ultrasons pour la détection d'obstacles, des buzzers pour les alarmes sonores et la communication Bluetooth pour le contrôle et la surveillance en temps réel, avec la participation de l'IA pour améliorer l'endurance, la prédiction des obstacles et la prise de décision instantanée.

Chapitre I
Revue de Littérature

Revue de Littérature

1.1 Le problème du pendule inversé

1.1.1 Introduction :

En tant que système mécanique classique, le pendule simple est le pilier de base de la recherche sur la dynamique des systèmes et le contrôle. Sa simplicité géométrique, alliée à la richesse de son comportement dynamique, en fait un modèle de référence pour l'analyse de phénomènes complexes dans des domaines allant de la physique à l'ingénierie. En particulier, le pendule inversé, une variante dans laquelle la masse est située au-dessus du point de pivot, a attiré beaucoup d'attention en raison de son instabilité inhérente et de ses applications pratiques. Les caractéristiques d'équilibre instable du système en font une plateforme de test idéale pour développer et évaluer des stratégies de contrôle avancées telles que le contrôle PID (proportionnel-intégral-dérivé) ou le contrôle optimal LQR (régulateur quadratique linéaire).

1.1.2 Modélisation mathématique du pendule inversé :

Un pendule inversé est généralement modélisé comme un objet de masse m , attaché à une tige rigide sans masse de longueur L , tournant autour d'un point fixe ; ou, dans une configuration plus courante, attaché à un chariot mobile de masse M . L'angle θ mesuré par rapport à une ligne verticale ascendante ($\theta = 0$ en équilibre instable) décrit la direction du pendule.

1.1.2.1 Cas 1 : Pendule inversé simple (sans chariot)

La dynamique de ce système est régie par les équations du mouvement issues du principe de Lagrange. L'énergie cinétique et l'énergie potentielle sont respectivement données par :

$$T = \frac{1}{2}m(L\dot{\theta})^2 = \frac{1}{2}mL^2\dot{\theta}^2$$

$$V = -mgL\cos(\theta)$$

La fonction de Lagrange est donc :

$$L = T - V = \frac{1}{2}mL^2\dot{\theta}^2 + mgL\cos(\theta)$$

En appliquant l'équation d'Euler-Lagrange :

$$\frac{d}{dt} \left(\frac{\partial L}{\partial \dot{\theta}} \right) - \frac{\partial L}{\partial \theta} = 0$$

on obtient l'équation du mouvement :

$$mL^2 \ddot{\theta} - mgL \sin(\theta) = 0 \Rightarrow \ddot{\theta} - \frac{g}{L} \sin(\theta) = 0$$

Pour de petits angles $\theta \approx 0$, on peut linéariser le système avec l'approximation $\sin(\theta) \approx \theta$, ce qui donne :

$$\ddot{\theta} - \frac{g}{L} \theta = 0$$

Il s'agit d'une équation différentielle linéaire d'un oscillateur harmonique instable.

1.1.2.2 Cas 2 : Pendule inversé monté sur un chariot

Dans ce cas, la masse m est toujours à l'extrémité d'une tige de longueur L , mais la tige est attachée à un chariot mobile de masse M . Une force externe F est appliquée au chariot. Ce système est plus complexe car il introduit un couplage entre la position du chariot x et l'angle θ .

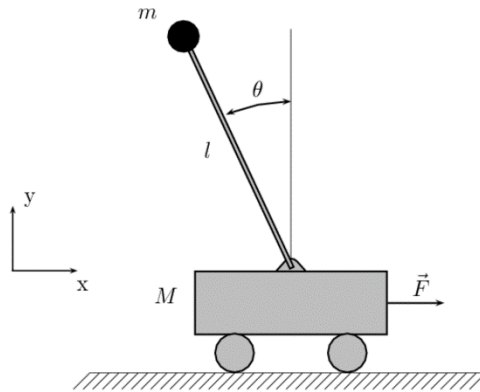


Figure I.1 : une illustration du système pendule-chariot avec (θ, m, M, L, F, x)

Les équations du mouvement sont obtenues par la méthode lagrangienne et aboutissent à un système d'équations non linéaire :

$$\begin{aligned} (M + m)\ddot{x} + mL\ddot{\theta}\cos(\theta) - mL\dot{\theta}^2\sin(\theta) &= F \\ L\ddot{\theta} + \ddot{x}\cos(\theta) - g\sin(\theta) &= 0 \end{aligned}$$

Après linéarisation autour de l'équilibre $\theta = 0$, $\dot{\theta} = 0$, $x = 0$, on obtient un système d'équations linéaires du second ordre, qui peut être exprimé en forme d'espace d'état :

$$\dot{x} = Ax + Bu$$

$$y = Cx$$

avec :

$$x = \begin{bmatrix} x \\ \theta \\ \dot{x} \\ \dot{\theta} \end{bmatrix}, u = F$$

Cette instabilité, où un léger écart de θ entraîne une divergence exponentielle, rend le pendule inversé idéal pour tester des algorithmes de stabilisation.

1.1.3 Application au robot auto-équilibré :

Le pendule inversé est un modèle de référence pour un robot auto-équilibré, un système à deux roues qui maintient l'équilibre en ajustant dynamiquement le couple appliqué aux roues. Comme un pendule inversé, le robot tourne autour d'un point d'équilibre instable et les écarts dans l'angle d'inclinaison nécessitent une correction rapide pour éviter de tomber. L'angle d'inclinaison du robot θ correspond directement à l'angle d'inclinaison du pendule, et le couple moteur joue un rôle similaire à la force F dans le modèle du pendule sur chariot. Le robot auto-équilibré est modélisé à l'aide de l'équation du pendule inversé, qui inclut la masse et le rayon des roues, ainsi que le moment d'inertie du corps. Des stratégies de contrôle telles que PID ou LQR, développées à l'origine pour les pendules, sont directement appliquées ici.

1.2 Applications des robots auto-équilibrés

Les robots auto-équilibrés utilisent des capteurs gyroscopiques et des algorithmes de contrôle avancés pour maintenir leur stabilité sur deux roues ou plateformes dynamiques. Ils offrent une excellente maniabilité et flexibilité. Leurs applications s'étendent du transport personnel aux domaines industriels, en passant par les applications sociales et médicales. Cette section explore trois principaux domaines d'application : le transport personnel (Segway), l'automatisation industrielle (Handle), et les applications de surveillance, d'éducation et de réadaptation médicale. Chaque sous-section détaille les technologies, les cas d'utilisation et les perspectives pertinents, et renvoie à des références récentes [1], [2].

1.2.1 Transport Personnel (Segway) :

1.2.1.1 Concept et historique

L'une des premières applications commerciales des robots auto-équilibrés a été le transport personnel, avec le Segway Personal Transporter (PT) lancé en 2001 par la société Segway. Ce véhicule à deux roues utilise des gyroscopes et des accéléromètres pour permettre aux utilisateurs de se déplacer en inclinant leur corps, offrant une expérience de conduite intuitive [3]. Le Segway a offert une alternative écologique et compacte aux modes de transport traditionnels, marquant un tournant dans le transport urbain [4]. Il couvre des domaines tels que le transport (Segway), l'automatisation industrielle (Handle), la surveillance, l'éducation et la réadaptation médicale. Chaque sous-section détaille les technologies, les cas d'utilisation et les perspectives concernés, et s'appuie sur des références récentes [1], [2].



Figure I.2 : Le Segway Personal Transporter (PT)

1.2.1.2 Applications actuelles

Les appareils de type Segway sont actuellement largement utilisés pour le transport urbain, les visites touristiques et la logistique légère. Par exemple, à Lyon, Leobotics a lancé le Segway Loomo, un robot de transport personnel intégrant des fonctions autonomes et une interface interactive [5]. Ces systèmes sont particulièrement adaptés aux environnements piétonniers tels que les aéroports, les centres commerciaux ou les parcs, réduisant la fatigue et optimisant les temps de trajet. Les robots de livraison de Segway Robotics, comme le

ServeBot S1, sont utilisés dans les restaurants et combinent fonctions d'auto-équilibrage et de transport autonome [6].

1.2.2 Gita cargo robot :

Un exemple d'innovation dans le domaine du transport personnel est le robot cargo Gita développé par Piaggio Fast Forward, présenté dans une vidéo promotionnelle en 2023.[7] Prononcé JEE'-tah (qui signifie « court trajet » en italien), ce robot à deux roues et auto-équilibré utilise des caméras et des capteurs pour suivre son utilisateur et peut transporter jusqu'à 18 kg (40 lb) de marchandises, comme des courses ou des effets personnels.[8] Avec une autonomie de quatre heures et une vitesse de pointe de 9,6 km/h, Gita est conçu pour la mobilité urbaine et les déplacements mains libres. Il utilise la technologie SLAM (Simultaneous Localization and Mapping) pour la navigation et l'évitement d'obstacles, et intègre des fonctionnalités telles qu'un haut-parleur Bluetooth et un système de verrouillage via une application mobile.[9]Gita, disponible depuis 2019 pour 3 250 \$, incarne l'évolution des robots auto-équilibrés vers des solutions pratiques et conviviales pour les tâches quotidiennes [10].



Figure I.3 : Robot Gita, sphérique à deux roues

1.2.2.1 Avancées et défis

Parmi les avancées récentes, on compte l'intégration de l'intelligence artificielle (IA) à la navigation autonome et de l'Internet des objets (IoT) à la gestion intelligente des flottes. Cependant, des obstacles subsistent, notamment une réglementation stricte sur l'utilisation des trottoirs et des coûts élevés, qui limitent son adoption généralisée. Par exemple, Gita ne

peut pas monter les escaliers et est sensible aux irrégularités du terrain, ce qui limite son utilisation dans certains environnements urbains [8].

1.2.3 Automatisation industrielle (Handle) :

1.2.3.1 Contexte et rôle

Dans le domaine de l'automatisation industrielle, les robots auto-équilibrés comme Handle, développés par Boston Dynamics, révolutionnent la logistique et la gestion des entrepôts. Handle associe une plateforme à deux roues à un bras robotisé capable de déplacer des charges avec précision dans des environnements dynamiques [11]. Ces robots répondent aux besoins de l'Industrie 4.0, où la flexibilité et l'adaptabilité sont essentielles pour des cycles de production rapides et personnalisés [12].

1.2.3.2 Cas d'utilisation

Les applications incluent le tri automatisé, le transport de marchandises et l'assistance aux opérateurs. Par exemple, Robotnik utilise des robots mobiles autonomes (AMR) similaires à Handle pour automatiser la logistique et la maintenance industrielle [13]. Une étude récente a montré que ces systèmes augmentent la productivité de 20 à 30 % tout en réduisant les erreurs humaines [14]. Ils peuvent également contribuer à limiter les tâches répétitives ou dangereuses, améliorant ainsi la sécurité.

1.2.3.3 Perspectives et défis

Les progrès de la vision par ordinateur et de l'apprentissage automatique ont permis à ces robots de naviguer dans des environnements complexes, mais les coûts élevés et la nécessité d'infrastructures adaptées (par exemple, un terrain plat) demeurent des obstacles [12].



Figure I.4 : travail d'un robot auto-équilibré Handle dans un entrepôt automatisé.

1.2.4 Surveillance, éducation, assistance médicale et assistance agricole :

1.2.4.1 Applications diversifiées :

Les robots auto-équilibrés excellent dans divers domaines tels que la surveillance, l'éducation et la réadaptation médicale en raison de leur maniabilité et de leur stabilité. Ces applications exploitent sa capacité à interagir avec des environnements complexes et à s'adapter aux besoins des utilisateurs.

1.2.4.2 Surveillance :

Dans le domaine de la surveillance, des robots tels que le GR100 de Running Brains Robotics et le Jack de TBC-France peuvent assurer la sécurité des sites industriels. Équipés de caméras, de lidars et de systèmes de communication, ils peuvent patrouiller de manière autonome et détecter les anomalies en temps réel, réduisant ainsi les risques pour le personnel[15],[16]. Le projet NOVA 2027 de Génération Robots illustre le développement de ces technologies dans le domaine de la surveillance des sites sensibles en France[17].

1.2.4.3 Éducation :

Dans le domaine de l'éducation, les robots auto-équilibrés à deux roues, tels que les plateformes éducatives dérivées de la technologie Segway, peuvent être utilisés comme outils pédagogiques pour les STEM. Ces robots sont équipés de capteurs gyroscopiques et d'interfaces interactives, permettant aux étudiants de réaliser des expériences de programmation et de robotique dans des environnements dynamiques tels que des salles de classe ou des laboratoires [18].

1.2.4.4 Réhabilitation médicale :

Dans le domaine médical, les robots auto-équilibrés tels que les exosquelettes ou les plateformes mobiles peuvent contribuer à la réadaptation. Par exemple, les exosquelettes peuvent aider les patients à récupérer leurs capacités motrices après des lésions neurologiques [19]. Le projet HARMONY, financé par l'UE, utilise des robots mobiles pour transporter du matériel médical dans les hôpitaux, réduisant ainsi la charge de travail du personnel [20].

1.2.4.5 Assistance agricole :

Dans le secteur agricole, des robots auto-équilibrés à deux roues, comme le Digital Farmhand développé par l'Université de Sydney, aident les petits exploitants agricoles, notamment dans les îles du Pacifique, à améliorer leur sécurité alimentaire[21]. Conçu par le Centre australien de robotique de terrain, dirigé par le professeur Salah Sukkarieh, ce

robot est une plateforme à deux roues alimentée à l'énergie solaire, d'un coût d'environ 1 000 dollars australiens. Il utilise des caméras et des capteurs pour effectuer des tâches telles que le semis, le désherbage et la surveillance des cultures, tout en collectant des données pour aider les agriculteurs à prendre des décisions éclairées et à améliorer leurs rendements[21].



Figure I.5 : robot auto-équilibré à deux roues dans l'assistance agricole (DIGITAL FARM ASSISTANT).

1.2.4.5 Défis et perspectives :

Ces applications nécessitent une intégration avancée de l'IA pour la reconnaissance environnementale et une fiabilité élevée pour garantir la sécurité, en particulier dans les domaines médical, industriel et agricole. Les coûts de développement et les exigences réglementaires (par exemple pour les dispositifs médicaux ou agricoles) restent des obstacles.

Conclusion :

Les robots auto-équilibrés changent de nombreuses industries grâce à leur maniabilité et leur stabilité. Les progrès de l'intelligence artificielle et des capteurs renforcent leur potentiel, mais des défis techniques, économiques et réglementaires doivent être surmontés avant qu'ils puissent être largement adoptés.

Chapitre II

Modélisation Mathématique

2.1 Principe de fonctionnement

Les modèles mathématiques sont des instruments puissants permettant de décrire et d'analyser des systèmes complexes dans de nombreuses disciplines, notamment la physique, l'ingénierie, l'économie et la biologie. Ils facilitent les prédictions, l'optimisation des systèmes et la vérification d'hypothèses dans un environnement pré-contrôlé. La création d'un modèle mathématique repose généralement sur la réduction du système complexe à ses composants essentiels, en omettant les détails superflus. Cette méthode permet de représenter le système par des équations mathématiques, qui peuvent ensuite être étudiées pour en déduire son comportement.

La modélisation mathématique est principalement utilisée dans l'étude des systèmes complexes de nature dynamique, ce qui est particulièrement important pour l'analyse et le contrôle de ces systèmes. Grâce aux équations mathématiques, le comportement du système peut être prédit, ses réponses anticipées et les lois de contrôle conçues pour atteindre le résultat souhaité.

Prenons l'exemple d'un robot auto-équilibré : le système peut être divisé en trois parties principales : le moteur, les roues (ou châssis) et la structure (qui agit comme un pendule). La dynamique de chaque composant peut être décrite par des équations différentielles, notamment l'équation du pendule inversé. En combinant ces dynamiques individuelles, on obtient un modèle mathématique unifié qui représente l'ensemble du système.

En tant que systèmes à entrées-sorties multiples (MIMO), les robots auto-équilibrés sont souvent représentés comme des espaces d'états. Dans cette représentation, la tension d'alimentation du moteur à courant continu est l'entrée du système, tandis que l'angle d'inclinaison (θ) et la position du centre de gravité (x) du robot sont les sorties.

2.2 Modèle linéaire d'un moteur à courant continu

Dans cette section, On va dériver le modèle en espace d'état des moteurs à courant continu alimentant le robot. Ce modèle prend en compte les dynamiques électrique et mécanique du moteur. Par la suite, on intégrera ce modèle dans le système dynamique global du robot, ce qui permettra d'établir une relation entre la tension d'entrée appliquée aux moteurs et le couple de commande requis pour maintenir l'équilibre du robot [22].

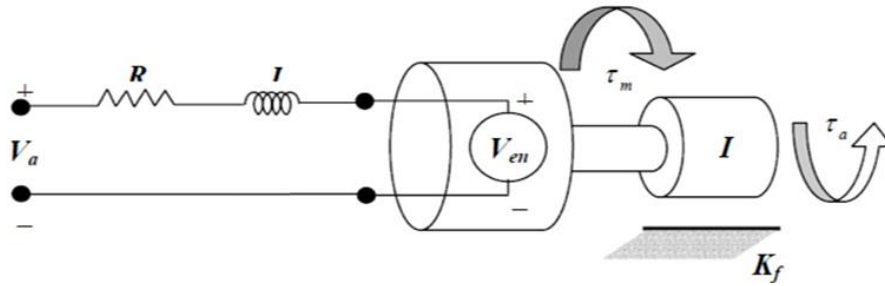


Figure II.1 : Schéma équivalent d'un moteur à courant continu.

Lorsqu'une tension est appliquée aux bornes du moteur, elle engendre un courant i dans l'armature. Ce courant produit, à son tour, un couple T_m au sein du moteur. La relation entre le courant et le couple s'exprime de la manière suivante :

$$T_m = K_t \cdot i \quad (2.1)$$

Où :

- T_m : Couple généré par le moteur,
- K_t : Constante de couple du moteur, représentant la proportionnalité entre le courant et le couple.
- i : Courant dans l'armature (A).

La constante de couple K_t est un paramètre caractéristique propre à chaque moteur. Elle peut être déterminée expérimentalement ou extraite de la fiche technique du moteur. Cette constante relie les aspects électriques et mécaniques du fonctionnement du moteur en fournissant une mesure quantitative du couple produit pour un courant donné [22].

Pour modéliser le circuit électrique du moteur, on considère un modèle simplifié composé d'une résistance et d'une inductance en série, accompagnées d'une source de tension appelée force contre-électromotrice (Fcem). Cette Fcem est générée par le mouvement des bobines à travers un champ magnétique, et la tension produite peut être approximée comme une fonction linéaire de la vitesse de rotation de l'arbre moteur.

Soient :

- V_{emf} : la tension de la force contre-électromotrice.

- ω : la vitesse angulaire de l'arbre moteur.

La relation linéaire entre ces grandeurs s'exprime ainsi :

$$V_{emf} = K_{emf} \cdot \omega \quad (2.2)$$

Où :

- V_{emf} : Tension de la force contre-électromotrice,
- K_{emf} : Constante de force électromotrice du moteur, représentant la proportionnalité entre la vitesse de l'arbre et la tension induite.

En appliquant la loi des tensions de Kirchhoff (LTV) au circuit électrique du moteur à courant continu, on obtient l'équation différentielle suivante :

$$V_a - R \cdot i - L \cdot \frac{di}{dt} - V_{emf} = 0 \quad (2.3)$$

Où :

- V_a : Tension appliquée au moteur,
- R : Résistance de l'armature,
- i : Courant circulant dans l'armature,
- L : Inductance de l'armature,
- $\frac{di}{dt}$: Dérivée du courant par rapport au temps,
- V_{emf} : Tension de la force contre-électromotrice.

Cette équation décrit l'équilibre des tensions dans le circuit électrique du moteur. Elle indique que la tension appliquée V_a , diminuée des chutes de tension à travers la résistance ($R \cdot i$), l'inductance ($L \cdot \frac{di}{dt}$) et la force contre-électromotrice (V_{emf}), est égale à zéro.

Lors de la dérivation de l'équation du mouvement du moteur électrique, la friction agissant sur l'arbre peut être approximée par une fonction linéaire de la vitesse de rotation. On

désigne par K_f le coefficient de friction sur l'arbre du moteur, ce dernier étant proportionnel à la vitesse angulaire ω [23].

Selon la deuxième loi de Newton appliquée à la rotation, la somme de tous les couples agissant sur l'arbre est égale au produit du moment d'inertie de l'armature I_r et de l'accélération angulaire $\frac{d\omega}{dt}$ [23]. Cette relation s'exprime comme suit :

$$\sum M = T_m - K_f \cdot \omega - T_a = I_r \cdot \frac{d\omega}{dt} \quad (2.4)$$

Où :

- $\sum M$: Somme des couples agissant sur l'arbre moteur,
- T_m : Couple généré par le moteur,
- K_f : Coefficient de friction visqueuse,
- ω : Vitesse angulaire de l'arbre moteur,
- T_a : Couple externe appliqué,
- I_r : Moment d'inertie de l'armature,
- $\frac{d\omega}{dt}$: Accélération angulaire de l'arbre.

Cette équation relie les différentes forces en jeu : le couple moteur T_m , le couple de friction $K_f \cdot \omega$, et le couple externe T_a , à l'inertie du rotor multipliée par l'accélération angulaire.

En substituant les équations puis en réorganisant les termes pour isoler les dérivées temporelles, on obtient les deux équations différentielles fondamentales suivantes, qui décrivent le comportement dynamique du moteur :

$$\begin{aligned} \frac{di}{dt} &= -\frac{R}{L} \cdot i - \frac{K_e}{L} \cdot \omega + \frac{V_a}{L} \\ \frac{d\omega}{dt} &= \frac{K_t}{I_r} \cdot i - \frac{K_f}{I_r} \cdot \omega - \frac{T_a}{I_r} \end{aligned} \quad (2.4), (2.5)$$

L'équation (3.5) décrit la dynamique électrique du moteur. Elle montre que la dérivée temporelle du courant $\frac{di}{dt}$ dépend de la résistance R , de l'inductance L , de la constante électromotrice K_e , ainsi que de la tension d'entrée V_a .

Quant à l'équation (2.6), elle traduit la dynamique mécanique, reliant l'accélération angulaire à l'intensité du courant, à la friction, et au couple externe appliqué.

Dans l'équation (.6), la dérivée temporelle de la vitesse angulaire $\frac{d\omega}{dt}$ représente les dynamiques mécaniques du moteur. Elle dépend de la constante de couple K_t , du moment d'inertie I_r , de la constante de force contre-électromotrice K_e , ainsi que de tout couple externe T_a appliqué au moteur.

Dans le cas d'un robot en équilibre, un modèle simplifié du moteur à courant continu est souvent suffisant. On peut alors supposer que l'inductance du moteur ainsi que les effets de friction sont négligeables. Sous cette hypothèse, les équations du système peuvent être simplifiées comme suit :

$$\begin{aligned} i &= \frac{V_a - K_e \cdot \omega}{R} \\ \frac{d\omega}{dt} &= \frac{K_t}{I_r} \cdot i - \frac{T_a}{I_r} \end{aligned} \quad (2.6), (2.7)$$

L'équation (2.7) exprime le courant i en fonction de la vitesse angulaire ω de l'arbre moteur. Le terme $\frac{K_e}{R}$ traduit l'influence de la force contre-électromotrice sur le courant, tandis que le terme $\frac{1}{R} \cdot V_a$ reflète l'impact direct de la tension appliquée.

L'équation (2.8), quant à elle, décrit l'évolution de la vitesse angulaire. Elle montre que l'accélération angulaire est proportionnelle au courant i via le facteur $\frac{K_t}{I_r}$, et affectée par le couple externe T_a , pondéré par le moment d'inertie I_r .

En substituant l'équation (2.7) dans l'équation (2.8), on obtient une approximation du comportement d'un moteur à courant continu exprimée uniquement en fonction de la vitesse angulaire actuelle, de la tension appliquée et du couple externe. L'équation obtenue est la suivante :

$$\frac{d\omega}{dt} = \frac{K_m \cdot K_e}{I_r \cdot R} \cdot \omega + \frac{1}{I_r \cdot R} \cdot V_a - \frac{T_a}{I_r \cdot R} \quad (2.8)$$

Cette équation décrit la dynamique simplifiée du moteur. Elle montre que la vitesse angulaire évolue en fonction de la tension d'entrée, du couple externe appliqué et des paramètres physiques du moteur.

Les dynamiques du moteur peuvent être représentées par un **modèle en espace d'état**, constitué d'un ensemble d'équations différentielles du premier ordre. Les variables d'état sont la position angulaire θ et la vitesse angulaire ω , qui décrivent complètement l'état du moteur. Les entrées du système sont la tension appliquée V_a et le couple externe T_a .

Le modèle d'état peut être exprimé sous la forme suivante :

$$\begin{bmatrix} \dot{\theta} \\ \dot{\omega} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & -\frac{K_t \cdot K_e}{I_r \cdot R} \end{bmatrix} \begin{bmatrix} \theta \\ \omega \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ \frac{K_t}{I_r \cdot R} & -\frac{1}{I_r} \end{bmatrix} \begin{bmatrix} V_a \\ T_a \end{bmatrix} \quad (2.9)$$

Dans cette représentation :

- Le vecteur à gauche représente les dérivées temporelles des variables d'état $(\dot{\theta}, \dot{\omega})$.
- La première matrice

$$\begin{bmatrix} 0 & 1 \\ 0 & -\frac{K_t \cdot K_e}{I_r \cdot R} \end{bmatrix} \quad (2.10)$$

est appelée **matrice d'état** et capture l'évolution des variables d'état sans action extérieure.

- La deuxième matrice

$$\begin{bmatrix} 0 & 0 \\ \frac{K_t}{I_r \cdot R} & -\frac{1}{I_r} \end{bmatrix} \quad (2.11)$$

est appelée **matrice de commande**, et relie les entrées du système (tension et couple) aux dérivées des variables d'état.

L'équation de sortie du système est donnée par :

$$y = \begin{bmatrix} 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} \theta \\ \omega \end{bmatrix} + \begin{bmatrix} 0 & 0 \end{bmatrix} \cdot \begin{bmatrix} V_a \\ T_a \end{bmatrix} \quad (2.12)$$

Cette équation indique que la **sortie du système** est la position angulaire θ , sans contribution directe des entrées dans la mesure, ce qui est typique dans les systèmes où seule la position est mesurée.

2.3 Modèle dynamique pour un pendule inversé à deux roues

Le système de pendule inversé à deux roues présente un comportement similaire à celui d'un pendule monté sur un chariot, bien qu'il soit plus complexe en termes de dynamiques. Pour mieux comprendre ce comportement, On analyse d'abord séparément les dynamiques du pendule et des roues. Cette approche permet d'obtenir deux équations de mouvement qui décrivent entièrement le comportement du robot auto-équilibré.

Puisque le comportement du robot est influencé à la fois par des perturbations extérieures et le couple moteur, le modèle mathématique doit prendre en compte ces forces. On commence par dériver les équations de mouvement associées aux roues gauche et droite. La figure ci-dessous illustre le diagramme de corps libre pour les deux roues. Étant donné que les équations pour les roues gauche et droite sont analogues, seule celle de la roue droite est présentée [24].

En appliquant la deuxième loi de Newton à la direction horizontale (axe x), la somme des forces s'exprime comme suit :

$$\sum F_x = M \cdot \ddot{x} \quad (2.13)$$

où :

- M : Masse totale du robot.
- a : Accélération linéaire du centre de masse.

Pour le pendule inversé à deux roues, cette relation devient :

$$M \cdot w \cdot \ddot{x} = H_f - H \quad (2.14)$$

où :

- w : Empattement (distance entre les deux roues).
- $\ddot{x}$: Accélération linéaire du robot.
- H_f : Force horizontale générée par le couple moteur.
- H : Force horizontale externe (perturbation).

La somme des moments autour du centre de la roue est donnée par :

$$\Sigma M_0 = I_w \cdot \ddot{\theta}_w \quad (2.15)$$

$$M \cdot w \cdot \ddot{\theta}_w = C - H_f \quad (2.16)$$

où :

- I_w : Moment d'inertie de la roue.
- $\ddot{\theta}_w$: Accélération angulaire de la roue.
- C : Couple appliqué à la roue.
- H_f : Couple de réaction dû à la force de frottement.

La relation entre le couple moteur et la vitesse angulaire du moteur est donnée par :

$$T_m = I_r \cdot \frac{d\omega}{dt} + \tau_a \quad (2.17)$$

En réarrangeant cette équation et en substituant les paramètres électriques et mécaniques du moteur, le couple transmis à la roue devient :

$$C = \frac{K_t \cdot K_e}{R} \cdot \dot{\theta}_w + \frac{K_t}{R} \cdot V_a \quad (2.18)$$

Par conséquent, la force horizontale générée par le moteur peut être exprimée par :

$$H_f = -\frac{K_t \cdot K_e}{R \cdot r} \cdot \dot{\theta}_w + \frac{K_t}{R \cdot r} \cdot V_a - \frac{I_w}{r} \cdot \ddot{\theta}_w \quad (2.19)$$

L'équation du mouvement pour les roues devient alors :

$$M \cdot w \cdot \ddot{x} = -\frac{K_t \cdot K_e}{R \cdot r} \cdot \dot{\theta}_w + \frac{K_t}{R \cdot r} \cdot V_a - \frac{I_w}{r} \cdot \ddot{\theta}_w - H \quad (2.20)$$

La rotation angulaire peut être convertie en déplacement linéaire à l'aide de la relation géométrique suivante :

$$\theta_w \cdot r = x \Rightarrow \theta_w = \frac{x}{r}, \quad \dot{\theta}_w = \frac{\dot{x}}{r}, \quad \ddot{\theta}_w = \frac{\ddot{x}}{r}$$

En appliquant cette transformation, l'équation dynamique du système devient :

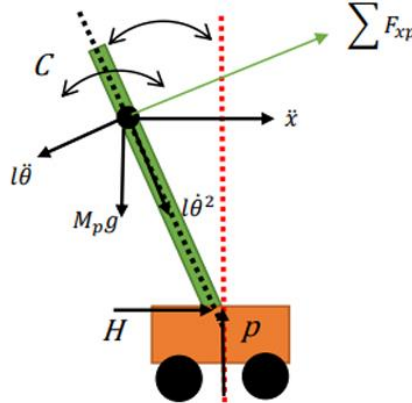
$$\theta_w \cdot r = x \Rightarrow \theta_w = \frac{x}{r}, \quad \dot{\theta}_w = \frac{\dot{x}}{r}, \quad \ddot{\theta}_w = \frac{\ddot{x}}{r} \quad (2.21)$$

Le châssis du robot, considéré comme un pendule inversé, sera modélisé dans la suite de cette section.

2.4 Modélisation dynamique du châssis du robot (pendule inversé)

Pour analyser le comportement dynamique du châssis du robot auto-équilibré, modélisé comme un pendule inversé, on applique les lois de Newton. On distingue les forces dans la direction horizontale et celles perpendiculaires à l'axe du pendule, ainsi que les moments agissant autour du centre de masse.

Figure II.2 : Diagramme du corps libre du châssis



La somme des forces dans la direction horizontale est donnée par :

$$\Sigma F_x = M_p \cdot \ddot{x} \quad (2.21)$$

La composante des forces perpendiculaires au pendule est exprimée comme :

$$\Sigma F_{xp} = M_p \cdot \ddot{x} \cdot \cos(\theta_p) \quad (2.22)$$

La somme des moments autour du centre de masse du châssis s'écrit :

$$\Sigma M_0 = I_p \cdot \ddot{\theta}_p \quad (2.23)$$

En remplaçant par les grandeurs pertinentes :

$$M_p \cdot l \cdot \cos\theta_p - P \cdot l \cdot \sin\theta_p - C = I_p \cdot \ddot{\theta}_p \quad (2.23)$$

Le couple C appliqué au pendule par le moteur, tiré de l'équation (2.18) après linéarisation, devient :

$$C = -\frac{2 \cdot K_t \cdot K_e}{R} \cdot \frac{\dot{x}}{r} + \frac{2 \cdot K_t}{R} \cdot V_a \quad (2.24)$$

En substituant cette expression dans l'équation (3.23), on obtient :

$$M_p \cdot l \cdot \cos\theta_p - P \cdot l \cdot \sin\theta_p = I_p \cdot \ddot{\theta}_p - 2 \cdot \frac{K_t \cdot K_e}{R \cdot r} \cdot \dot{x} + 2 \cdot \frac{K_t}{R} \cdot V_a \quad (2.24)$$

Ensuite, en multipliant l'équation (2.14) par $-l$, on obtient :

$$H \cdot l \cdot \cos\theta_p - P \cdot l \cdot \sin\theta_p + M_p \cdot g \cdot l \cdot \sin\theta_p + M_p \cdot l^2 \cdot \ddot{\theta}_p = -M_p \cdot l \cdot \ddot{x} \cdot \cos\theta_p \quad (2.25)$$

En remplaçant l'équation (2.23) dans l'équation (2.24), on obtient :

$$(I_p + M_p \cdot l^2) \cdot \ddot{\theta}_p - 2 \cdot \frac{K_t \cdot K_e}{R \cdot r} \cdot \dot{x} + 2 \cdot \frac{K_t}{R} \cdot V_a + M_p \cdot g \cdot l \cdot \sin\theta_p = -M_p \cdot l \cdot \ddot{x} \cdot \cos\theta_p \quad (2.26)$$

Afin d'éliminer la variable H des équations dynamiques, on substitue l'équation (2.24)

dans l'équation (2.19), ce qui donne :

$$2 \cdot \left(M_w + \frac{I_w}{r^2} \right) \cdot \ddot{x} = -2 \cdot \frac{K_t \cdot K_e}{R \cdot r^2} \cdot \dot{x} + 2 \cdot \frac{K_t}{R \cdot r} \cdot V_a + M_p \cdot l \cdot \dot{\theta}_p^2 \cdot \sin\theta_p - M_p \cdot l \cdot \ddot{\theta}_p \cdot \cos\theta_p - M_p \cdot \ddot{x} \quad (2.27)$$

2.5 Équations de mouvement non linéaires

En réarrangeant les équations précédentes, on obtient les équations de mouvement non linéaires décrivant le système complet :

$$(I_p + M_p \cdot l^2) \cdot \ddot{\theta}_p - \frac{2 \cdot K_m \cdot K_e}{R \cdot r} \cdot \dot{x} + \frac{2 \cdot K_m}{R} \cdot V_a + M_p \cdot g \cdot l \cdot \sin(\theta_p) = -M_p \cdot l \cdot \ddot{x} \cdot \cos(\theta_p) \quad (2.28)$$

$$\frac{2 \cdot K_m}{R \cdot r} \cdot V_a = \left(2 \cdot M_w + \frac{2 \cdot I_w}{r^2} + M_p \right) \cdot \ddot{x} + \frac{2 \cdot K_m \cdot K_e}{R \cdot r^2} \cdot \dot{x} - M_p \cdot l \cdot \dot{\theta}_p^2 \cdot \sin(\theta_p) + M_p \cdot l \cdot \ddot{\theta}_p \cdot \cos(\theta_p) \quad (2.29)$$

2.6 Linéarisation du modèle

Pour faciliter la conception d'un contrôleur, les équations précédentes sont linéarisées autour de la position d'équilibre vertical instable. On suppose :

$$\theta_p = \pi + \phi$$

où ϕ est un petit angle. Cela implique :

$$\cos\theta_p = -1, \quad \sin\theta_p = -\phi, \quad \dot{\theta}_p = \dot{\phi}, \quad \ddot{\theta}_p = \ddot{\phi} \quad (2.30)$$

Les équations deviennent alors :

$$(I_p + M_p \cdot l^2) \cdot \ddot{\phi} - \frac{2 \cdot K_t \cdot K_e}{R \cdot r} \cdot \dot{x} + \frac{2 \cdot K_t}{R} \cdot V_a - M_p \cdot g \cdot l \cdot \phi = M_p \cdot l \cdot \ddot{x} \quad (2.31)$$

$$\frac{2 \cdot K_t}{R \cdot r} \cdot V_a = \left(2 \cdot M_w + \frac{2 \cdot I_w}{r^2} + M_p \right) \cdot \ddot{x} + \frac{2 \cdot K_t \cdot K_e}{R \cdot r^2} \cdot \dot{x} + M_p \cdot l \cdot \ddot{\phi} \quad (2.32)$$

2.7 Représentation en espace d'état

Enfin, en isolant $\ddot{x}$ et $\ddot{\phi}$, on obtient les équations différentielles, prêtes à être exprimées sous forme d'état, où la force de perturbation H est prise en compte dans l'équation du mouvement linéaire.

$$\begin{aligned}\ddot{x} &= \frac{M_p \cdot l \cdot g \cdot \phi - \frac{2 \cdot K_t \cdot K_e}{R \cdot r} \cdot \dot{x} + \frac{2 \cdot K_t}{R} \cdot V_a}{I_p + M_p \cdot l^2 - M_p \cdot l} \\ \ddot{\phi} &= \frac{M_p \cdot l \cdot \ddot{x} + M_p \cdot g \cdot l \cdot \phi - \frac{2 \cdot K_t \cdot K_e}{R \cdot r} \cdot \dot{x} + \frac{2 \cdot K_t}{R} \cdot V_a}{I_p + M_p \cdot l^2}\end{aligned}\quad (2.33), (2.34)$$

2.8 Analyse et déduction de la fonction de transfert

Afin d'obtenir la fonction de transfert à partir du modèle en espace d'état du robot auto-équilibré, on doit dériver les relations entre les entrées et les sorties du système, en utilisant la représentation en espace d'état et les équations associées.

On a précédemment défini le vecteur d'état X comme :

$$X = \begin{bmatrix} x \\ \dot{x} \\ \phi \\ \dot{\phi} \end{bmatrix} \quad (2.35)$$

1. Détermination de la fonction de transfert :

À partir des équations en espace d'état, on peut calculer la fonction de transfert $G(s)$, qui décrit la relation entre l'entrée V_a (la tension appliquée aux moteurs) et la sortie $x(t)$ (le déplacement du robot).

Le modèle en espace d'état est donné par :

$$\dot{X}(t) = A \cdot X(t) + B \cdot V_a(t) \quad (2.36)$$

Où la matrice d'état A et la matrice de contrôle B sont définies comme suit :

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & \frac{2 \cdot K_t \cdot K_e \cdot (M_p \cdot l \cdot r - I_p \cdot l^2)}{R \cdot r^2 \cdot \alpha} & \frac{M_p \cdot g \cdot l^2}{\alpha} & 0 \\ 0 & 0 & 0 & 1 \\ 0 & \frac{2 \cdot K_t \cdot K_e \cdot (r \cdot \beta - M_p \cdot l)}{R \cdot r^2 \cdot \alpha} & \frac{M_p \cdot g \cdot l \cdot \beta}{\alpha} & 0 \end{bmatrix} \quad (2.37)$$

$$B = \begin{bmatrix} 0 \\ \frac{2 \cdot K_t \cdot (M_p \cdot l^2 - M_p \cdot l \cdot r + I_p)}{R \cdot r^2 \cdot \alpha} \\ 0 \\ \frac{2 \cdot K_t \cdot (M_p \cdot l - r \cdot \beta)}{R \cdot r \cdot \alpha} \end{bmatrix} \quad (2.38)$$

L'objectif est de relier la sortie $x(t)$ à l'entrée $V_a(t)$. La sortie Y est généralement définie comme le déplacement du robot, ce qui dans ce cas est $x(t)$. La matrice de sortie C est donc un vecteur ligne représentant cette relation :

$$C = [1 \quad 0 \quad 0 \quad 0] \quad (2.39)$$

La matrice de transmission directe D est souvent nulle dans les systèmes mécaniques comme celui-ci, ce qui simplifie l'expression de la sortie en fonction des entrées.

2. Fonction de transfert :

La fonction de transfert est dérivée en prenant la transformée de Laplace des équations d'état :

$$s \cdot X(s) = A \cdot X(s) + B \cdot V_a(s) \quad (2.40)$$

Cela se réécrit sous la forme suivante :

$$(sI - A) \cdot X(s) = B \cdot V_a(s)$$

De là, on peut résoudre pour $X(s)$:

$$X(s) = (sI - A)^{-1} \cdot B \cdot V_a(s)$$

Ensuite, en prenant la sortie $Y(s)$, qui est égale à $C \cdot X(s)$, on obtient la fonction de transfert $G(s)$:

$$G(s) = \frac{Y(s)}{V_a(s)} = C \cdot (sI - A)^{-1} \cdot B \quad (2.41)$$

Cela permet d'obtenir la fonction de transfert du système en termes des matrices A , B , et C .

3. Application des valeurs numériques :

Avec les valeurs physiques du robot (comme la masse du châssis, le rayon des roues, l'inertie des composants, etc.), on peut calculer les termes de la matrice A , B , et C et obtenir une représentation de la fonction de transfert numérique du système. Cela facilitera l'analyse du comportement dynamique du robot et la conception du contrôleur adapté pour maintenir l'équilibre du robot.

Résumé du processus de détermination de la fonction de transfert :

1. **Équations d'état** : Définir le modèle en espace d'état en termes de variables d'état et de contrôles (comme $x(t)$, $\dot{x}(t)$, $\phi(t)$, et $\dot{\phi}(t)$).
2. **Matrices d'état et de contrôle** : Identifier les matrices A , B , C (et D si nécessaire) pour le système.
3. **Transformée de Laplace** : Appliquer la transformée de Laplace pour obtenir la relation $G(s)$.
4. **Calcul de la fonction de transfert** : Résoudre $G(s) = C \cdot (sI - A)^{-1} \cdot B$ pour obtenir la fonction de transfert du système.

Cela permet d'analyser la réponse du système à différentes entrées et de concevoir des contrôles pour l'équilibre dynamique du robot.

Dans l'équation ci-dessus, le terme "den" représente le dénominateur des équations, qui est une combinaison des paramètres physiques du système, comme la masse du robot, les inerties, la longueur du bras, et d'autres facteurs. Cette dénomination permet de simplifier la présentation des équations, en rassemblant tous les facteurs communs dans un seul terme. L'expression complète du dénominateur "den" pourrait être représentée comme suit :

$$den = \left(m + 2 \cdot M + \frac{2 \cdot I_w}{r^2} \right) \cdot (m \cdot l^2 + I_c) - (m \cdot l)^2 \quad (2.42)$$

2.9 Calcul de la matrice d'état

Avec les équations ci-dessus, il est possible de dériver la matrice d'état A et la matrice d'entrée B qui décrivent le système dynamique du robot. Pour ce faire, on doit exprimer les dérivées des variables d'état en fonction des autres variables.

La matrice d'état A représente la relation entre les variables d'état et leurs dérivées, et la matrice d'entrée B représente comment les entrées du système affectent ces variables d'état. L'équation de l'état devient alors une relation matricielle :

$$\begin{bmatrix} \dot{x} \\ \ddot{x} \\ \dot{\phi} \\ \ddot{\phi} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & -\frac{(m \cdot l)^2 \cdot g}{den} & -\frac{2 \cdot \mu_0 \cdot (m \cdot l^2 + I_c)}{den} & \frac{2 \cdot m \cdot l \cdot \mu_1}{den} \\ 0 & 0 & 0 & 1 \\ 0 & \frac{\left(m + 2 \cdot M + \frac{2 \cdot I_w}{r^2} \right) \cdot m \cdot g \cdot l}{den} & \frac{2 \cdot \mu_0 \cdot m \cdot l}{den} & -\frac{2 \cdot \mu_1 \cdot \left(m + 2 \cdot M + \frac{2 \cdot I_w}{r^2} \right)}{den} \end{bmatrix}$$

$$\begin{bmatrix} x \\ \dot{x} \\ \phi \\ \dot{\phi} \end{bmatrix} + \begin{bmatrix} 0 \\ -\frac{m \cdot l}{den} \\ 0 \\ -\frac{(m + 2 \cdot M + \frac{2 \cdot I_w}{r^2})}{den} \end{bmatrix} (\tau_L + \tau_R) \quad (2.43)$$

Où τ_L et τ_R sont respectivement les couples appliqués par les moteurs gauche et droit.

Matrice de sortie (C)

La matrice de sortie C représente la relation entre les variables d'état et la sortie du système, qui peut être, par exemple, la position ou l'angle de tangage du robot. La sortie Y peut être exprimée comme suit :

$$Y = C \cdot X + D \cdot u \quad (2.44)$$

Supposons que la sortie Y soit simplement la position du châssis, on aura donc :

$$Y = [1 \ 0 \ 0 \ 0] \cdot \begin{bmatrix} x \\ \dot{x} \\ \phi \\ \dot{\phi} \end{bmatrix} \quad (2.45)$$

Cela signifie que C est un vecteur ligne $[1 \ 0 \ 0 \ 0]$, et la matrice D est souvent nulle pour ce type de système, car il n'y a pas de transmission directe entre l'entrée et la sortie.

Matrice de transmission directe (D)

La matrice D représente l'effet direct de l'entrée sur la sortie sans passer par les états. Dans la plupart des systèmes physiques, et en particulier pour un robot auto-équilibré, cette matrice est souvent nulle, car il n'y a pas d'effet immédiat de l'entrée sur la sortie sans qu'elle passe par les états.

2.10 Matrices en espace d'état du système d'équilibrage à deux roues

Les matrices d'état A , B , C , et D sont définies comme suit pour le système dynamique du robot auto-équilibré, représentant les dynamiques du robot en termes de ses variables d'état.

➤ Matrice d'état (A) :

La matrice A représente les relations entre les dérivées des variables d'état et les variables d'état elles-mêmes. Elle est donnée par :

$$A = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & -\frac{(m \cdot l)^2 \cdot g}{den} & -\frac{2 \cdot \mu_0 \cdot (m \cdot l^2 + I_c)}{den} & \frac{2 \cdot m \cdot l \cdot \mu_1}{den} \\ 0 & \frac{(m + 2 \cdot M + \frac{2 \cdot I_w}{R^2}) \cdot m \cdot g \cdot l}{den} & \frac{2 \cdot \mu_0 \cdot m \cdot l}{den} & -\frac{2 \cdot \mu_1 \cdot (m + 2 \cdot M + \frac{2 \cdot I_w}{R^2})}{den} \end{bmatrix} \quad (2.45)$$

Ici, den est le dénominateur utilisé dans les équations de dynamique du système, exprimé comme :

$$den = \left(m + 2 \cdot M + \frac{2 \cdot I_w}{R^2}\right) \cdot (m \cdot l^2 + I_c) - (m \cdot l)^2 \quad (2.46)$$

➤ **Matrice de contrôle (B) :**

La matrice B décrit la relation entre les entrées du système (comme la tension d'entrée u) et les variables d'état. Elle est donnée par :

$$B = \begin{bmatrix} 0 \\ 0 \\ \frac{m \cdot l}{den} \\ -\frac{\left(m + 2 \cdot M + \frac{2 \cdot I_w}{R^2}\right)}{den} \end{bmatrix} \quad (2.47)$$

➤ **Matrice de sortie (C) :**

La matrice C définit la sortie du système en termes des variables d'état. Dans ce cas, la sortie du système correspond aux positions x et ϕ .

$$C = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \quad (2.48)$$

➤ **Matrice de transmission directe (D) :**

La matrice D est souvent nulle dans les systèmes mécaniques comme celui-ci, car il n'y a généralement pas de lien direct entre l'entrée et la sortie sans passer par un processus dynamique. Par conséquent, D est :

$$D = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \quad (2.49)$$

2.11 Modèle en espace d'état simplifié

Le modèle simplifié en espace d'état, qui ne prend pas en compte les influences externes et représente uniquement la dynamique du robot en termes de ses variables d'état sans entrées, est :

$$\begin{bmatrix} \dot{x} \\ \ddot{x} \\ \dot{\phi} \\ \ddot{\phi} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} x \\ \dot{x} \\ \phi \\ \dot{\phi} \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \cdot u \quad (2.50)$$

Sortie du système

La sortie du système Y , qui est une combinaison linéaire des variables d'état, est donnée par :

$$Y = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} x \\ \dot{x} \\ \phi \\ \dot{\phi} \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \end{bmatrix} \quad (2.51)$$

Ici, Y représente les positions $x(t)$ et $\phi(t)$ du robot en fonction de son vecteur d'état.

2.12 Commandabilité

Matrice de commandabilité :

$$C = \begin{bmatrix} B & AB & A^2B & A^3B \end{bmatrix} \quad (2.52)$$

Déterminant :

$$\det(C) = \frac{(2M+m)^3 m^3 g^2 L^2 K_t^4}{r^4 D^6} \neq 0 \quad (2.53)$$

Le système est donc **complètement commandable**.

2.13 Résumé

Les matrices A , B , C , et D du système définissent entièrement la dynamique du robot d'équilibrage à deux roues en termes de ses variables d'état, permettant de simuler et d'analyser son comportement dynamique. La fonction de transfert et le modèle en espace d'état peuvent ensuite être utilisés pour concevoir un contrôleur efficace pour maintenir l'équilibre du robot.

Chapitre III
Conception et Construction du
Robot

3.1 Architecture Logicielle

Présentation de l'architecture logicielle :

L'architecture logicielle du robot auto-équilibré est conçue pour assurer une gestion robuste et modulaire des fonctions nécessaires à son fonctionnement. Il intègre l'acquisition de données, le traitement de l'information, l'algorithme de contrôle, la communication externe et le mécanisme d'alarme.

3.1.1 Définition de l'IDE Arduino :

 L'**Arduino IDE** (Integrated Development Environment) est un logiciel open source conçu pour programmer des microcontrôleurs, principalement des cartes Arduino, mais il est également compatible avec d'autres comme l'ESP32. Il fournit une interface graphique intuitive pour écrire, compiler et télécharger du code sur la carte via une connexion USB ou sans fil. L'IDE comprend un éditeur de texte, un compilateur et une interface pour gérer la communication avec le matériel.

3.1.2 Langage utilisé :

Le langage utilisé dans l'IDE Arduino est une version simplifiée de C++, appelée **langage Arduino** ou **Wiring**. Il s'agit d'un sous-ensemble de C++ adapté aux microcontrôleurs, avec des bibliothèques et des fonctions spécifiques pour faciliter la gestion des entrées/sorties et des périphériques (capteurs, moteurs, etc.). Il conserve la syntaxe du C++ tout en fournissant des abstractions pour rendre la programmation accessible.

3.1.3 Comparaison C++ (Arduino IDE) vs Python (MicroPython) sur ESP32 :

Pourquoi C++ ? Voici un tableau concis comparant les performances de C++ (via Arduino IDE) et Python (via MicroPython) sur un ESP32.

Aspect	C++ (Arduino IDE)	Python (MicroPython)
Vitesse d'exécution	Élevée : Code compilé, rapide pour le contrôle en temps réel (ex. : PID, capteurs).	Faible : Interprété, plus lent, moins adapté aux tâches rapides.
Utilisation mémoire	Efficace : Faible empreinte, idéal pour les 520 Ko de RAM de l'ESP32.	Élevée : Consomme plus de RAM et flash, limite les applications complexes.
Facilité de développement	Modérée : Syntaxe C++ plus complexe, mais précise pour le matériel.	Élevée : Syntaxe simple, rapide à coder, mais moins précis.

Contrôle matériel	Excellent : Accès direct aux broches et périphériques (ex. : moteurs, buzzer).	Limité : Abstraction simplifiée, moins performant pour le contrôle précis.
Pertinence pour le robot	Idéal : Parfait pour l'équilibre, les capteurs et Bluetooth de votre robot.	Peu adapté : Convient au prototypage, mais trop lent pour votre projet.

Tableau III.1: Comparaison C++ (Arduino IDE) vs Python (MicroPython) sur ESP32

Remarques

- C++ est optimal pour un robot auto-équilibré grâce à sa rapidité et son contrôle précis.
- Python est plus simple mais trop limité pour les besoins en temps réel de votre projet.

3.2 Les principaux composants du logiciel

3.2.1 Obtenir des données à partir des capteurs :

Le robot utilise le capteur MPU6050 pour collecter les données d'accélération (axes X, Y, Z) et de vitesse angulaire (gyroscope) afin de calculer l'angle d'inclinaison. Ces données sont obtenues via une communication I2C, implémentée à l'aide des bibliothèques « Wire.h » et « MPU6050.h ». Le capteur à ultrasons connecté aux broches « trigPin » et « echoPin » peut mesurer la distance aux obstacles pour éviter les collisions. De plus, les encodeurs (broches « ENCODER_RIGHT_A », « ENCODER_RIGHT_B », « ENCODER_LEFT_A », « ENCODER_LEFT_B ») fournissent des informations sur la vitesse de la roue. Avant le traitement, les données brutes ont été filtrées pour garantir leur exactitude.

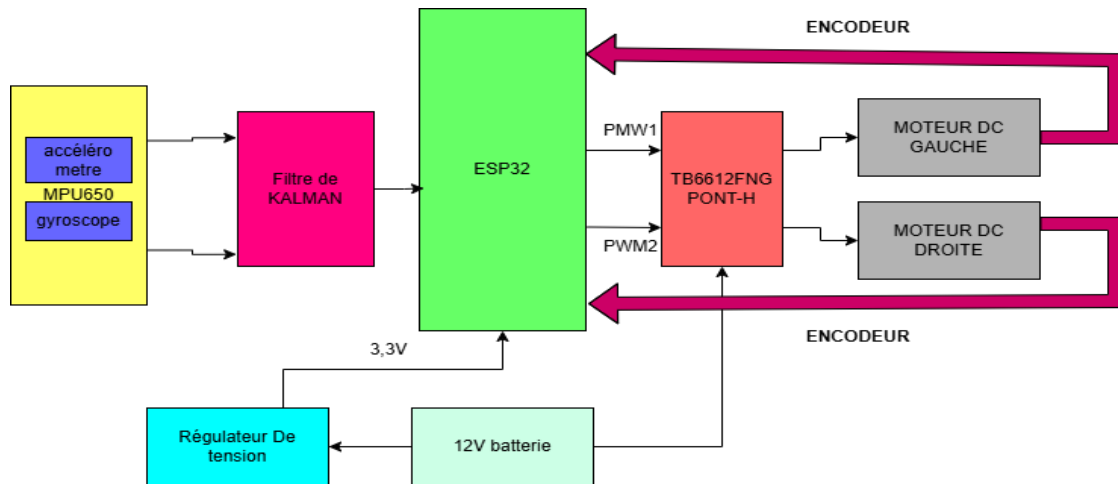


Figure III.1 : Architecture générale du robot.

3.2.2 Mise en œuvre de l'algorithme de contrôle :

Le contrôle de l'équilibre est basé sur des algorithmes avancés. Le filtre de Kalman combine les données d'accélération et de gyroscope pour estimer avec précision l'angle d'inclinaison (fonction « Kalman_Filter »), tandis qu'un filtre complémentaire est appliqué à l'axe Y (fonction « Yiorderfilter »). Le contrôle du moteur est assuré par trois boucles PID :

- La boucle PID angulaire (paramètres "kp", "ki", "kd") maintient l'équilibre du robot.
- La boucle PID de vitesse (paramètres "kp_speed", "ki_speed", "kd_speed") régule la vitesse du moteur en fonction des données du codeur.
- La boucle PID pour la direction (paramètres "kp_turn", "ki_turn", "kd_turn") gère les virages.

Ces algorithmes permettent un contrôle précis de l'équilibre, de la vitesse et de la direction. De plus, la fonction de chute prédite arrête le moteur en cas de risque, améliorant ainsi la sécurité.

3.2.3 Communication Bluetooth avec l'application :

La communication avec les applications externes est réalisée via le module Bluetooth, qui est implémenté par la bibliothèque « BluetoothSerial.h ». Le robot, nommé « ESP32_Robot », transmet des données telles que l'angle d'inclinaison, la vitesse des roues et la détection d'obstacles en temps réel (via « SerialBT.println »). Il reçoit également des commandes pour ajuster les paramètres PID, contrôler le mouvement (avancer, reculer, tourner à gauche, tourner à droite, s'arrêter) ou effectuer des actions spécifiques telles qu'une rotation à 360

degrés (fonction «rotate360 »). Lorsqu'un obstacle est détecté, la commande est ignorée pour privilégier la sécurité (fonction « processCommand »).

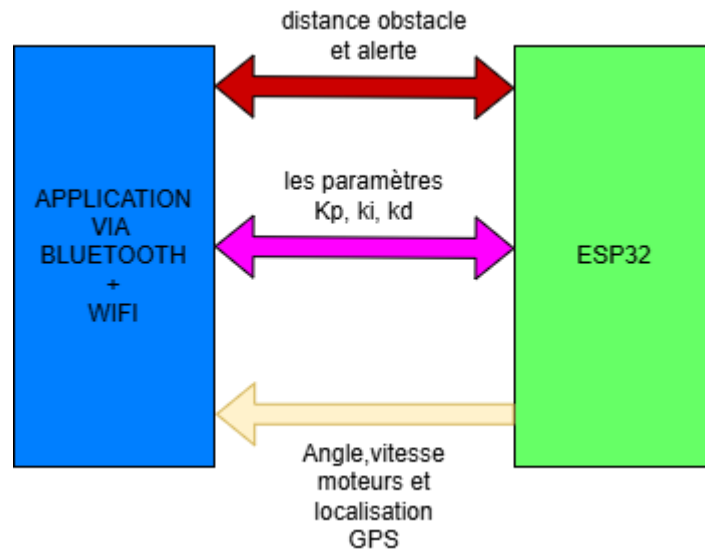


Figure III.2 : Interface de communication Bluetooth +Wifi entre le robot et l'application externe.

3.2.4 Système d'alarme par buzzer :

Un buzzer connecté à la broche « buzzerPin » est utilisé pour signaler divers événements clés. Une alarme sonore retentira dans les situations suivantes :

- Connexion Bluetooth établie: Deux bips courts consécutifs (« beepBeep ») indiquent une connexion réussie.
- Détection d'obstacle: Lorsque le capteur à ultrasons détecte un obstacle à moins de 15 cm, un bip sonore continu est émis, indiquant que le robot s'arrête.
- Risque de chute: Si l'angle d'inclinaison dépasse 30 degrés et que la vitesse angulaire dépasse 50 degrés par seconde, un bip plus grave (« fallWarningBeep ») retentit pour avertir d'un risque de chute imminent.

Ces alarmes sonores améliorent la réactivité et la sécurité du robot en avertissant l'utilisateur des situations critiques.

Donner la priorité à la sécurité lorsque des obstacles sont détectés (fonction 'processCommand').

L'architecture logicielle permet au robot de maintenir l'équilibre, d'éviter les obstacles, d'interagir avec les utilisateurs et de signaler efficacement les événements critiques en

intégrant l'acquisition de données, les algorithmes de contrôle, la communication Bluetooth et les systèmes d'alarme sonore.

3.3 Partie électrique

La création d'un système de robot qui se bouge seule dépend d'un bon choix de pièces électroniques. Chaque pièce fait quelque chose de spécifique pour que le tout marche bien. dans cette parti on parle des gros élément utiliser dans notre projet entrant dans leur caractéristique technique et leur partie à jouer dans tout cela la pertinence dans la robotique actuelle .

3.3.1 ESP32 microcontrôleur :

L'ESP32, faite par Espressif, est un petit ordinateur utile pour plein de projets sur l'Internet des Choses (IoT). Cette pièce sert comme "cerveau" du système avec ses bonnes capacités pour traiter et se connecter sans fil. D'après la doc officielle d'Espressif, l'ESP32 a un processeur deux-cœur

Tensilica LX6 qui marche à 240 MHz, ce qui aide à faire plusieurs tâches difficiles en même temps. Il a aussi des parties pour Wi-Fi (802.11 b/g/n) et Bluetooth (Classic et BLE), rendant plus facile la communication avec des appareils extérieurs ou d'autres interfaces. Sa mémoire à bord, qui comprend 520 Ko de RAM et jusqu'à 4 Mo de flash, donne assez d'espace pour faire tourner des algorithmes de navigation et de contrôle en temps réel.

Dans notre système, l'ESP32 joue un rôle principal en guidant les différentes tâches. Il gère les informations des capteurs, crée des signaux de commande pour les moteurs avec le pont en H, et fait aussi la communication sans fil pour une interaction extérieure. Sa compatibilité avec des règles comme I2C, SPI et UART le rend un grand choix pour mettre ensemble des éléments comme le gyroscope MPU6050 et le capteur sonore.

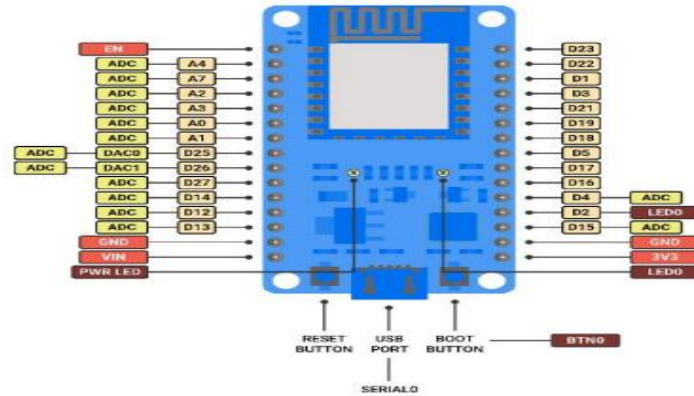


Figure III.3 : Schéma de l'ESP32 et ses principales interfaces.

3.3.2 TB1266FNG pont en H :

Le TB1266FNG est un gérant de moteur à pont en H fait pour guider des moteurs à courant continu qui vont dans deux sens. Ce morceau est très important pour faire bouger les moteurs soit vers l'avant soit vers l'arrière tout en changeant leur rapidité par le biais de signes PWM (Pulse Width Modulation). D'après les spécifications techniques données par Toshiba, le TB1266FNG peut marcher avec une tension d'alimentation qui va de 4,5 V à 13,5 V, un courant de sortie maximal de 1,5 A. Il a aussi des protections contre la surchauffe, les hautes tensions et courts circuits ce qui garantit bonne fiabilité dans des milieux dynamiques.

Dans notre système, le TB1266FNG a le rôle d'un lien fort entre le ESP32 et les moteurs. Il renforce les signaux faibles de commande faits par le microcontrôleur pour donner assez d'énergie aux moteurs. Avec ses broches pour diriger (IN1, IN2 et EN), il peut changer la voie des moteurs et ajuster leur rapidité avec attention, permettant des mouvements faciles et contrôlés.

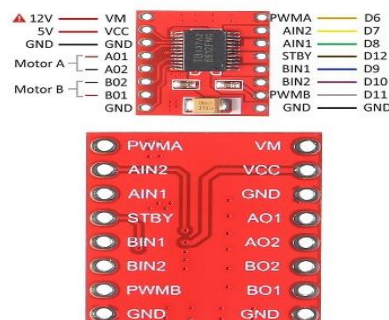


Figure III.4 : Configuration du pont en H TB1266FNG pour le contrôle des moteurs.

3.3.3 Gyroscope MPU6050 (et accéléromètre intégré) :

Le MPU6050 est un module qui mêle un gyroscope et un accéléromètre sur 3 axes, offrant de bonnes mesures d'orientation et d'accélération droite. Ce composant est surtout utile pour les applications qui ont besoin une stabilisation ou corriger orientation comme robots bipèdes ou véhicules à deux roues. Selon InvenSense , le fabricant du MPU6050, ce module peut prendre des vitesses angulaires jusqu'à ± 2000 °/s et une accélération linéaire jusqu'à $\pm 16g$ avec résolution ajustable selon besoin appli.

Dans notre projet, le MPU6050 est utilisé pour trouver l'inclinaison et la rotation du robot, ainsi que pour mesurer les mouvements droits et les chocs reçus. Son DMP (Digital Motion Processor) à l'intérieur aide à gérer les données tout de suite, rendant plus simple le fait de mettre en place un algorithme pour stabiliser la trajectoire. Ce composant est donc très important pour tenir le robot en équilibre et changer sa route automatiquement.

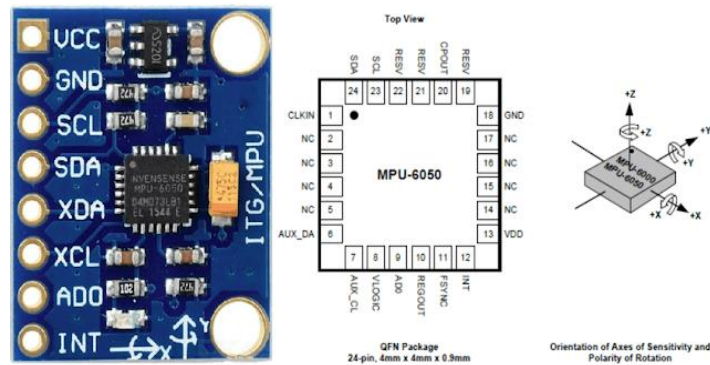


Figure III.5: Module gyroscope et accéléromètre MPU6050 avec ses axes de mesure.

3.3.4 Capteur ultrasonique :

Le capteur d'ondes sonores HC-SR04 est un outil simple mais bien pour mesurer la distance entre le robot et ce qui l'entoure. Il marche en envoyant des petites ondes sonores à une fréquence de 40 kHz et en disant la distance avec le temps que l'écho prend pour revenir. D'après les règles du fabricant, ce capteur montre une plage allant de 2 cm à 400 cm, avec une justesse d'environ ± 3 mm.

Dans notre système, le détecteur sonore a un rôle important dans le mouvement tout seul. Il aide au robot à trouver des obstacles et à changer son chemin en suivant. Cette partie est aussi employée pour faire des cartes ou suivre le chemin, ce qui rend une navigation plus sûre et très précise.

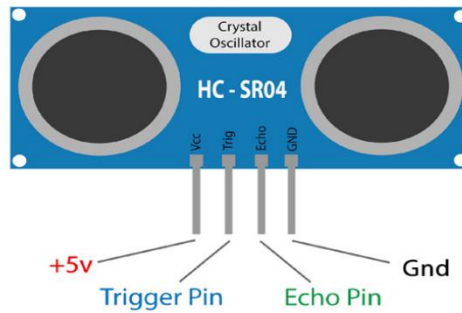


Figure III.6 : Capteur ultrasonique HC-SR04 et son principe de mesure de distance.

3.3.5 Moteurs avec encoders et roues :

Les moteurs utilisés dans notre système sont des moteurs CC avec encodeurs incrémentaux, qui donnent des informations précises sur la place et la vitesse des roues. Les encodeurs incrémentaux font des impulsions à chaque tour partiel d'une roue, aidant à mesurer distance parcourue et vitesse instantanée. Selon Polou Robotics & Electronics , ces encodeurs sont très importants pour faire des algorithmes d'odométrie. Ces algorithmes permettent de suivre où se trouve le robot dans son espace.

Dans notre plan les moteurs avec encodeurs aident à bouger d'une manière précise et maîtrisée. Ils servent à faire avancer, reculer ou tourner le robot, et donnent aussi des infos sûres pour réctifier les écarts causés par des zones inégales ou des erreurs de trajet. Les roues, elles, sont choisies pour leur adhérence et leur accord avec le poids et la taille du robot.



Figure III.7 : Moteur à courant continu avec encodeur incrémental et ses caractéristiques.

Caractéristique	Valeur
Tension de fonctionnement	DC 12V
Rapport de réduction	1:30

Caractéristique	Valeur
Courant à vide	≤ 100 mA
Vitesse à vide	247 tr/min
Couple nominal	1,4 Kg.cm
Couple nominal	137,3 mN.m
Vitesse nominale	160 tr/min
Courant nominal	$\leq 0,45$ A
Couple de blocage	5,5 Kg.cm
Courant d'arrêt	2,4 A
Longueur du réducteur	22 mm

Tableau III .2 : Caractéristiques techniques du moteur à courant continu

3.3.6 Module GPS GY-NEO6MV2 :

Le module GPS GY-NEO6MV2, basé sur la puce u-blox NEO-6M, permet la localisation précise du robot auto-équilibré (latitude, longitude) avec une précision de 2,5 m. Alimenté en 3-5 V, il communique via UART (9600 bauds) au format NMEA, transmettant les données à l'application Android pour affichage sur Google Maps. Avec une antenne céramique, il suit jusqu'à 22 satellites. Une EEPROM et une batterie de sauvegarde assurent un démarrage rapide (27 s à froid, 1 s à chaud). Compact et économique, ce module est idéal pour le suivi en temps réel dans notre projet.

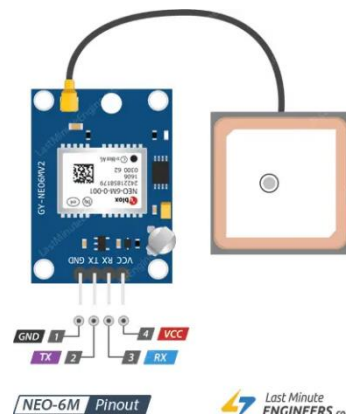


Figure III.8 : Module GPS GY-NEO6MV2.

3.3.7 Batteries 18650 :

Les batteries 18650 sont des batteries lithium-ion rechargeables cylindriques de 18 mm de diamètre et de 65 mm de longueur. Chaque cellule a une tension nominale de 3,7 V et une capacité typique de 2 000 à 3 500 mAh . Grâce à leur densité énergétique élevée, leur fiabilité et leur disponibilité, ces batteries sont largement utilisées dans les applications nécessitant une source d'alimentation compacte et puissante, comme les robots auto-équilibrés.



Figure III.9 : Batteries 18650.

3.3.8 LM2596 :

Le LM2596 est un régulateur abaisseur de tension (buck) largement utilisé pour convertir efficacement une tension d'entrée élevée en une tension de sortie plus faible. Il peut fournir jusqu'à 3 A de courant et accepte une plage de tension d'entrée typique de 4,5 V à 40 V. Le LM2596 est couramment utilisé dans les circuits électroniques pour alimenter des composants nécessitant une tension régulée, tels que les microcontrôleurs ou les LED. Il est populaire dans les projets DIY et les applications industrielles en raison de son rendement énergétique élevé et de sa simplicité d'utilisation.

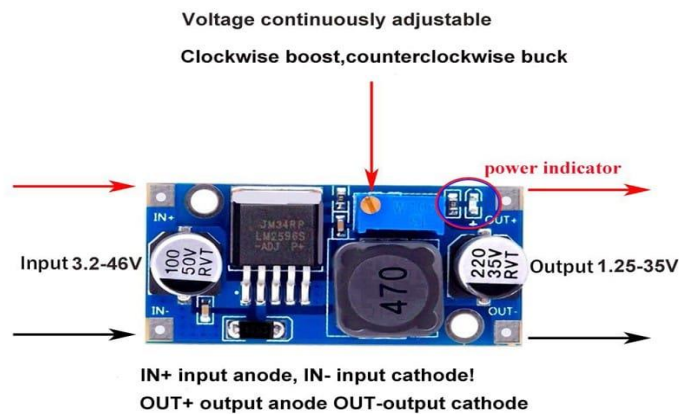


Figure III.9 : Module LM2596 avec entrée et sortie .

3.3.9 Importance des condensateurs dans le Projet :

Les condensateurs jouent un rôle essentiel dans la conception et le fonctionnement des robots auto-équilibrés, notamment dans la gestion de l'électronique embarquée. Ces composants assurent la stabilité de l'alimentation électrique en filtrant les fluctuations de tension, ce qui est essentiel pour garantir un fonctionnement fiable des microcontrôleurs et des capteurs tels que les gyroscopes et les accéléromètres utilisés pour mesurer les angles d'inclinaison. De plus, les condensateurs aident à réduire le bruit électrique dans le circuit de commande du moteur, améliorant ainsi la précision des signaux de commande appliqués aux actionneurs. Dans les systèmes dynamiques tels que les robots auto-équilibrés, qui nécessitent des réponses rapides pour maintenir l'équilibre, la capacité des condensateurs à stocker et à libérer rapidement de l'énergie contribue à améliorer la robustesse et la réactivité du système. Par conséquent, leur sélection et leur intégration sont des aspects fondamentaux pour optimiser les performances et la fiabilité des prototypes, les condensateurs utilisées sont (0.1µF céramique et 10 µF électrolytique) pour broche VCC du pont H et 100µF électrolytique pour broche VM du pont H aussi.

Chaque partie mentionnée plus haut a un rôle précis dans le système global. L'ESP32 sert de chef central, tandis que le TB1266FNG gère les moteurs, le MPU6050 donne des informations sur l'orientation et la vitesse, le capteur ultrasonique aide à trouver des

obstacles, et les moteurs avec encodeurs permettent un bougement exacte et contrôlé En gros ces composants font un fort système qui peut se déplacer seul et bien communiquer avec son environnement bien que le GPS GY-NEO6MV2 pour localiser le robot .

3.4 Conception du PCB

EasyEDA est un outil gratuit de conception de cartes de circuits imprimés (PCB) en ligne qui fournit une interface intuitive pour la saisie de schémas, la simulation électronique et la conception de PCB. Il permet de réaliser des projets simples ou complexes sans installer de logiciel local.

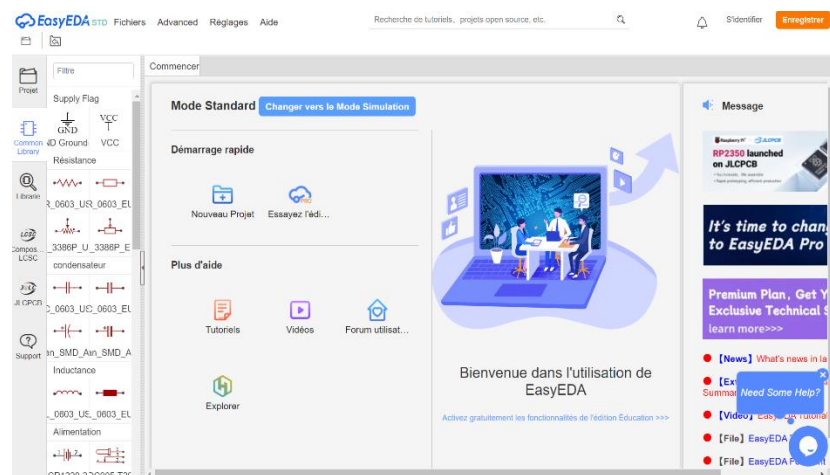


Figure III.11: Interface de conception de PCB avec EasyEDA.

3.4.1 Étapes pour créer un PCB à l'aide d'EasyEDA :

➤ Saisie du Schéma

- Ouvrez un nouveau projet dans EasyEDA.
- Placez les composants de la bibliothèque et connectez-les avec des fils.
- Utilisez l'outil ERC (Electrical Rules Check) pour vérifier les erreurs électriques.

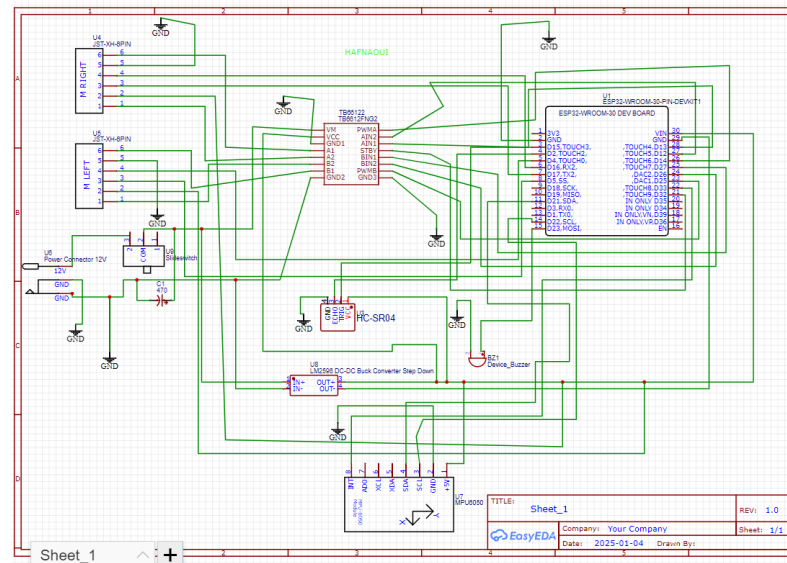


Figure III.12 : Exemple de saisie de schéma électronique dans EasyEDA.

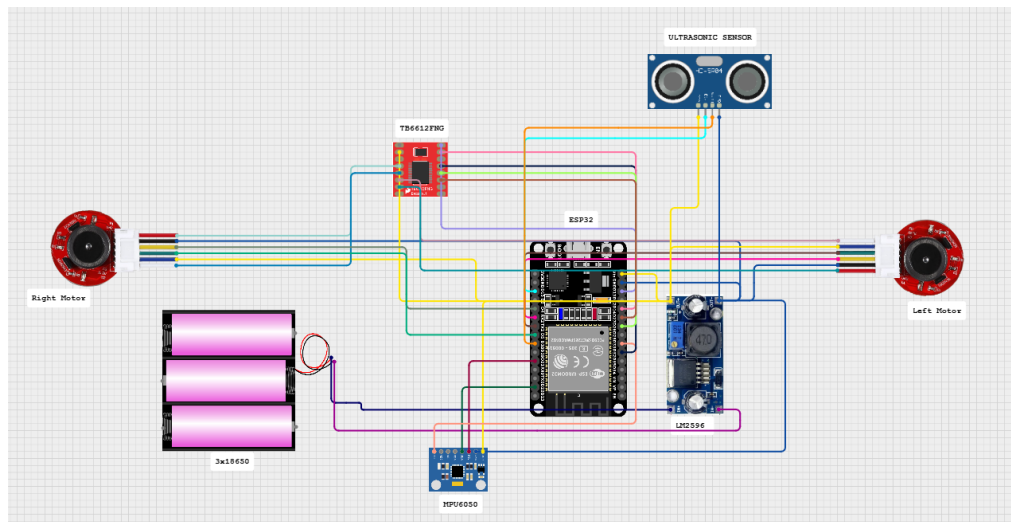


Figure III.13 : Exemple de Cablage de schéma électronique dans Circuit.

➤ Convertir en PCB

- Passez en mode conception de PCB via Conception > Convertir en PCB.
- Disposez les composants sur la carte pour optimiser l'espace et les chemins de routage.
- Route orbitale

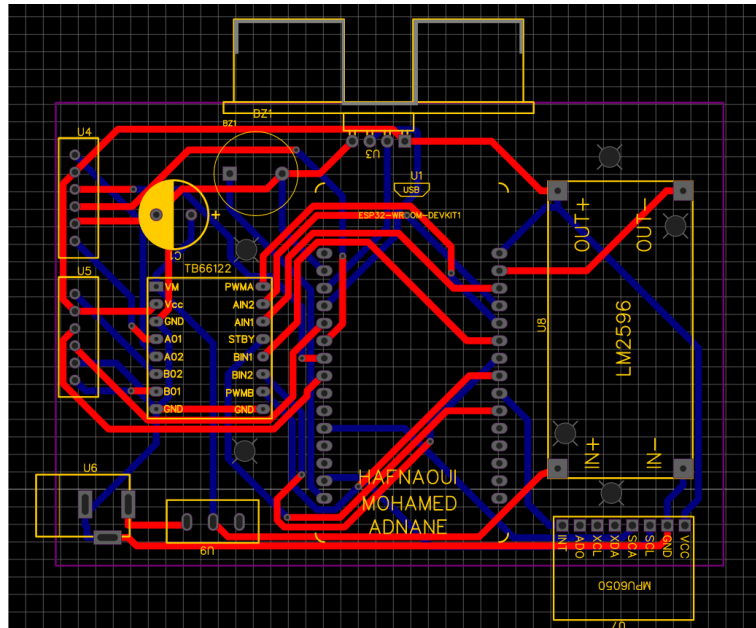


Figure III.14 : Disposition des composants sur le PCB après routage.

➤ **Routage des pistes :**

✓ Définir l'épaisseur de la piste en termes de courant :

- 0,3 mm (12 mil) : 1 A
- 0,5 mm (20 mil) : 2 A
- 1,0 mm (40 mil) : 4 A (pour des courants plus élevés)

✓ Utilisation des outils de routage automatique pour le routage automatique ou le routage manuel pour un meilleur contrôle.

➤ **Validation des règles (RDC) :**

- Configuration des règles de conception : espacement, largeur minimale, etc.
- Exécution DRC (Design Rule Check) pour détecter les erreurs.

➤ **Exporter des fichiers Gerber :**

- Fabrication > Exporter Gerber.
- Sélectionner les calques nécessaires (couche supérieure, couche inférieure, couche de sérigraphie, etc.).
- Téléchargement du fichier .zip et envoyez-le au fabricant du PCB.

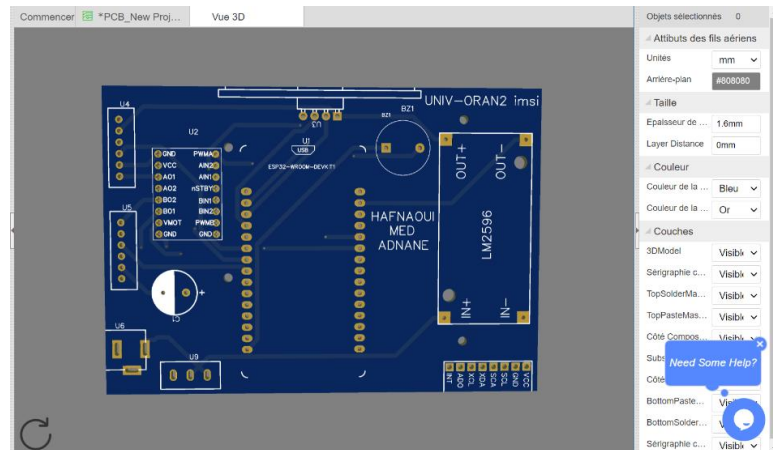


Figure III.14 : Exportation des fichiers Gerber pour fabrication du PCB.

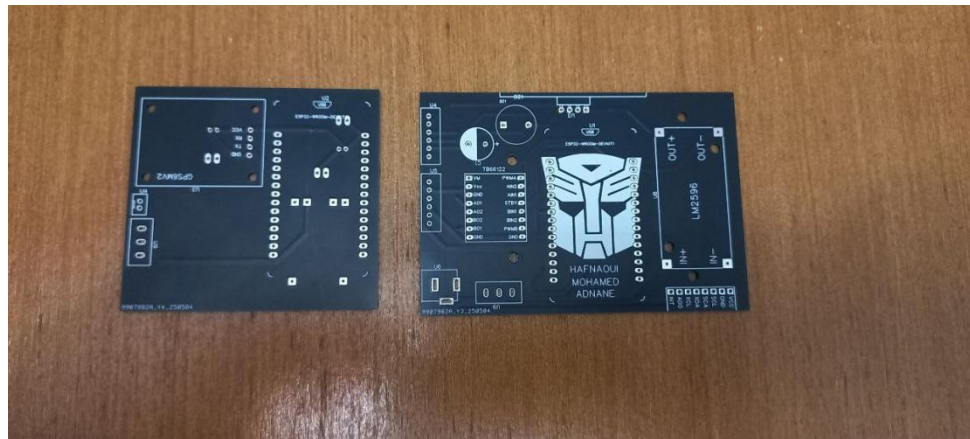


Figure III.15 : PCBs Fabriqués (PCB GPS / PCB du contrôle)

3.5 Développement de l'Application Mobile pour le Contrôle du Robot Auto-Équilibré

Dans le cadre de ce mémoire sur les robots auto-équilibrés, une application mobile a été développée pour assurer une interaction en temps réel avec le système robotique.

Applications mobiles créées en utilisant Android Studio comme environnement de développement intégré (IDE). Le code est écrit en langage Java, qui a été choisi pour sa robustesse et sa compatibilité avec la plateforme Android. L'objectif était de créer une interface utilisateur intuitive qui permettrait de contrôler le robot via une connexion Bluetooth, d'ajuster les paramètres PID (proportionnel, intégral, dérivé) pour optimiser l'équilibre et de visualiser des données en temps réel telles que l'angle d'inclinaison, les

vitesse du moteur gauche et droit en mètres par seconde (m/s) via des encodeurs et d'autres retours du robot. Des composants tels que des boutons permettant d'établir ou de déconnecter la connexion au robot, des arrêts d'urgence et des curseurs permettant de modifier dynamiquement les coefficients PID ont été intégrés. Ajout d'un affichage en temps réel des vitesses du moteur gauche et droit calculées à partir des données de l'encodeur pour une meilleure surveillance des performances. De plus, une fonction de retour audio est implémentée pour signaler les obstacles, améliorant ainsi la sécurité et l'interactivité de l'application.

 Android Studio est l'IDE officiel de Google pour le développement d'applications Android. Il fournit des outils puissants, notamment un éditeur de code intelligent, un simulateur et un système de gestion des dépendances (Gradle). Ces fonctionnalités permettent de concevoir des interfaces graphiques et d'intégrer des bibliothèques externes, telles que celles pour la communication Bluetooth et la visualisation de graphiques (via MPAndroidChart).

 Java est un langage de programmation orienté objet largement utilisé dans le développement Android en raison de sa portabilité, de sa gestion automatique de la mémoire (via un garbage collector) et de sa riche bibliothèque standard. Ce projet utilise Java pour implémenter la logique de contrôle, gérer les threads de communication Bluetooth et organiser le code selon le paradigme MVC (Modèle-Vue-Contrôleur) en utilisant ViewModel.

3.5.1 Architecture de l'Application :

La communication entre l'application mobile et le robot auto-équilibré repose sur le protocole Bluetooth, une technologie sans fil puissante et économe en énergie particulièrement adaptée aux systèmes embarqués. Dans le code, la classe BluetoothAdapter initialise la connexion, tandis que le BluetoothSocket établit un canal bidirectionnel avec le robot en utilisant un UUID prédéfini (00001101-0000-1000-8000-00805F9B34FB), correspondant au profil de port série (SPP). Les données telles que les commandes de contrôle (telles que l'arrêt ou le réglage PID) et les retours du robot (angle, vitesse) sont échangées via InputStream et OutputStream. Cette communication asynchrone est gérée par des gestionnaires et des boucleurs pour éviter de bloquer le thread principal de l'interface utilisateur, garantissant une expérience fluide.

Initialisez les composants d'interface dans la méthode onCreate, y compris les TextViews qui affichent l'angle (textAngle) et la vitesse du moteur (textRightSpeed, textLeftSpeed) et

les boutons (btnConnect, btnStop, btnGraph). Utilisez ViewModel pour séparer la logique métier des données d'affichage, améliorant ainsi la maintenabilité et la testabilité du code.

Le bouton de connexion déclenche checkPermissionsAndConnect() dans le ViewModel, qui gère les autorisations Android et établit la connexion Bluetooth.

Le bouton d'arrêt envoie une commande immédiate au robot via sendStopCommand().

➤ **Intégration de la fonctionnalité GPS à l'API Google Maps :**

La fonctionnalité de suivi GPS est implémentée dans une activité dédiée, « MapActivity », qui utilise l'API Google Maps pour afficher la position du robot auto-équilibré sur une carte interactive. La bibliothèque « com.google.android.gms:play-services-maps » a été intégrée au projet Android Studio et la clé API a été configurée dans le fichier « AndroidManifest.xml ». La carte est chargée via « SupportMapFragment » dans le fichier « activity_map.xml » via l'interface « OnMapReadyCallback », qui configure un objet « GoogleMap ».

Les coordonnées GPS (latitude et longitude) sont obtenues via une communication Wi-Fi (décrite ci-dessous) et sont affichées sous forme de marqueurs intitulés « Position du robot ». La polyligne bleue créée avec « PolylineOptions » trace le trajet du robot en fonction de la liste des points de cheminement. La méthode « CameraUpdateFactory.newLatLngZoom » centre la carte sur la position du robot avec un zoom de 15. Les autorisations de localisation « ACCESS_FINE_LOCATION » sont gérées dynamiquement via « ActivityResultLauncher » ; en cas de refus, un message d'erreur « Toast » s'affiche afin de garantir la conformité aux exigences de sécurité Android.

➤ **Communication Wi-Fi avec le serveur 192.168.4.1 :**

Les données de localisation sont récupérées via une connexion Wi-Fi à un serveur local « http://192.168.4.1/gpsdata », éventuellement hébergé par le robot. Les requêtes HTTP GET sont envoyées dans un thread séparé via « HttpURLConnection », avec un délai d'expiration de connexion et de lecture de 5 secondes. Les données JSON reçues contiennent un champ « valid » indiquant la validité des coordonnées, des champs « latitude » et « longitude » indiquant la position actuelle, et un tableau « path » facultatif indiquant les points de cheminement.

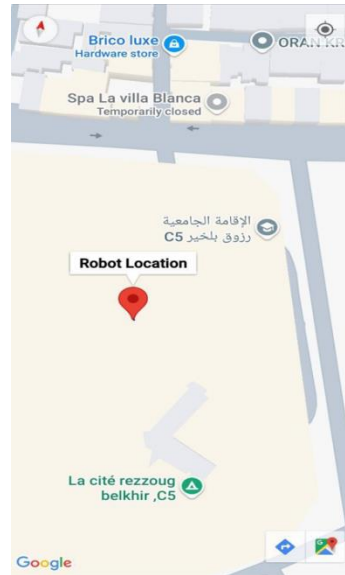


Figure III.16 : Capture d'écran de localisation du robot sur l'application

➤ **Bouton d'activation de l'IA :**

L'interface MainActivity intègre un bouton « Activer l'IA » qui active ou désactive le mode d'intelligence artificielle du robot, permettant ainsi d'optimiser l'équilibre ou la navigation autonome. Ce bouton est utilisé conjointement avec un OnClickListener pour envoyer une commande Bluetooth (AI:1 pour activer, AI:0 pour désactiver) via l'OutputStream du socket RFCOMM. Un message « Toast » confirme l'envoi de la commande, et les erreurs sont enregistrées via un autre « Toast » pour fournir un retour clair à l'utilisateur.

➤ **Rotation à 360 degrés :**

La fonction de rotation à 360 degrés permet au robot d'effectuer une rotation complète sur place, utile pour l'exploration ou l'étalonnage. Un bouton dédié dans MainActivity déclenche l'envoi d'une commande Bluetooth (ROT:360) via le flux de sortie. L'application vérifie la connexion Bluetooth avant l'envoi et indique la réussite ou l'échec de la transmission par un message « Toast ». La commande est interprétée par le microcontrôleur du robot pour effectuer une rotation précise.

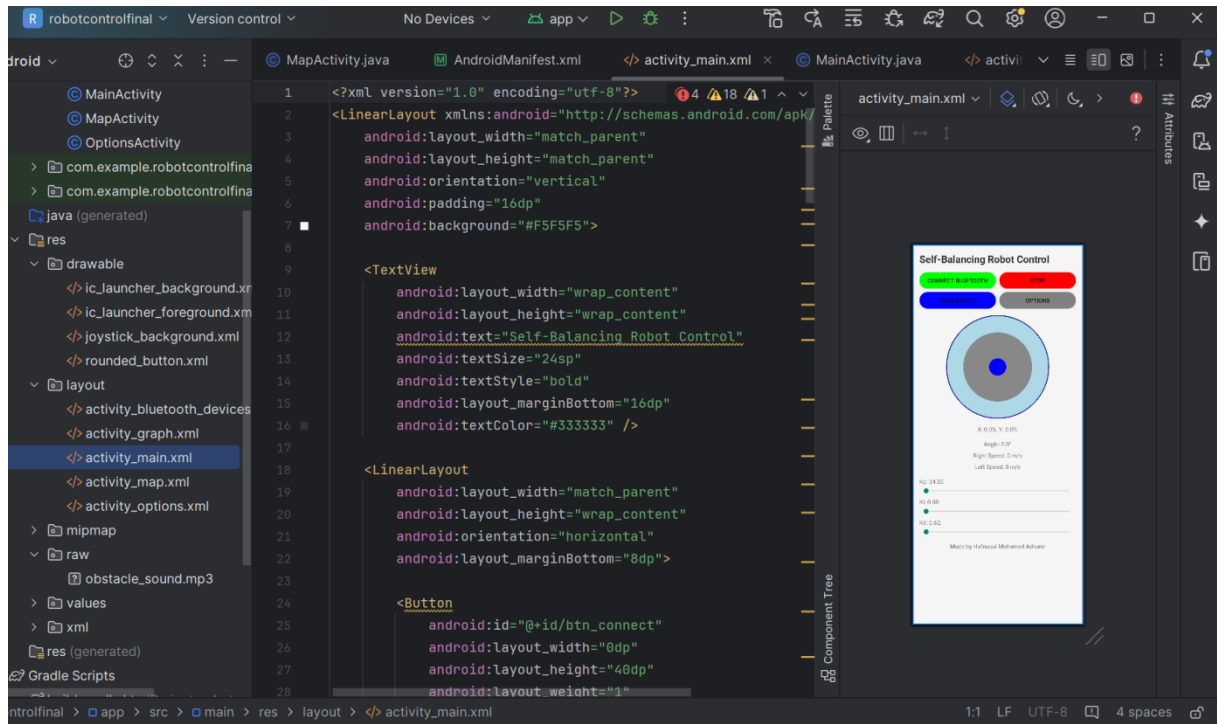


Figure III.17 : Interface de conception de l'application avec Android Studio.

Les barres coulissantes (SeekBar) pour les paramètres PID (K_p , K_i , K_d) permettent un réglage dynamique des coefficients de contrôle. Chaque modification est transmise au robot via la méthode correspondante dans MainViewModel (par exemple updateKp)

- ✓ Avantages : Cette fonctionnalité améliore la sécurité, un aspect clé pour éviter d'endommager le robot ou son environnement en cas de perte de contrôle, et la vitesse affichée permet une vérification rapide d'un arrêt valide.

Le bouton btnGraph ouvre une activité de visualisation graphique, vraisemblablement basée sur une bibliothèque telle que MPAndroidChart.

- ✓ Avantages : Les graphiques facilitent l'analyse des performances du robot (stabilité, réponse aux perturbations, vitesse), ce qui constitue un outil précieux pour vérifier les hypothèses de cette étude.

Un son est diffusé (via MediaPlayer) pour signaler les obstacles, initialisé avec R.raw.obstacle_sound.

- ✓ Avantage: Cette alerte sonore améliore l'interaction homme-machine et rend l'application plus intuitive et adaptée à une utilisation pratique.

Affichage de la vitesse via l'encodeur :

Les vitesses du moteur gauche et droit sont en m/s, calculées à partir des données de l'encodeur reçues via Bluetooth. Ces données sont traitées dans MainViewModel et affichées

en temps réel dans TextView (`textLeftSpeed`, `textRightSpeed`) en observant LiveData. La conversion des impulsions du codeur en vitesse linéaire dépend de paramètres tels que le diamètre de la roue et le taux d'échantillonnage.

- ✓ Avantages : Cet affichage en temps réel fournit une mesure précise et immédiate des performances du moteur, permettant une analyse approfondie du comportement dynamique du robot et une validation expérimentale des modèles théoriques.

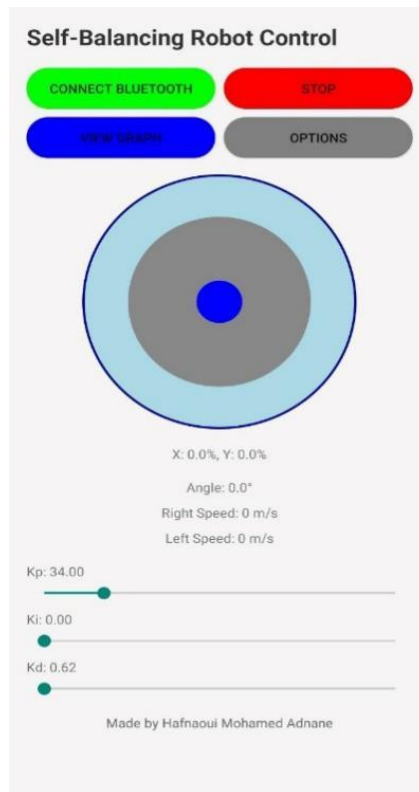


Figure III.18 : L'interface de l'application (Control Panel)

L'application, développée dans Android Studio en utilisant Java, représente une avancée significative dans le contrôle et l'analyse des robots auto-équilibrés. La combinaison d'une interface utilisateur intuitive, d'une communication Bluetooth efficace et d'outils de réglage en temps réel (PID, graphiques) offre une solution polyvalente et puissante. Ces éléments ont facilité les expériences menées dans cet article et ont en même temps ouvert des perspectives d'applications pratiques dans des domaines tels que l'éducation ou la robotique industrielle.

3.6 Utilisation d'un Réseau de Neurones pour la Navigation du Robot

3.6.1 Historique des Réseaux de Neurones :

Les réseaux neuronaux sont bien plus anciens qu'on ne le pense. Bien que l'idée d'une « machine pensante » remonte aux Grecs anciens, cette histoire se concentre sur les événements clés qui ont façonné l'évolution conceptuelle des réseaux neuronaux, une technologie dont la popularité a fluctué au fil des décennies.

En 1943, Warren S. McCulloch et Walter Pitts ont publié un article fondateur intitulé « Le calcul logique des pensées intrinsèques dans l'activité nerveuse ». La recherche vise à comprendre comment le cerveau humain génère des modèles complexes à travers des cellules cérébrales interconnectées, appelées neurones. L'une des principales idées qui a émergé de ce travail a été de comparer des neurones à logique booléenne avec des seuils binaires, c'est-à-dire des valeurs de 0 ou 1, équivalentes à des assertions vrai/faux.

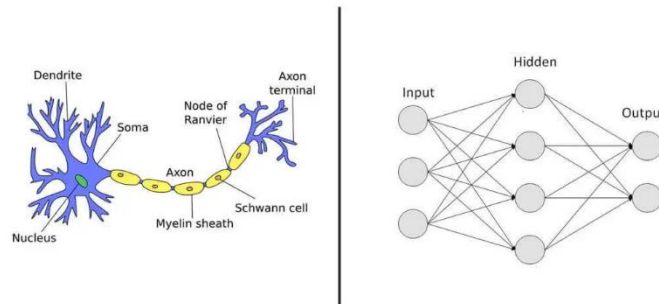


Figure III.19 : Comparaison entre les neurones humains et les réseaux de neurones

En 1958, Frank Rosenblatt a réalisé une avancée majeure avec les perceptrons, qu'il a documentée dans son article « Perceptrons : un modèle probabiliste de stockage et d'organisation de l'information dans le cerveau ». S'appuyant sur les travaux de McCulloch et Pitts, Rosenblatt a introduit le concept de poids dans l'équation, ce qui permet d'ajuster l'importance des entrées. À l'aide d'un ordinateur IBM 704, il a pu former un système capable de distinguer les cartes marquées à gauche et à droite, démontrant ainsi une capacité d'apprentissage rudimentaire.

En 1974, bien que de nombreux chercheurs aient contribué à l'idée de rétropropagation, Paul Werbos a souligné pour la première fois son application spécifique dans les réseaux neuronaux dans sa thèse de doctorat aux États-Unis. La rétropropagation est devenue une

méthode clé pour ajuster les poids du réseau tout en minimisant l'erreur entre les prédictions et les résultats attendus.

Finalement, en 1989, Yann LeCun publie un article montrant comment les contraintes de rétropropagation, combinées à son intégration avec les architectures de réseaux neuronaux, pourraient être utilisées pour entraîner des algorithmes complexes. Les résultats de la recherche ont permis de développer un réseau neuronal capable de reconnaître les chiffres manuscrits dans les codes postaux du service postal américain, marquant ainsi une étape importante dans l'application pratique des réseaux neuronaux.

Ces étapes historiques ont ouvert la voie à l'utilisation des réseaux neuronaux dans des applications modernes, comme celle développée dans ce projet pour la navigation autonome d'un robot auto-équilibré.

3.6.2 Définition d'un Réseau de Neurones :

Un réseau neuronal artificiel (RNA) est un modèle informatique inspiré du fonctionnement du cerveau humain, conçu pour apprendre à partir de données et résoudre des problèmes complexes tels que la classification, la prédiction ou la reconnaissance de formes. Il se compose de plusieurs couches de neurones interconnectés : couche d'entrée, couche cachée et couche de sortie. Chaque neurone reçoit une entrée, la pondère à l'aide de poids ajustables, applique une fonction d'activation (telle que ReLU ou softmax) et transmet le résultat à la couche suivante. La formation du réseau repose sur l'ajustement des poids via un processus d'optimisation, généralement via la rétropropagation des erreurs, pour minimiser une fonction de perte, telle que l'entropie croisée pour les problèmes de classification.

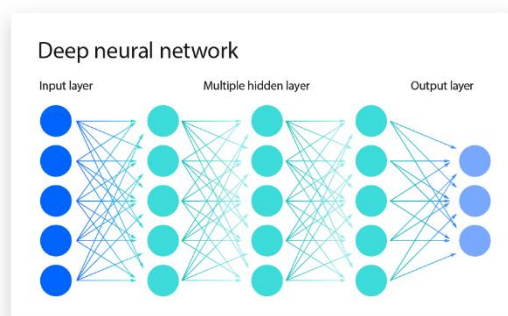


Figure III.20 : Architecture d'un réseau de neurones profond avec couche d'entrée, couches cachées multiples et couche de sortie.

Dans le cadre de ce projet, un RNA a été utilisé pour permettre au robot auto-équilibré de naviguer et d'éviter les obstacles en fonction des distances mesurées par un capteur

ultrasonique HC-SR04. Le réseau a été entraîné pour prédire quatre actions possibles : arrêt (0-10 cm), recul (10-20 cm), tourner à gauche et tourner à droite (20-40 cm).

3.6.3 Étapes de Développement du Modèle d'Intelligence Artificielle :

Le développement du modèle d'IA a suivi plusieurs étapes clés, réalisées en utilisant Google Colab et déployées sur un microcontrôleur ESP32 Dev Kit V1.

- **Collecte et génération de données** : Initialement, les données réelles ont été collectées à l'aide du capteur HC-SR04 (broche TRIG = 12, ECHO = 14) à un débit en bauds série de 115 200. Cependant, pour obtenir un ensemble de données équilibré, 100 échantillons synthétiques ont été générés pour chaque classe, pour un total de 400 échantillons. Un bruit gaussien de $\pm 0,5$ cm a été ajouté pour simuler les variations du capteur et garantir la robustesse aux conditions réelles.
- **Préparation des données** : Normalisez les distances à l'aide de « StandardScaler » pour standardiser l'entrée du modèle. L'ensemble de données est divisé en 80 % de formation et 20 % de tests et stratifié pour maintenir l'équilibre des classes.
- **Conception et formation du modèle** : Un réseau neuronal à trois couches a été construit : une couche d'entrée (1 neurone pour la distance), deux couches cachées (48 neurones chacune avec normalisation par lots et 20 % d'abandon) et une couche de sortie (4 neurones avec activation softmax avec 4 actions). Le modèle a été formé pendant 200 époques en utilisant la fonction de perte d'entropie croisée et l'optimiseur Adam avec un taux d'apprentissage de 0,0001. Des techniques telles que l'arrêt précoce et la réduction du taux d'apprentissage ont été utilisées pour éviter le surapprentissage.

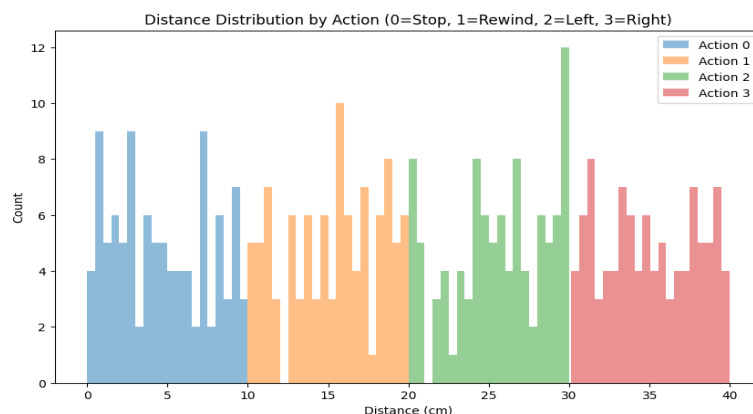


Figure III.21 : Histogramme de la distribution des distances par action (0=Arrêt, 1=Recul, 2=Gauche, 3=Droite)

- **Évaluer et ajuster** :

les premières itérations ont montré des performances limitées (la précision du test initial était de 60 %) en raison d'un déséquilibre de classe et de plages de distance qui se chevauchent. Pour analyser les erreurs de classification, une matrice de confusion a été utilisée. Une **matrice de confusion** est un outil d'évaluation qui représente graphiquement les performances d'un modèle de classification en comparant les valeurs prédites aux valeurs réelles. Il est structuré comme une matrice carrée où les lignes correspondent aux catégories réelles et les colonnes aux catégories prédites. Les éléments sur la diagonale principale représentent des prédictions correctes, tandis que les éléments hors diagonale représentent des classifications erronées (par exemple, une action « à gauche » est prédite comme « à droite »). Dans notre cas, l'analyse initiale de la matrice de confusion a révélé des confusions fréquentes entre les actions « gauche » et « droite » en raison de leur chevauchement initial (15-30 cm). Après plusieurs ajustements, notamment la réassignation des plages (arrêt : 0-10 cm, recul : 10-20 cm, gauche, droite : 30-40 cm), la matrice de confusion finale a montré une diagonale parfaite, confirmant une précision de 100 %.

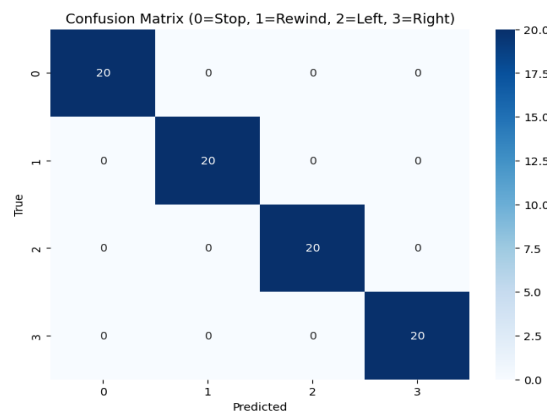


Figure III.22 : Matrice de confusion finale illustrant le précision (0=Arrêt, 1=Recul, 2=Gauche, 3=Droite).

- **Déploiement sur ESP32** : Le modèle a été converti en format TensorFlow Lite ('robot_nav.tflite') et transformé en un tableau C ('robot_nav.h') à l'aide de la commande 'xxd'. Il a été intégré dans un code Arduino utilisant la bibliothèque 'EloquentTinyML', avec un espace d'arène de tenseur de 8 Ko pour éviter les plantages mémoire observés précédemment.
- **Résultats** :

Après l'ajustement final, le modèle a atteint une précision de test de 100 % et une perte de test de 0,1407, dépassant les objectifs fixés (précision > 90 %, perte < 0,2). La précision de l'entraînement est de 96,56 % et la perte est de 0,1555, ce qui indique de bonnes

performances de généralisation et aucun surajustement. Les courbes de perte et de précision (générées dans Colab) montrent une convergence constante, avec une perte en diminution constante et une précision en augmentation constante au fil du temps. La matrice de confusion finale générée après la formation a confirmé qu'il n'y avait aucune erreur de classification et que toutes les prédictions pour chaque classe étaient correctes. Sur ESP32, le modèle offre une navigation fiable : arrêt à 0-10 cm, marche arrière à 10-20 cm, virage à gauche et virage à droite à 30-40 cm et conditions supplémentaires pour éviter les obstacles lors du stationnement.

```

Training Accuracy: 96.56%
Training Loss: 0.1555
Test Accuracy: 100.00%
Test Loss: 0.1407

```

Figure III.23 : Résultats de la précision et la perte

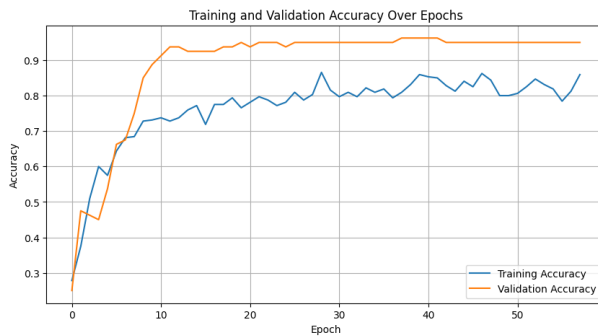


Figure III.24

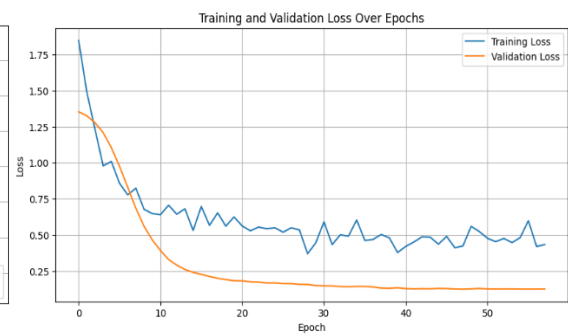


Figure III.25

Figure III.24 : Évolution de la précision d'entraînement et de validation au fil des époques.

Figure III.25 : Évolution de la perte d'entraînement et de validation au fil des époques.

3.7 Partie Mécanique

3.7.1 Conception :

 **SolidWorks** est un logiciel de conception assistée par ordinateur (CAO) et d'ingénierie assistée par ordinateur (IAO) 3D développé par Dassault Systèmes. Il est largement utilisé dans des secteurs tels que l'ingénierie mécanique, la conception de produits et la fabrication, permettant de créer des modèles 3D détaillés, des assemblages et des dessins 2D. SolidWorks est connu pour son interface conviviale et ses fonctions puissantes, prenant en charge la conception paramétrique, la simulation et la gestion des données produit, simplifiant le processus de la conception à la fabrication.



Figure III.26 : SolidWorks

SolidWorks fournit un ensemble puissant de fonctionnalités pour créer des modèles 3D de robots auto-équilibrés, telles que base-bossage, cercle et d'autres outils associés, et leur application dans la modélisation d'un tel robot :

Quelques fonctions de SolidWorks utilisées :

- **Base-bossage(Bossage de base)** : Cette fonction crée une géométrie solide de base en extrudant des esquisses 2D.
- **Cercle** : L'outil d'esquisse **Cercle** est essentiel pour créer des pièces telles que des roues .
- **Extrusion Cut** : Cette fonction supprime de la matière et peut être utilisée pour créer des trous ou des encoches.
- **Congé** : Arrondir les bords pour améliorer la résistance ou l'esthétique. Sur le châssis du robot.
- **Motif circulaire** : pour les robots dotés de plusieurs roues ou de pièces symétriques, cette fonctionnalité répète une caractéristique (comme un trou ou une roue) autour d'un axe central.
- **Assemblage** : Une fois les pièces modélisées (châssis, roues, moteurs), utilisez le mode Assemblage pour les assembler.
- **Équations globales (Paramètres globaux)** : pour les conceptions paramétriques, utilisez des équations globales pour lier les dimensions.

Ces capacités, combinées à l'approche paramétrique de SolidWorks, permettent la création de modèles précis et adaptables tout en facilitant les itérations pour optimiser l'équilibre et la fonctionnalité du robot.

3.7.2 Modèle 3D du robot à auto-équilibrage :

3.7.2.1 Châssis :

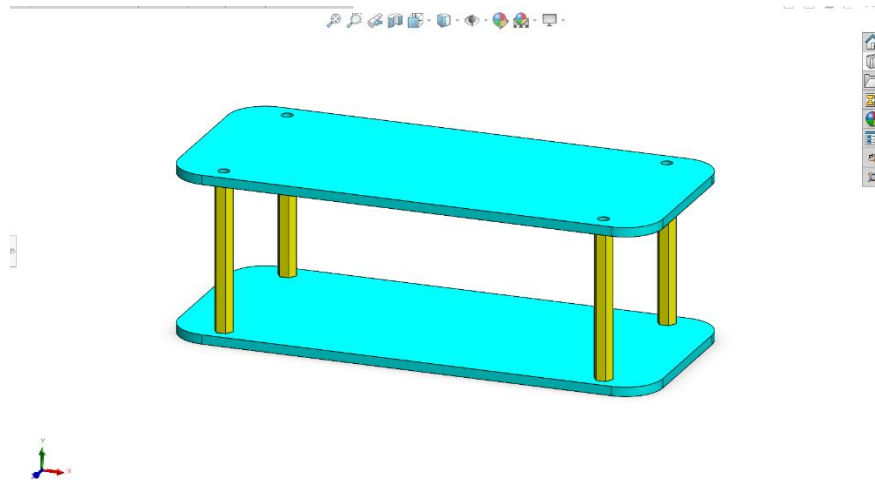


Figure III.27 : Modèle 3D du châssis

3.7.2.2 Roues :

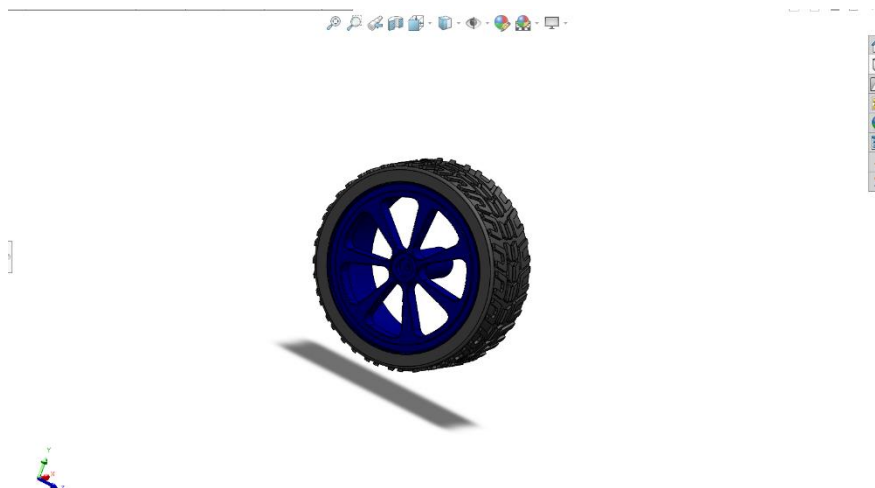


Figure III.28 : Modèle 3D des roues

3.7.2.3 Supports :

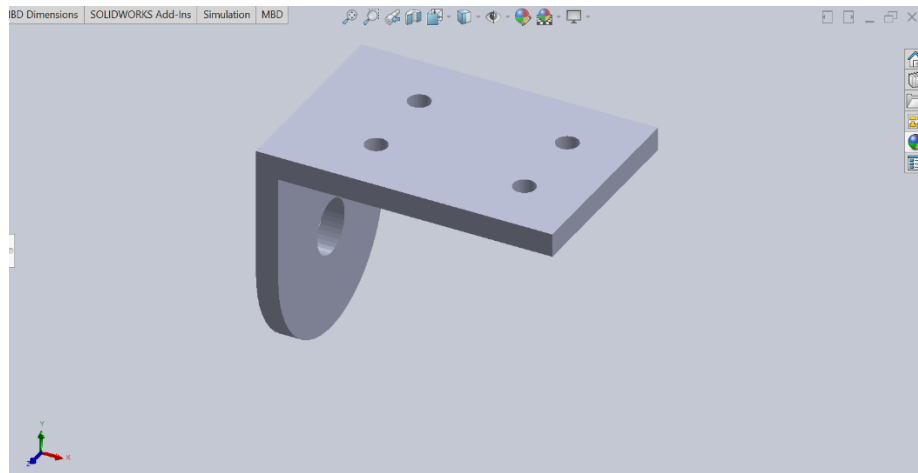


Figure III.29 : Modèle 3D des supports

3.7.2.4 Robot complet :

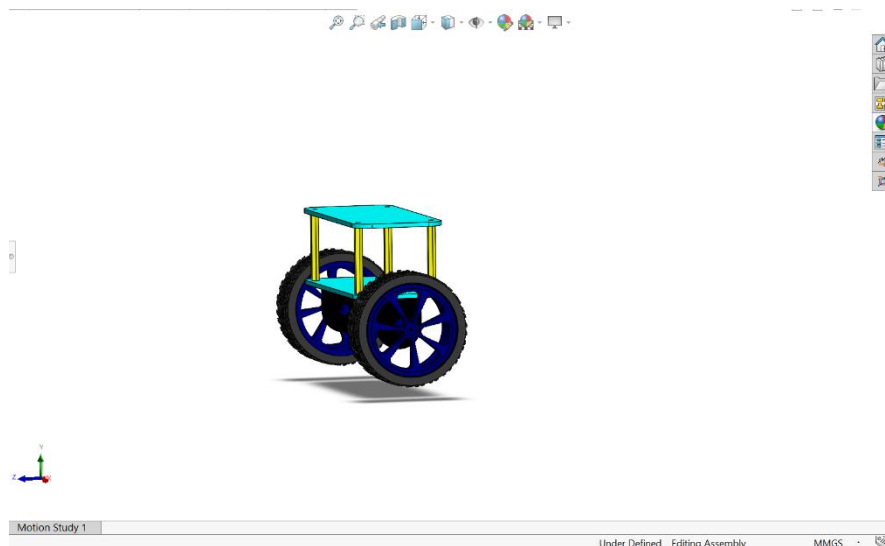


Figure III.30 : Modèle 3D complet du robot à auto-équilibrage

3.8 Assemblage :

Les plaques acryliques ont été assemblées à l'aide de vis hexagonales M3 et M6, puis le circuit imprimé a été fixé. Les composants suivants ont ensuite été placés sur le circuit imprimé : MPU6050, TB6612FNG, ESP32 et LM2596. Les câbles des moteurs ont été connectés, puis les moteurs, leurs supports et leurs pneus, ainsi que le boîtier de batterie, ont été installés.

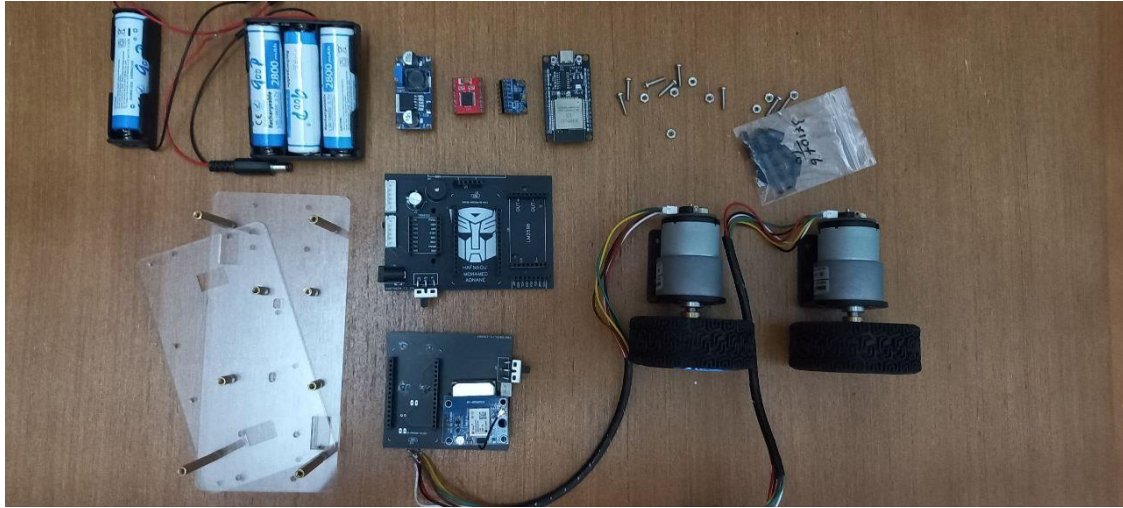


Figure III.31 : Tous les composants et pièces avant assemblage

Le PCB est soudé à l'aide de connecteurs femelles, ce qui évite de devoir souder définitivement les composants, facilitant ainsi leur remplacement si l'un d'entre eux est endommagé.

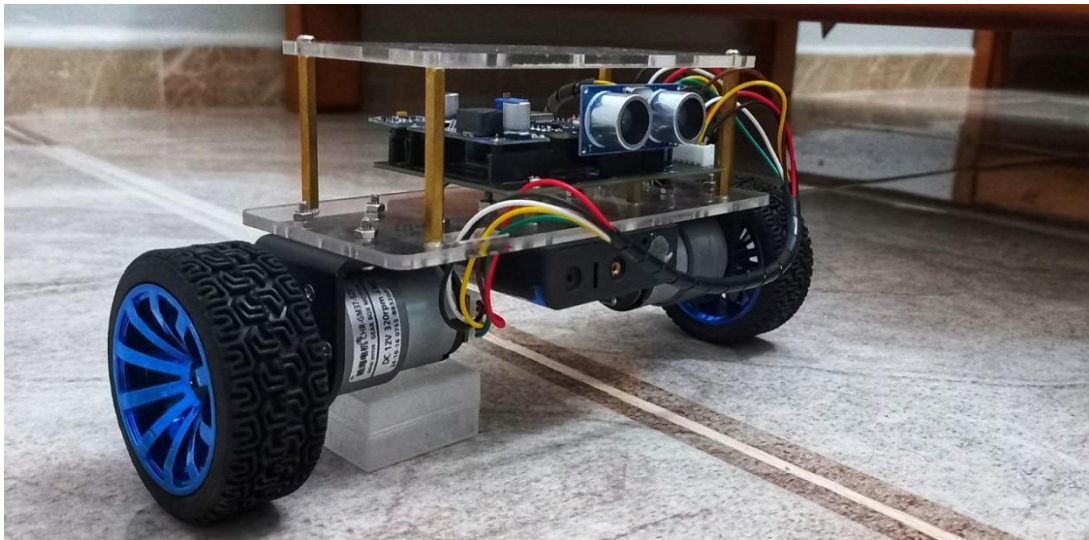


Figure III.32 : Robot assemblé

3.9 Conclusion

Le développement du robot auto-équilibré a nécessité une intégration rigoureuse des choix technologiques en matière de logiciels, d'électricité, de mécanique et d'IA, tout en relevant divers défis pour garantir des performances optimales.

Choix techniques :

- Logiciel : L'IDE Arduino écrit en C++ a été choisi pour programmer l'ESP32. MicroPython offre une complexité accrue, mais une vitesse et une précision de contrôle supérieures. Des algorithmes tels que le filtre de Kalman et trois boucles PID (angle, vitesse, direction) assurent l'équilibre et la navigation. L'application mobile Java développée avec Android Studio prend en charge le contrôle Bluetooth, le réglage PID en temps réel et le suivi GPS.
- Électronique : L'ESP32 gère les capteurs, les moteurs et les communications (Wi-Fi/Bluetooth). Le MPU6050 mesure l'inclinaison, le HC-SR04 détecte les obstacles et le TB6612FNG contrôle les moteurs via des encodeurs. Une batterie 18650, un régulateur LM2596 et des condensateurs (0,1 μ F céramique, 10 μ F et 100 μ F électrolytiques) stabilisent l'alimentation.
- Mécanique : SolidWorks a été utilisé pour concevoir un châssis robuste et des roues optimisées.
- IA : Un réseau neuronal à trois couches a été entraîné sur Google Colab et déployé sur ESP32 via TensorFlow Lite, permettant une navigation autonome avec une précision de 100 %.

Défis rencontrés :

- Équilibrage : Ajustement précis du filtre de Kalman pour compenser le bruit du capteur.
- Ressources : Optimisation du code pour la RAM limitée de l'ESP32.
- Communications : Gestion asynchrone Bluetooth/Wi-Fi complexe.
- IA : Conditionnement des données pour corriger les erreurs de classification initiales.
- Conception : Vérification rigoureuse du circuit imprimé et alignement précis des composants mécaniques.
- Puissance : Stabilisation de la tension malgré les fluctuations du moteur.

Chapitre IV

Théorie de Contrôle

4.1 Modélisation, analyse et contrôle d'un robot auto-équilibré

Ce chapitre explore les bases des algorithmes de contrôle PID (Proportional Integral Derivative) et examine différentes méthodes de réglage. Le contrôleur PID est l'un des algorithmes de contrôle les plus couramment utilisés dans les applications industrielles et d'ingénierie, offrant un contrôle précis et stable des systèmes et des processus. Il fonctionne en utilisant trois composants clés : proportion, intégrale et dérivée. Chacun de ces composants fournit un type différent d'opération de contrôle qui contribue collectivement au résultat global du contrôle. Afin d'améliorer les performances du contrôleur PID et de réduire les erreurs, une méthode de réglage est adoptée. Trois principales méthodes de réglage sont abordées : le réglage manuel et le réglage automatique.

De plus, ce chapitre intègre une simulation à l'aide d'un modèle 3D du robot pour démontrer l'utilisation de contrôleurs PID dans des environnements dynamiques et justifie l'utilisation d'un filtre de Kalman. La simulation met en évidence la manière dont les transformations rigides et autres propriétés cinématiques du robot peuvent être intégrées dans le processus de contrôle, offrant ainsi une visualisation réaliste du comportement du système dans diverses conditions.

4.2 Système en boucle fermée

Les systèmes en boucle fermée utilisent des mécanismes de rétroaction pour ajuster les entrées en fonction des sorties afin d'obtenir une sortie souhaitée stable. Ils sont généralement constitués de capteurs, de contrôleurs et d'actionneurs. Ces systèmes sont largement utilisés dans divers domaines et offrent une précision, une stabilité et une robustesse supérieures à celles des systèmes en boucle ouverte. Les systèmes en boucle fermée sont capables de s'adapter aux changements de l'environnement ou des conditions de fonctionnement du système.

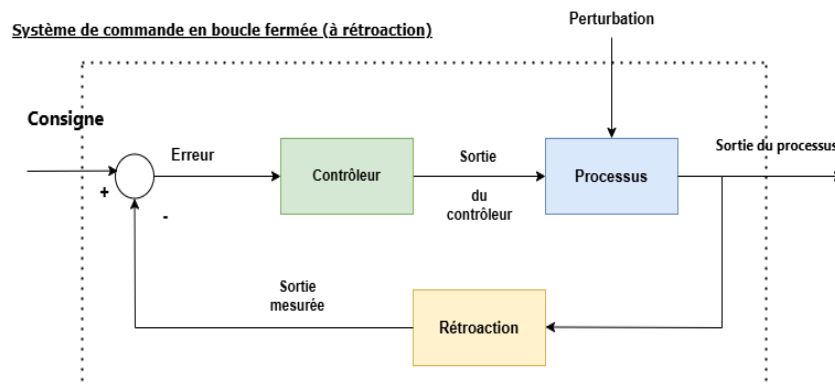


Figure IV.1 : Schéma d'un système en boucle fermée avec capteurs, contrôleur et actionneurs.

4.3 Aperçu du contrôleur PID

Les contrôleurs PID (Proportionnel-Intégral-Dérivé) sont un outil essentiel dans l'automatisation industrielle et le contrôle des processus. Il fonctionne en recevant un retour appelé variable de processus (PV) et en le comparant à une valeur cible prédéfinie appelée point de consigne (SP). Sur la base de cette comparaison, « l'erreur » est calculée. Le rôle d'un contrôleur PID est de produire une réponse sous la forme d'un signal de sortie qui dépend de trois composantes : la proportion de l'erreur actuelle, l'intégrale de l'erreur accumulée dans le temps et la dérivée reflétant le taux de variation de l'erreur.

Ce signal de commande est ensuite envoyé via le pilote de moteur à l'actionneur (ici un moteur à courant continu). Ce dernier a pour fonction d'amplifier le signal et de le convertir en une forme d'énergie électrique capable de faire fonctionner le moteur. La boucle de contrôle ajuste en continu le signal de contrôle jusqu'à ce que la différence entre la variable mesurée (PV) et le point de consigne (SP) devienne nulle, garantissant ainsi que le système atteigne l'état souhaité.

Contrairement aux contrôleurs tout ou rien (ON/OFF), qui activent ou désactivent simplement les appareils lorsque certaines conditions sont remplies, les contrôleurs PID permettent une gestion plus granulaire et stable du système. Cependant, leur efficacité peut varier en fonction de plusieurs paramètres, notamment les réglages du contrôleur, la vitesse de réaction des actionneurs et la complexité du procédé à réguler.

4.4 Algorithme du contrôleur PID

Dans un contrôleur PID parallèle typique, le signal de commande (U) est une combinaison linéaire de termes proportionnels (P), intégraux (I) et différentiels (D), chacun multiplié par ses gains respectifs K_p , K_i et K_d . L'erreur de contrôle (E) correspond à la différence entre le point de consigne (SP) et la variable de processus mesurée (Y).

Mathématiquement, le signal de commande peut être exprimé comme suit :

$$U = K_p \cdot P + K_i \cdot I + K_d \cdot D$$

Où :

$$U = K_p \cdot e(t) + K_i \cdot \int e(t) dt + K_d \cdot \frac{de(t)}{dt}$$

- Le terme proportionnel (P) agit directement sur l'erreur actuelle, apportant une correction proportionnelle à son ampleur.

- Le terme intégral (I) accumule les erreurs passées pour éliminer le biais résiduel à l'état stable.
- Le terme dérivé (D) prédit les changements futurs en tenant compte de la rapidité avec laquelle l'erreur évolue, augmentant ainsi la vitesse et limitant les dépassements.

Le gain de K_p , K_i , et K_d joue un rôle essentiel dans le comportement des contrôleurs. Ils ajustent respectivement l'importance des contributions proportionnelles, intégrales et dérivées. En modifiant ces paramètres, la réponse du système peut être affinée pour atteindre les performances souhaitées.

4.5 Influence des actions proportionnelle, intégrale et dérivée

4.5.1 Contrôleur proportionnel (P) :

Le contrôleur proportionnel (P) est la forme la plus basique de contrôle PID. Il génère une sortie proportionnelle à l'erreur entre le point de consigne (SP) et la variable mesurée (Y). Son équation est donnée par :

$$U(t) = K_p \cdot e(t)$$

Ici, $U(t)$ représente la commande appliquée au temps t , K_p est le gain proportionnel et $e(t)$ est l'erreur instantanée.

L'action de ce type de contrôleur est proportionnelle à l'erreur : plus la différence entre le point de consigne et la valeur mesurée est grande, plus la commande est forte. Cette approche garantit une réponse rapide, mais peut présenter certaines limites.

L'efficacité du contrôleur P dépend en grande partie de la valeur choisie de K_p . Un gain élevé accélère la réponse et réduit les erreurs en régime permanent, mais risque de provoquer une instabilité du système, des oscillations excessives ou un dépassement. En revanche, un faible gain stabilise le système mais entraîne une réponse lente et d'importantes erreurs résiduelles.

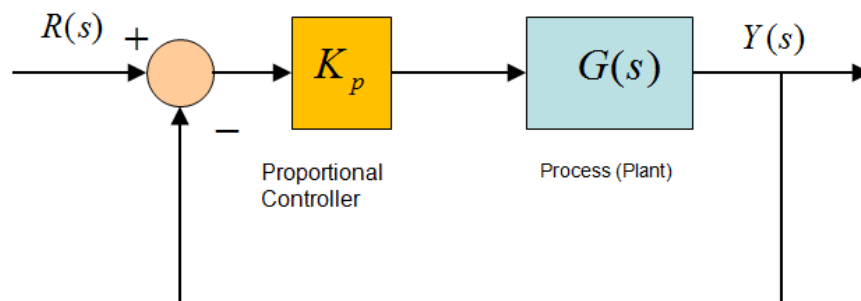


Figure IV.2 : Réponse d'un contrôleur proportionnel (P) à une entrée de consigne.

4.5.2 Contrôleur proportionnel intégral (PI) :

Pour éliminer l'erreur résiduelle en régime permanent, un terme intégral est introduit dans le contrôleur, créant ainsi un contrôleur proportionnel-intégral (PI). Ce terme d'intégration fonctionne en accumulant l'erreur au fil du temps, ce qui équivaut à intégrer le signal d'erreur. Cette accumulation permet une réduction progressive de l'erreur en régime permanent. En combinant ce terme intégral avec un contrôleur proportionnel (P), un contrôleur PI offre un contrôle plus précis et plus stable.

Le signal de commande (U) du contrôleur PI s'exprime comme suit :

$$U = K_p \cdot e(t) + K_i \cdot \int e(t) dt$$

Ici, K_p représente le gain proportionnel, K_i est le gain intégral, et $e(t)$ représente l'erreur instantanée au temps t .

La fonction du terme intégral est d'accumuler l'erreur sur une période donnée, produisant ainsi une correction proportionnelle à l'erreur totale accumulée. Cette méthode élimine non seulement l'erreur résiduelle en régime permanent, mais améliore également la précision et la stabilité globale du système.

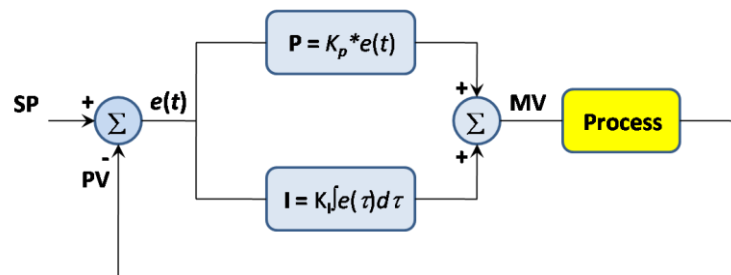


Figure IV.3 : Réponse d'un contrôleur proportionnel-intégral (PI) à une entrée de consigne.

4.5.3 Contrôleur proportionnel-intégral-dérivé (PID) :

En ajoutant des termes dérivés au contrôleur, on obtient un contrôleur Proportional-Integral-Derivative (PID) qui améliore les performances en prédisant le comportement des erreurs futures. Le terme dérivé analyse la rapidité avec laquelle l'erreur change et génère une correction proportionnelle à ce taux de changement. Cette anticipation permet au contrôleur de réagir plus rapidement et avec plus de précision aux fluctuations de la variable du procédé, tout en réduisant les dépassements et en améliorant la stabilité du système.

Le signal de commande (U) du contrôleur PID est exprimé comme suit :

$$U = K_p \cdot e(t) + K_i \cdot \int e(t) dt + K_d \cdot \frac{de(t)}{dt}$$

Dans cette équation :

- K_p est le gain proportionnel, qui ajuste la taille de l'erreur actuelle.

- K_i est le gain intégral, qui corrige l'erreur accumulée au fil du temps.
- K_d est le gain dérivé, qui réagit à la vitesse de variation de l'erreur.
- $e(t)$ représente l'erreur instantanée à l'instant t .
- $\frac{de(t)}{dt}$ désigne la dérivée de l'erreur, c'est-à-dire son taux de variation.

Ce modèle combine les trois termes pour offrir une réponse optimale, capable de gérer à la fois les erreurs actuelles, passées et futures.

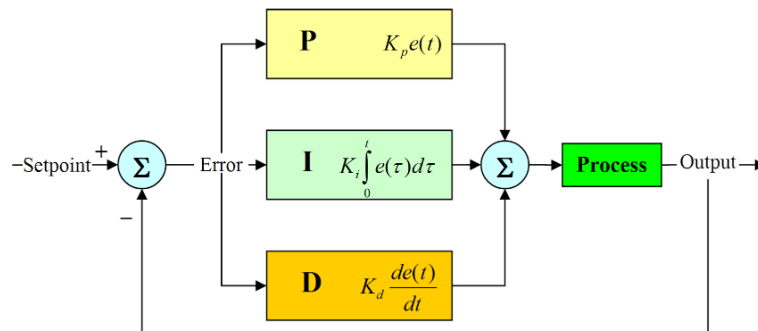


Figure IV.4 : Réponse d'un contrôleur proportionnel-intégral-dérivé (PID) à une entrée de consigne.

Le tableau suivant fournit des lignes directrices générales concernant les effets des coefficients de chaque composante du régulateur PID :

Paramètre	Temps de montée	Dépassement	Temps d'établissement	Erreur en régime permanent	Stabilité
K_p	Diminue	Augmente	Changement faible	Diminue	Diminue
K_i	Diminue	Augmente	Augmente	Élimine	Diminue
K_d	Changements mineurs	Diminue	Diminue	Aucun effet	Améliore si K_d est faible

Tableau IV.1 : Effets des coefficients du régulateur PID sur la réponse du système

Pour optimiser les paramètres PID (Proportionnel-Intégral-Dérivé) du robot auto-équilibré, on utilise une application intuitive avec des curseurs dédiés pour ajuster les valeurs de K_p , K_i et K_d . Cette approche permet de modifier ces paramètres en fonction des besoins spécifiques pour stabiliser le robot. Grâce à l'interface de l'application, il est facile d'ajuster dynamiquement les curseurs tout en observant l'impact sur la stabilité du robot en temps réel.

4.6 Pourquoi un contrôleur PID est nécessaire

Les robots auto-équilibrés sont modélisés à partir de pendules inversés, qui sont intrinsèquement instables sans contrôle. Le pendule inversé est placé sur un support mobile avec son centre de masse au-dessus du point de pivot, il a donc naturellement tendance à s'incliner sous l'influence de la gravité. Sans intervention active, l'angle d'inclinaison θ divergerait rapidement et l'équilibre ne pourrait pas être maintenu. Un contrôleur PID (proportionnel-intégral-dérivé) est nécessaire pour appliquer une force corrective au chariot, en ajustant l'erreur angulaire, son intégrale (pour éliminer les erreurs statiques) et sa dérivée (pour amortir les oscillations). Le PID a été choisi pour sa simplicité, son efficacité dans les systèmes linéaires et sa capacité à stabiliser le système en ajustant dynamiquement la force en fonction des changements de θ .

4.6.1 Démonstration de l'instabilité du système :

Les simulations de modélisation dynamique en boucle ouverte montrent que le système est instable sans contrôle. En utilisant les paramètres physiques du robot (masse corporelle $m = 1.0$ kg, masse de la roue $M = 0.5$ kg, distance du centre de masse $L = 0.05$ m, gravité $g = 9.81$ m/s², moment d'inertie), la matrice d'état et la fonction de transfert sont dérivées pour l'analyse dynamique. L'analyse des pôles montre systématiquement des pôles avec des parties réelles positives, indiquant une instabilité exponentielle. De plus, les simulations de réponse par étapes et par impulsions montrent que l'angle θ et la position du chariot x_c divergent rapidement, confirmant que le système ne peut pas maintenir l'équilibre sans un mécanisme de contrôle actif.

4.7 Pourquoi utiliser un filtre de Kalman

Dans un robot auto-équilibré, l'angle d'inclinaison θ est mesuré à l'aide d'un capteur tel que le MPU6050, qui combine un accéléromètre et un gyroscope. Cependant, les données de l'accéléromètre sont bruyantes en raison des vibrations et de l'accélération non gravitationnelle, tandis que les gyroscopes dérivent en raison des erreurs accumulées. Un filtre de Kalman est utilisé pour fusionner ces mesures complémentaires afin d'estimer l'angle θ avec une plus grande précision. Ce filtre combine une prédiction basée sur le gyroscope de la vitesse angulaire (intégrée dans le temps) avec la correction fournie par l'accéléromètre et pondère chaque mesure en fonction du niveau de bruit. Cela garantit une estimation robuste de θ , ce qui est essentiel pour fournir une entrée fiable au contrôleur PID.

4.8 Modélisation symbolique et analyse de stabilité

Cette simulation a établi un modèle mathématique du robot auto-équilibré, en utilisant des variables symboliques pour représenter les paramètres physiques du robot (masse corporelle, masse des roues, distance du centre de masse, gravité, moment d'inertie). Il définit les équations qui décrivent comment l'angle d'inclinaison (θ) et la position du chariot (x_c) changent en réponse à une force appliquée. Dérivez des expressions générales entre les angles et les forces et évaluez-les avec des valeurs spécifiques ($m = 1.0 \text{ kg}$, $M = 0.5 \text{ kg}$, $L = 0.05 \text{ m}$, $g = 9.81 \text{ m/s}^2$). L'analyse a montré que le système était instable, car certains paramètres (pôles) présentaient une croissance exponentielle incontrôlée des angles.

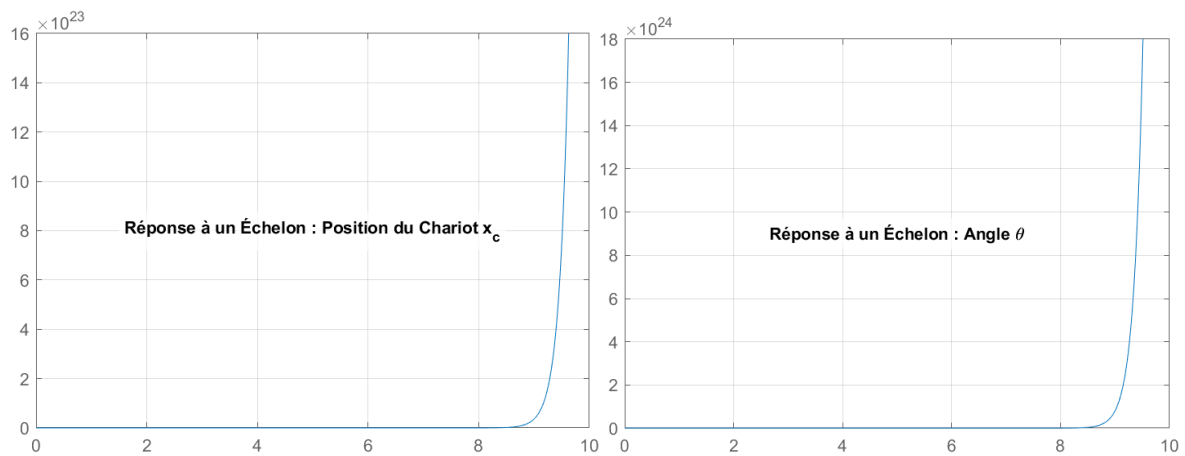


Figure IV.5 : Réponse à un échelon (Position + Angle)

4.8.1 Analyse numérique de l'instabilité en boucle ouverte :

La simulation utilise des valeurs numériques pour modéliser le comportement du robot, décrivant comment l'angle d'inclinaison (θ) et la position du chariot (x_c) changent sous l'influence des forces. Il crée une relation mathématique entre la force appliquée et l'angle, puis examine les caractéristiques du système (pôles et zéros) pour confirmer l'instabilité. Les simulations visuelles montrent que lorsqu'une force constante est appliquée, l'angle θ augmente infiniment, renforçant la nécessité d'un contrôle actif.

4.8.2 Analyse des réponses dynamiques multi-sorties :

étude des réponses à des perturbations et entrées

La simulation modélise le comportement du robot en prenant en compte l'angle d'inclinaison (θ) et la position du chariot (x_c). Il examine la réponse du système aux perturbations

soudaines (telles que les chocs) et aux forces constantes. Les résultats montrent que l'angle et la position sont déviés dans les deux cas, confirmant que le système est instable lorsqu'il n'est pas sous contrôle. Cette analyse met en évidence l'interaction entre l'angle et le mouvement du chariot, ce qui est essentiel pour le contrôle futur.

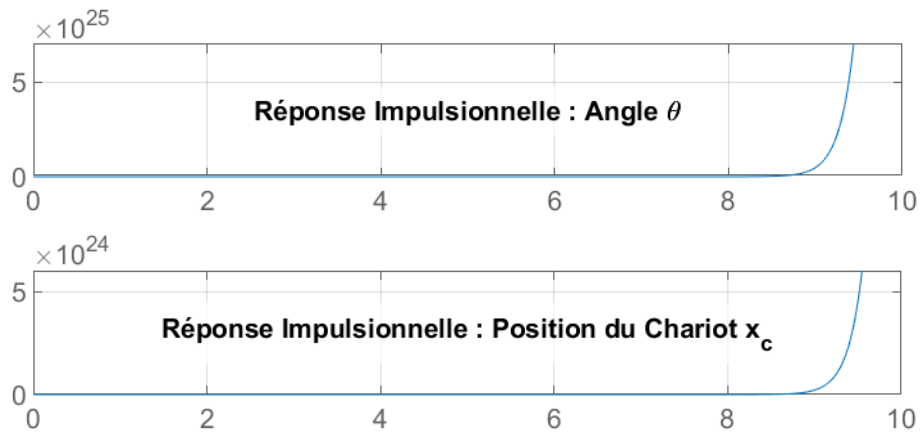


Figure IV.6 : Réponse impulsionnelle du position et angle du système

4.8.3 Estimation de l'angle avec un filtre de Kalman :

Traitement des données capteurs pour une estimation précise :

Cette simulation se concentre sur la mesure de l'angle d'inclinaison (θ) à l'aide d'un capteur combiné accéléromètre et gyroscope. Il simule des données bruyantes, où les accéléromètres produisent des mesures fluctuantes et les gyroscopes présentent des erreurs cumulatives. Une méthode de fusion est utilisée pour combiner ces données afin d'obtenir une estimation précise de θ . Les résultats montrent que les résultats estimés sont très proches de l'angle réel, réduisant ainsi l'influence du bruit et des erreurs.

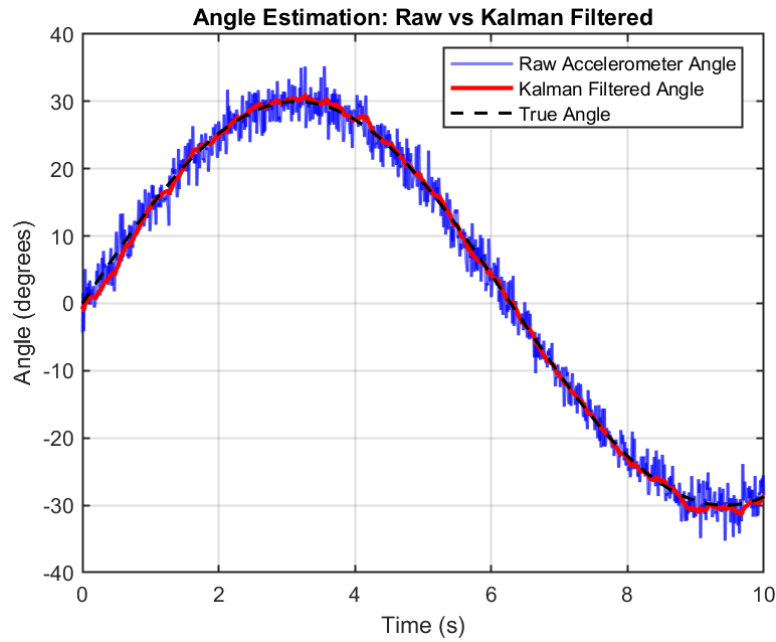


Figure IV.7 : Estimation de l'angle et l'accéléromètre avec le filtre de Kalman

4.8.4 Stabilisation avec un contrôleur PID :

Cette simulation introduit un mécanisme de contrôle pour stabiliser l'angle d'inclinaison (θ) du robot. Elle utilise la relation angle-force établie précédemment et applique un contrôleur qui ajuste la force en fonction de l'erreur d'angle, de son accumulation, et de sa variation rapide. Le contrôleur est conçu automatiquement pour garantir la stabilité. La simulation montre que, lorsqu'une force constante est appliquée, l'angle θ reste proche de zéro, contrairement aux divergences observées sans contrôle. Une vérification confirme que le système est stable avec ce contrôle.

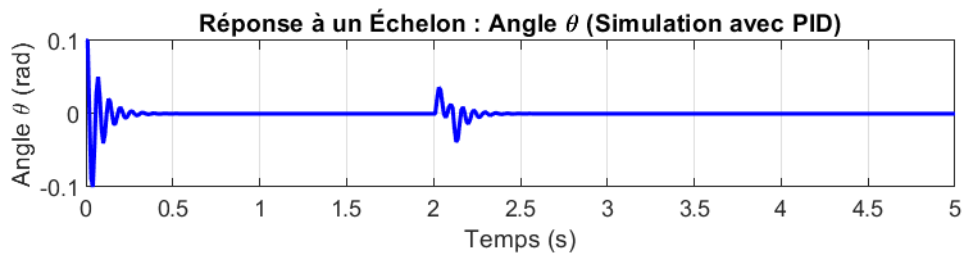


Figure IV.8 : Réponse à l'échelon d'angle en Boucle Fermé

Ces simulations forment une progression logique pour la conception d'un robot auto-équilibré. Les simulations initiales établissent que le système est instable sans contrôle, avec des angles et positions divergeant rapidement, comme démontré par les analyses de stabilité et les réponses dynamiques. La simulation du traitement des données capteurs montre

comment obtenir une estimation précise de l'angle θ malgré le bruit, grâce à une méthode de fusion. Enfin, la simulation de contrôle démontre qu'un mécanisme bien conçu peut stabiliser le système, maintenant l'angle proche de zéro. Les figures proposées (cartes des pôles, réponses à un échelon, diagrammes de Bode, erreurs d'estimation) enrichissent l'analyse, offrant une visualisation claire des concepts théoriques et pratiques. Les prochaines étapes devraient inclure l'intégration de l'estimation d'angle avec le contrôle, l'exploration de modèles non linéaires, et la validation expérimentale avec un prototype.

4.9 Analyse des forces du robot auto-équilibré

La stabilité d'un robot auto-équilibré repose sur l'application de la deuxième loi de Newton et sur l'équilibre des moments. Deux types de forces principales interviennent :

- **La force gravitationnelle** : elle agit vers le bas à partir du centre de gravité et tend à faire basculer le robot.
- **La force d'inertie due à l'accélération des roues** : en réaction (loi de Newton III), cette force pousse le corps du robot dans la direction opposée à l'accélération des roues, générant un couple compensateur.

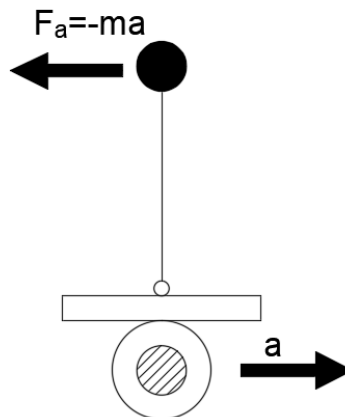


Figure IV.9 : Schéma d'un pendule simple avec force appliquée

4.9.1 Équation d'équilibre des couples :

Pour que le robot reste équilibré, les couples doivent se compenser :

$$m \cdot g \cdot L \cdot \sin(\theta) = m \cdot a \cdot L \cdot \cos(\theta)$$

où :

- m : masse du robot
- g : accélération gravitationnelle ($9,81 \text{ m/s}^2$)
- L : distance entre l'axe des roues et le centre de gravité
- θ : angle d'inclinaison

- a : accélération linéaire du robot (via les roues)

4.9.2 Forces agissant sur le robot auto-équilibré :

Le robot auto-équilibré est assimilé à un pendule inversé monté sur une base mobile. Deux forces principales interviennent lorsque le robot s'incline :

- F_2 : force gravitationnelle tangentielle, agissant pour faire tomber le robot ;
- F_1 : force d'inertie de réaction, engendrée par l'accélération de la base, agissant pour rétablir l'équilibre.

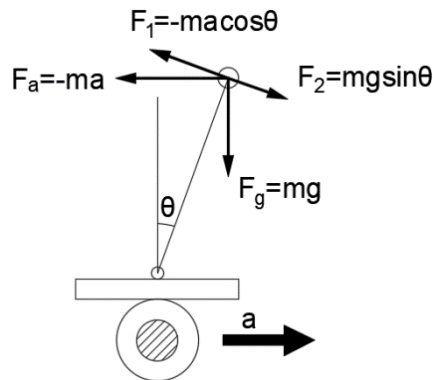


Figure IV.10 : Schéma d'un pendule avec forces composantes et accélération

Ces forces s'expriment par :

$$F_2 = m \cdot g \cdot \sin(\theta), F_1 = -m \cdot a \cdot \cos(\theta)$$

L'équilibre est atteint lorsque la somme des forces tangentielles est nulle :

$$F_1 + F_2 = -m \cdot a \cdot \cos(\theta) + m \cdot g \cdot \sin(\theta) = 0 \Rightarrow a = g \cdot \tan(\theta)$$

Pour les petits angles θ , on utilise les approximations :

$$\sin(\theta) \approx \theta, \cos(\theta) \approx 1, \tan(\theta) \approx \theta \Rightarrow a \approx g \cdot \theta$$

Cela signifie que le robot doit accélérer proportionnellement à son inclinaison pour maintenir son équilibre.

4.9.3 Rôle du moteur à courant continu avec réducteur :

Le moteur génère un couple τ_{moteur} , transformé en accélération angulaire α , puis en accélération linéaire des roues :

$$\tau_{\text{moteur}} = J_{\text{total}} \cdot \alpha$$

Avec :

$$a = r \cdot \alpha \Rightarrow \tau_{\text{moteur}} = \frac{J_{\text{total}}}{r} \cdot a$$

où :

- J_{total} : inertie totale (robot + roues + moteur)
- r : rayon des roues

4.10 Contrôle par régulateur PD

On utilise un **contrôleur proportionnel-dérivé (PD)** pour estimer l'accélération requise a :

$$a = -K_p \cdot \theta - K_d \cdot \dot{\theta}$$

où :

- K_p : gain proportionnel
- K_d : gain dérivé
- $\dot{\theta}$: vitesse angulaire

Le terme proportionnel ramène le robot vers la verticale, tandis que le terme dérivé amortit les oscillations.

4.11 Implémentation du contrôle moteur

Le moteur reçoit la commande suivante :

- Mesure de θ et $\dot{\theta}$ via IMU
- Calcul de a avec le contrôleur PD

Conversion en couple :

$$\tau_{\text{moteur}} = \frac{J_{\text{total}}}{r} \cdot a$$

4.11.1 Exemple numérique:

Soit :

- $\theta = 5^\circ = 0,087 \text{ rad}$
- $\dot{\theta} = 0,1 \text{ rad/s}$
- $K_p = 10, K_d = 2$

Alors :

$$a = -10 \cdot 0,087 - 2 \cdot 0,1 = -1,07 \text{ m/s}^2$$

Si

$$J_{\text{total}} = 0,1 \text{ kg} \cdot \text{m}^2, \quad r = 0,05 \text{ m}$$

alors :

$$\tau_{\text{moteur}} = \frac{0,1}{0,05} \cdot (-1,07) = -2,14 \text{ N} \cdot \text{m}$$

4.12 Importance des engrenages

Un réducteur avec un rapport 1:30 permet :

- d'augmenter le couple ($\times 30$),
 - de réduire la vitesse ($\div 30$),
- rendant le système plus précis et plus puissant à faible vitesse, idéal pour l'équilibrage.

4.13 Problèmes fréquents et solutions

- Oscillations : K_p trop élevé $\rightarrow$ augmenter K_d
- Bruit capteurs : utiliser un filtre passe-bas ou de Kalman
- Erreur statique : ajouter un terme intégral $K_i \rightarrow$ PID

4.14 Résumé

Le contrôleur PD calcule l'accélération requise à partir de θ et $\dot{\theta}$, convertit cette accélération en couple moteur via les lois dynamiques, et applique ce couple via le moteur à courant continu avec réducteur. Les gains K_p et K_d sont essentiels pour assurer une réponse rapide et stable du robot sans sur/sous-oscillations.

4.15 Justification du choix $K_i = 0$ dans la commande

L'architecture de commande utilisée repose volontairement sur un **correcteur Proportionnel-Dérivé (PD)**, en laissant de côté le terme intégral K_i . Cette décision s'appuie sur plusieurs fondements théoriques et pratiques liés à la nature dynamique du robot auto-équilibré.

4.15.1 Réactivité et stabilité immédiate :

Le système à stabiliser est intrinsèquement **instable** et requiert une réaction **rapide** à toute perturbation. Le rôle principal du contrôleur est de maintenir le robot à la verticale en ajustant continuellement l'accélération des roues selon l'angle d'inclinaison mesuré.

L'ajout d'un terme intégral K_i , qui agit sur l'**erreur cumulée dans le temps**, introduit une réponse **plus lente**. Cette lenteur compromet la **stabilité immédiate** indispensable à un système de type pendule inversé.

4.15.2 Inutilité de la correction de l'erreur statique :

Dans le cas d'un robot auto-équilibré, le point de consigne est généralement nul :

$$\theta_{\text{ref}} = 0$$

L'objectif est donc de maintenir un **angle nul**. Grâce à l'action du terme proportionnel $K_p \cdot \theta$ et du terme dérivé $K_d \cdot \dot{\theta}$, l'erreur statique sur l'angle est déjà réduite à un niveau négligeable. Le recours au terme intégral ne contribue pas significativement à la performance, tout en ajoutant un risque de sur-régulation.

4.15.3 Risque de "windup" de l'intégrateur :

Le terme intégral accumule l'erreur sur de longues périodes. Lors de basculements importants ou de perturbations, cette accumulation peut provoquer un **dépassement important** ou des **oscillations** non souhaitées. Ce phénomène est appelé **intégration excessive** ou *integral windup*.

Le traitement de ce problème implique des mécanismes supplémentaires (anti-windup), ce qui alourdit la complexité du système sans gain notable en performance.

4.15.4 Sensibilité accrue au bruit des capteurs :

Le capteur IMU (Inertial Measurement Unit) utilisé pour mesurer l'angle d'inclinaison fournit des données entachées de bruit. Le terme intégral ne fait pas de distinction entre une erreur réelle et une fluctuation due au bruit. Ainsi, le **bruit est intégré**, ce qui génère des dérives et des réponses erronées dans la commande moteur.

4.15.5 Propriétés du modèle dynamique :

Le robot auto-équilibré est modélisé comme un **système à double intégrateur**. Son équation différentielle type est :

$$\ddot{\theta} = \frac{m \cdot g \cdot L \cdot \sin(\theta) - b \cdot \dot{\theta} + a \cdot L}{J}$$

Ce modèle contient déjà deux intégrations naturelles (l'angle étant une double intégrale de l'accélération). L'introduction d'un **troisième intégrateur** via le terme K_i rend le système **trop lent** et **difficile à stabiliser**. Cela compromet la robustesse de la commande.

4.15.6 Bilan technique du choix $K_i = 0$:

Critère	Justification de l'exclusion de K_i
Réactivité	Le terme intégral ralentit la réponse du système
Stabilité	Risque d'oscillations ou d'instabilité
Complexité	Nécessite une gestion du windup
Résistance au bruit	L'intégration amplifie les erreurs des capteurs
Nature du modèle dynamique	Le système possède déjà deux intégrateurs

Tableau IV .2 : Bilan technique du choix $K_i = 0$

L'utilisation d'un contrôleur **PD** (et non PID) se révèle la plus adaptée pour le maintien de l'équilibre d'un robot à deux roues. Le correcteur fournit une réponse rapide, amortie et robuste aux perturbations. L'exclusion du terme intégral simplifie le système, améliore sa stabilité, et renforce sa capacité à réagir efficacement aux déséquilibres sans introduire d'instabilité numérique ou physique.

4.16 Analyse critique des modèles mathématiques pour robots autostabilisants : limites et perspectives d'amélioration

4.16.1 Introduction :

La modélisation mathématique des robots auto-équilibrés, en particulier ceux basés sur la dynamique du pendule inversé, est une étape clé dans le développement de systèmes de contrôle efficaces. Cependant, les modèles classiques utilisés dans la littérature ou l'enseignement sont souvent basés sur des hypothèses idéalisées. L'analyse met en évidence les principales limites de ces approches théoriques et propose des pistes spécifiques pour les enrichir, en s'appuyant sur les avancées récentes en robotique, en identification de systèmes et en contrôle robuste [25].

4.16.2 Principales limitations des modèles théoriques :➤ **Hypothèses physiques trop simplifiées :**

Les modèles actuels ignorent plusieurs aspects physiques fondamentaux :

- Modélisation du point de masse : La répartition réelle des composants (batteries, électronique, châssis) affecte directement l'inertie et la stabilité globales du système [26].

- Ignorer les effets aérodynamiques : Pour les structures plus grandes, la traînée devient une force non négligeable et son modèle est le suivant :

$$F_d = \frac{1}{2} \rho C_d A v^2$$

Elle peut représenter plus de 15 % des forces actives dans certains cas [27].

➤ **Représentation incomplète des interactions mécaniques :**

- **Contact des roues avec le sol :** Les phénomènes de frottement, de glissement ou de roulement sont généralement linéarisés, mais ils dépendent de nombreux facteurs, comme la nature du sol ou la vitesse du robot. La résistance au roulement est généralement modélisée par la formule suivante :

$$F_{rr} = C_{rr} \cdot F_n$$

Cependant, sa dépendance non linéaire à la vitesse a rarement été prise en compte [28].

- **Flexibilité structurelle :** Les éléments mécaniques déformables peuvent induire des vibrations parasites, typiquement entre 5 et 20 Hz, qui peuvent altérer la précision du modèle [29].
- **Absence de composantes électromécaniques :**
- **Dynamique des moteurs :** les modèles simplifiés omettent souvent l'équation complète de fonctionnement des moteurs à courant continu :

$$\tau = K_t \left(\frac{V - K_e \omega}{R} \right) - J_m \dot{\omega}$$

sans considérer la saturation du couple ni la limite de vitesse [30].

- **Alimentation électrique :** les pertes par résistance interne, les temps de réponse des convertisseurs et les effets transitoires sont généralement négligés, alors qu'ils impactent directement la stabilité en boucle fermée [31].

4.16.3 Validation expérimentale et ajustements nécessaires :

➤ **Disparités entre modèles et réalité :**

Des études expérimentales récentes ont montré que les performances atteintes par les modèles purement théoriques sont en moyenne de 62 ± 8 % de celles observées dans les systèmes physiques. Ces différences sont principalement dues à :

1. Dynamique électromécanique sans modélisation ($\approx 30\%$)

2. Incertitude des paramètres physiques ($\approx 25\text{--}30\%$)
3. Perturbations environnementales imprévues ($\approx 20\%$)

➤ **Intégration de paramètres réalistes :**

L'enrichissement du modèle passe par l'intégration de paramètres mesurés expérimentalement :

- L'inertie réelle de l'élément mobile
- Coefficient de frottement spécifique au sol utilisé
- Constantes du moteur issues de fiches techniques ou d'essais au banc

Des outils tels que l'identification des paramètres, y compris les méthodes des moindres carrés récursifs, permettent un réglage en ligne de ces paramètres [32].

➤ **Stratégies de commande et architecture de contrôle :**

Pour compenser les incertitudes et les effets non modélisés, plusieurs méthodes de contrôle avancées peuvent être employées :

- **PID adaptatif** : ajuste dynamiquement les gains en fonction de l'évolution des conditions physiques
- **Contrôle H_∞ robuste** : maximiser la stabilité face aux perturbations externes et aux incertitudes internes [33].
- **Filtrage actif des interférences** : suppression améliorée des signaux parasites causés par la flexibilité ou les changements de charge

4.17 Justification de l'intégration des effets réalistes

Dans la modélisation dynamique des robots auto-équilibrés, l'intégration d'effets réalistes tels que le frottement visqueux, le frottement de Coulomb, le bruit du capteur et les perturbations externes est cruciale pour garantir que la simulation est fidèle aux situations du monde réel. Ces effets aident à mieux représenter les défis rencontrés dans les environnements physiques, où les imperfections des capteurs et les forces externes peuvent affecter considérablement les performances du système. Plus précisément, les modèles de frottement visqueux ($b = 0,2$) et de Coulomb ($F_c = 0,5$) modélisent les pertes d'énergie dues aux interactions mécaniques, tandis que le bruit du capteur ($\sigma_n = 0,005$ rad) simule les

erreurs de mesure typiques des systèmes embarqués. Des perturbations externes, appliquées sous forme d'une force d'impulsion de 1 N entre 5 et 5,1 s, permettent d'évaluer la robustesse du contrôleur face à des événements imprévus. Ces considérations réalistes garantissent que le contrôleur PD conçu peut maintenir la stabilité et les performances du robot dans des scénarios presque réalistes.

4.18 Résultats graphiques

Les figures suivantes illustrent les performances du robot à auto-équilibrage sous l'effet du contrôleur PD et des conditions réalistes décrites.

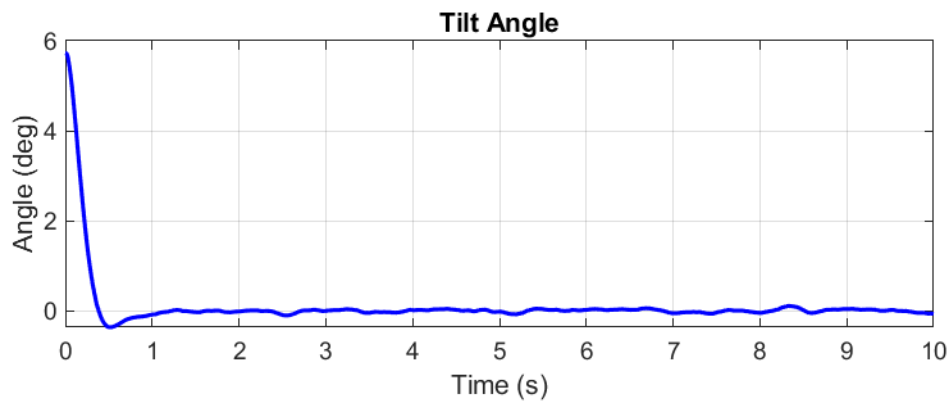


Figure IV.11 : Angle d'inclinaison du robot en fonction du temps

La Figure IV.7 montre l'évolution de l'angle d'inclinaison du robot (en degrés) sur une période de 10 secondes. On observe une convergence rapide vers l'équilibre.

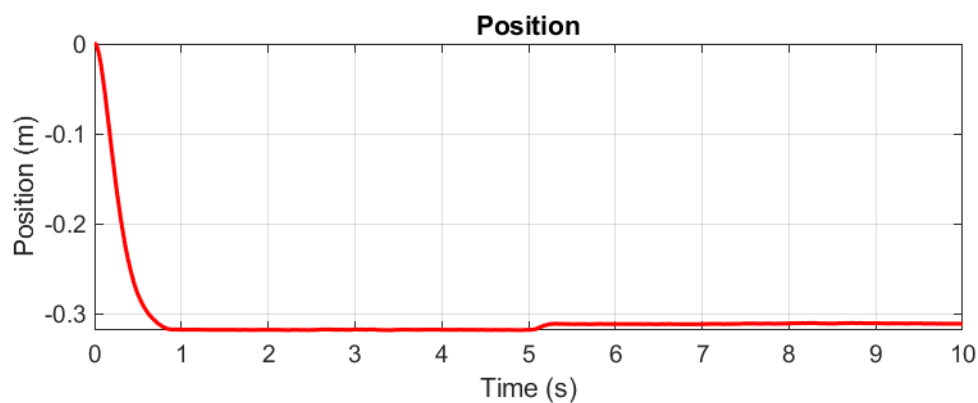


Figure IV.12 : Position du robot en fonction du temps

La Figure IV.8 illustre la position linéaire du robot (en mètres). Le contrôleur PD maintient une position stable, avec des ajustements minimes en réponse à la perturbation externe.

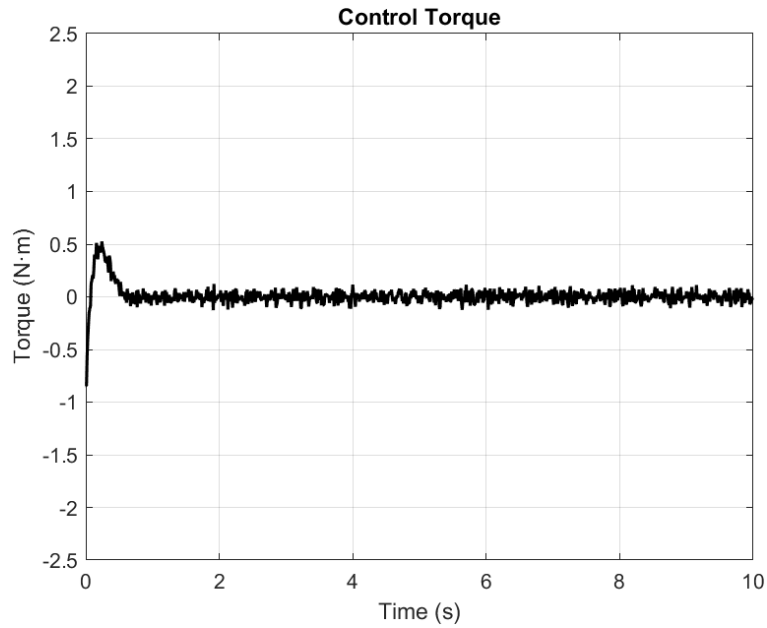


Figure IV.13 : Couple de contrôle appliqué au robot

La Figure IV.9 représente le couple de contrôle (en $\text{N}\cdot\text{m}$) généré par le contrôleur PD. Les valeurs restent dans les limites du couple maximal ($T_{\text{max}} = 2 \text{ N}\cdot\text{m}$), démontrant l'efficacité de la saturation du contrôle.

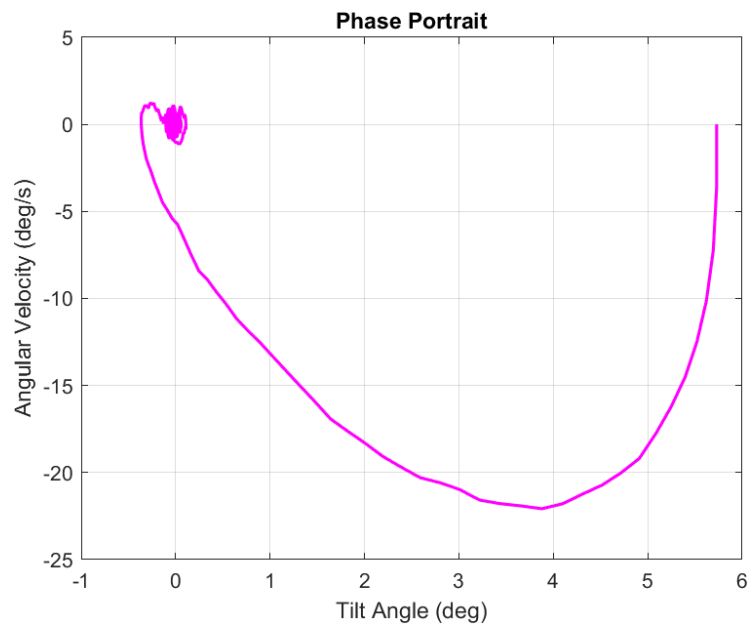


Figure IV.14 : Portrait de phase du robot

La Figure IV.10 affiche le portrait de phase, montrant la relation entre l'angle d'inclinaison et la vitesse angulaire. La trajectoire converge vers l'origine, indiquant une stabilisation du système.

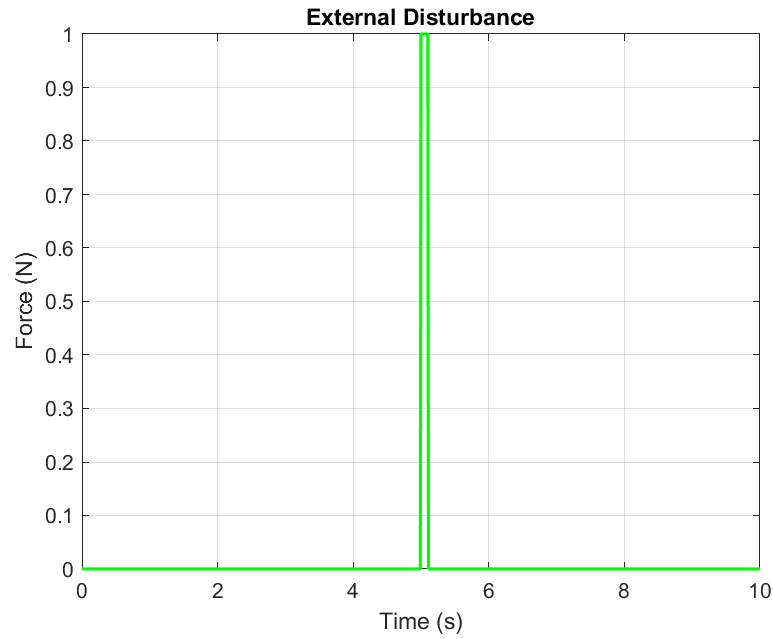


Figure IV.15 : Perturbation externe appliquée au robot

La Figure IV.11 illustre la perturbation externe appliquée (en Newtons). Cette force impulsive permet d'évaluer la capacité du contrôleur à rejeter les perturbances.

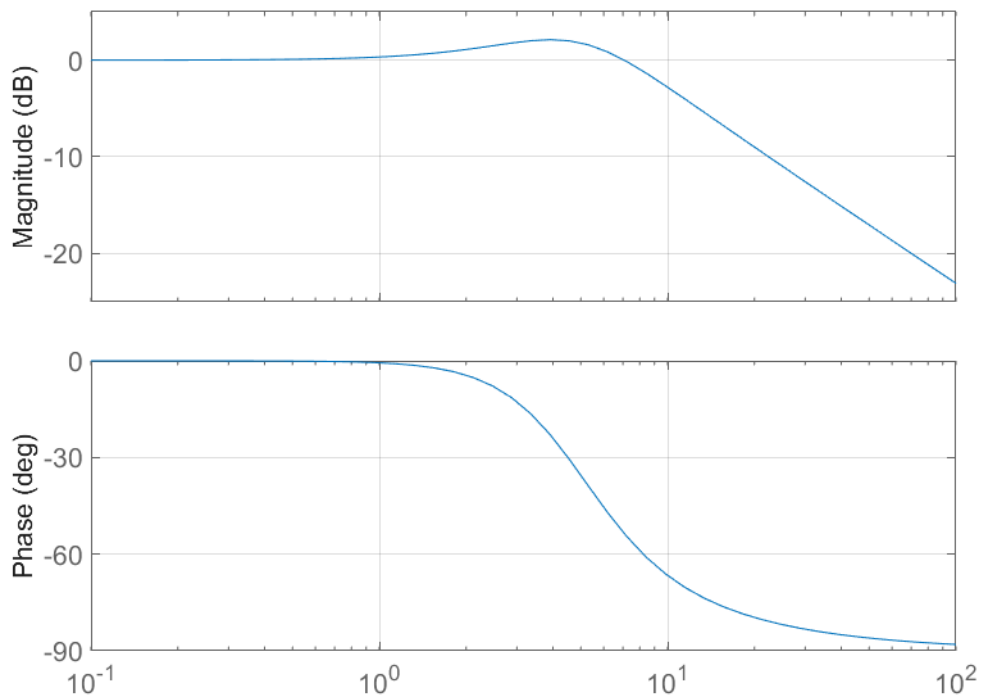


Figure IV.16: Diagramme de Bode du système en boucle fermée

La Figure IV.12 présente le diagramme de Bode du système en boucle fermée, révélant les marges de gain et de phase, ainsi que la bande passante du système.

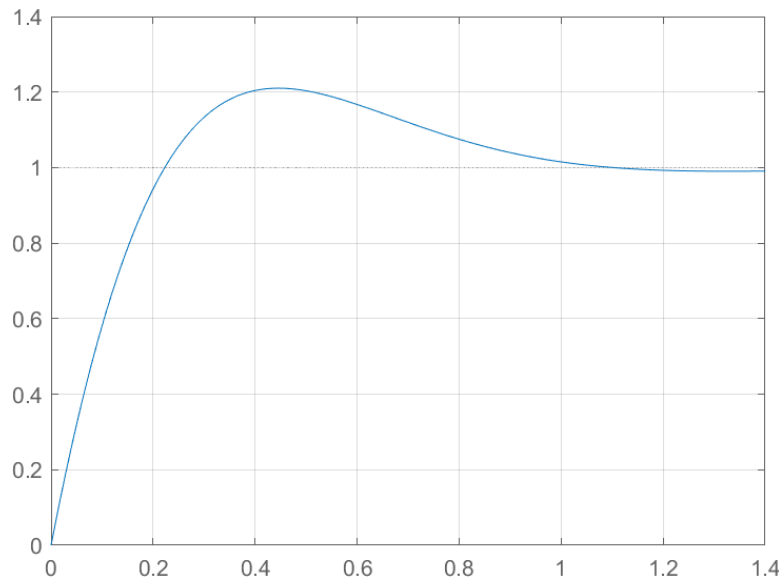


Figure IV.17 : Réponse à un échelon du système en boucle fermée

La Figure IV.13 montre la réponse à un échelon du système, mettant en évidence le temps de montée, l'amortissement et l'absence de dépassement significatif, conformément aux paramètres de conception ($\zeta = 0.7$, $\omega_n = 5$ rad/s).

4.1.1 Conclusion :

La modélisation mathématique est une base importante pour le développement de robots auto-stabilisants, mais son efficacité dépend fortement de la fidélité du système à la réalité physique. Une approche intégrée doit combiner :

- Modélisation riche : prendre en compte les effets mécaniques, électriques et environnementaux
- Identification précise : Collecter des données réelles pour corriger les hypothèses initiales
- Contrôle robuste : Capacité à assurer la stabilité dans un cadre incertain

Ces trois mesures sont cruciales pour garantir les performances et la fiabilité du robot dans des conditions réelles.

4.19 Construction de la simulation dans Simulink

Cette étude présente la modélisation et le contrôle d'un robot 3D auto-équilibré dans Simulink. Le processus commence par une conception mécanique dans un logiciel SOLIDWORKS, où chaque composant (supports moteur, plaques de montage, moteurs, roues et jantes) est exporté individuellement. Ces éléments ont ensuite été intégrés dans

l'environnement Simulink Multibody et assemblés à l'aide du bloc Rigid Transform pour garantir un positionnement précis.

Modélisez la dynamique du robot à l'aide d'articulations prismatiques (mouvement linéaire) et rotationnelles (rotation des roues). Appliquez une force le long de l'axe Z via le bloc STEP pour simuler des perturbations externes et convertissez-la en signal physique à l'aide d'un convertisseur PS. Cette approche permet d'analyser la réponse d'un système à des déséquilibres soudains.

Le système de contrôle est basé sur un régulateur PID dont les paramètres (gain proportionnel K_p , intégral K_i et différentiel K_d) sont optimisés à l'aide de l'outil Simulink PID Tuner. L'objectif est d'obtenir une réponse rapide avec un temps de stabilisation minimal tout en limitant les oscillations pour garantir la stabilité. De multiples itérations permettent d'identifier le meilleur compromis entre réactivité et robustesse, garantissant un retour à l'équilibre du fluide après perturbations.

4.19.1 Utilisation d'un contrôleur PID :

➤ **Modélisation 3D des Composants :**

Chaque partie du robot a été conçue individuellement en 3D pour garantir une représentation précise de la géométrie et des propriétés physiques. Les composants principaux incluent :

- **Châssis** : Composé de deux plaques , des supports de moteur.
- **Chariot** : Inclut les moteurs et les roues.

Chaque composant a été modélisé dans SOLIDWORKS, avec un soin particulier porté aux dimensions, aux matériaux, et aux centres de masse. Une fois les modèles terminés, ils ont été exportés au format STL, compatible avec Simulink.

➤ **Importation et assemblage dans Simulink :**

Les modèles 3D ont été importés dans Simulink à l'aide du bloc **File Solid** de la bibliothèque Simscape Multibody. Pour organiser le modèle, les composants ont été regroupés en sous-systèmes distincts :

- **Sous-système Châssis** : Regroupe les plaques, les supports de moteur.
- **Sous-système Chariot** : Inclut les moteurs et les roues.

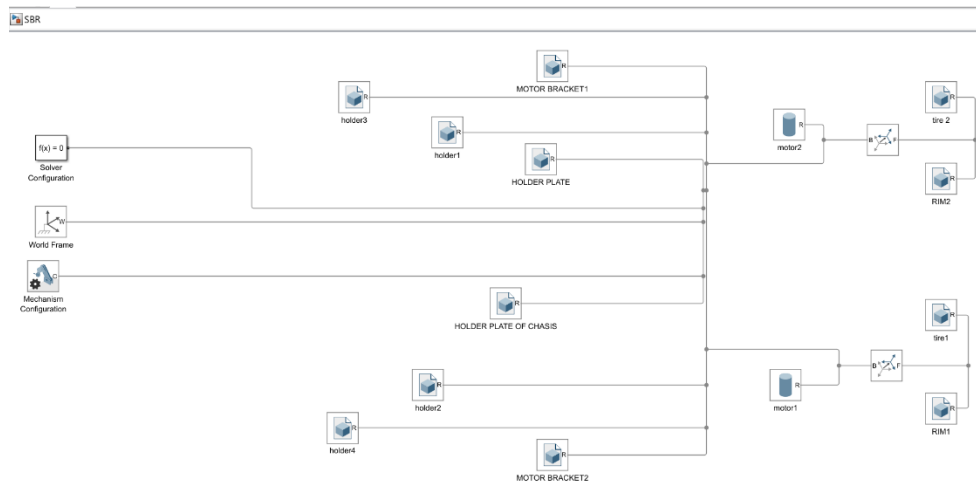


Figure IV.18 : Architecture générale du système dans simulink

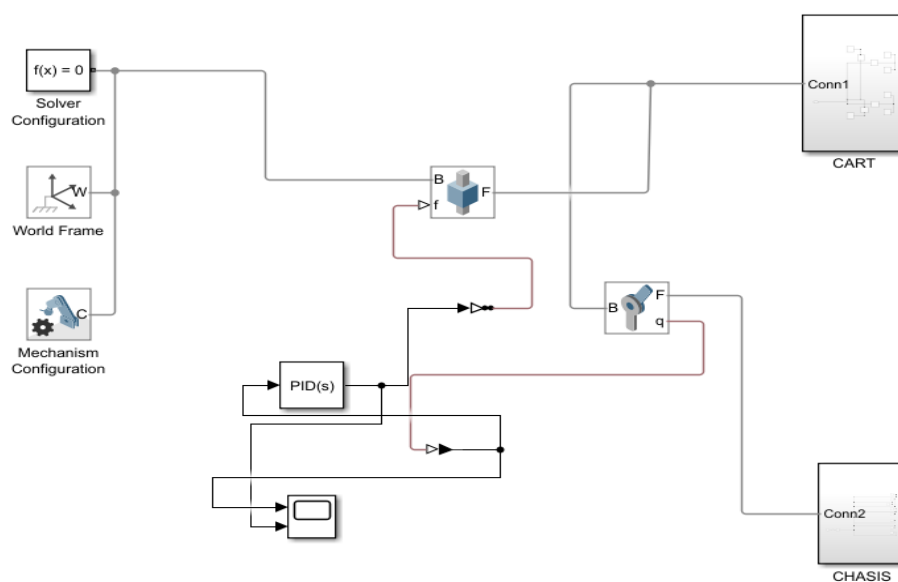


Figure IV.19 : Architecture générale du système dans simulink avec les sous-systèmes (châssis-chariot)

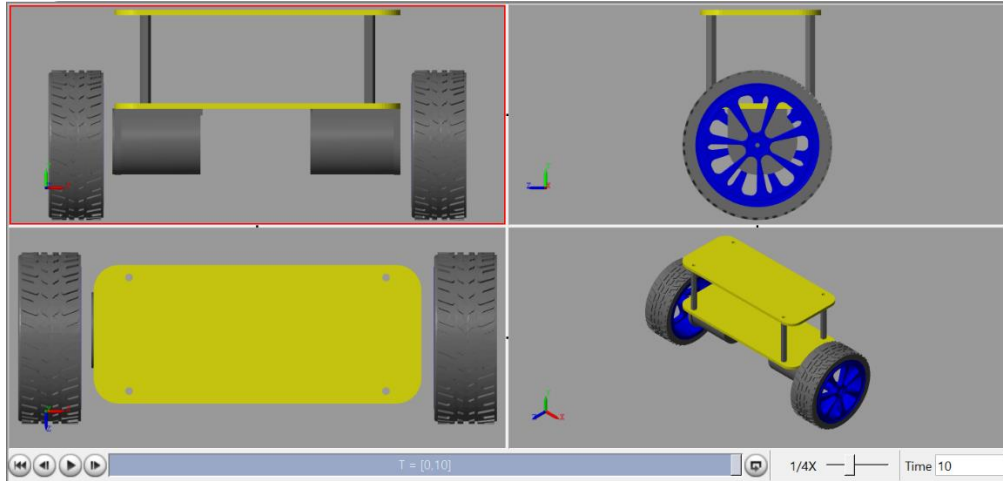


Figure IV.20 : le modèle 3D du robot dans la fenêtre du Simulation

Les composants du châssis et du chariot ont été connectés à l'aide de blocs **Rigid Transform** pour définir leurs positions et orientations relatives. Par exemple, le chariot a été soumis à une séquence de rotations (« y-x-z ») comprenant une rotation de 90 degrés autour de l'axe y et de 180 degrés autour de l'axe x, afin d'aligner correctement les composants par rapport au référentiel global.

➤ **Blocs Essentiels pour la Simulation :**

Trois blocs fondamentaux sont nécessaires pour configurer la simulation dans Simulink :

- **Solver Configuration** : Ce bloc définit la méthode d'intégration numérique et le pas de temps pour la simulation. Il garantit que les calculs dynamiques sont effectués avec précision.

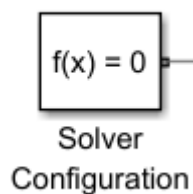


Figure IV.21 : Blocs Solver Configuration

- **World Frame** : Ce bloc représente le référentiel global (le sol) et sert de base pour définir les positions et orientations des composants mécaniques.

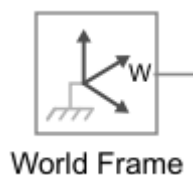
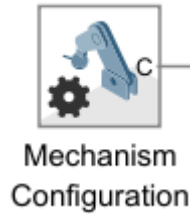


Figure IV.22: World Frame

- **Mechanism Configuration** : Ce bloc permet de spécifier les paramètres physiques du système, tels que la gravité uniforme ($9,81 \text{ m/s}^2$) appliquée à l'ensemble du mécanisme.

**Figure IV.23 :** Mechanism Configuration

Ces blocs sont placés à droite du modèle Simulink et connectés pour établir le cadre de la simulation.

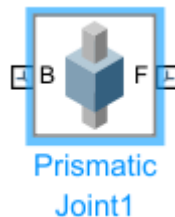
➤ **Connexion des Composants :**

Le chariot est connecté au référentiel global (**World Frame**) et au châssis à l'aide de blocs **Rigid Transform**. Les interactions dynamiques entre le châssis et le chariot sont modélisées à l'aide de joints :

- **Joint Rotatif (Revolute Joint)** : Permet au châssis de pivoter comme un pendule autour de l'axe des roues.

**Figure IV.24 :** Revolute Joint

- **Joint Prismatique (Prismatic Joint)** : Permet au chariot de se déplacer linéairement (avant et arrière).

**Figure IV.25 :** Prismatic Joint

➤ **Implémentation du contrôleur PID :**

Un contrôleur PID a été intégré pour stabiliser le robot en ajustant l'angle d'inclinaison (θ). Le système fonctionne comme suit :

- **Mesure de l'angle** : Un capteur (implémenté via un bloc Simscape) mesure l'angle d'inclinaison du châssis.
- **Comparaison** : L'angle mesuré est comparé à la consigne ($\theta_{\text{réf}} = 0$ pour l'équilibre vertical).
- **Génération de la force** : Le contrôleur PID calcule la force nécessaire pour stabiliser le robot en ajustant les moteurs.

Le contrôleur PID est implémenté à l'aide du bloc **PID Controller** de Simulink. Des blocs supplémentaires ont été ajoutés pour l'intégration et le suivi :

- **Blocs de conversion** : Pour convertir les signaux entre les domaines physique (Simscape) et de contrôle (Simulink).
- **Blocs Scope** : Pour visualiser l'évolution de l'angle et de la force au fil du temps.
- **Bloc Simulink-PS Converter** : Pour interfacer les signaux Simulink avec les composants physiques Simscape.

➤ **Spécification des masses des composants pour une modélisation précise :**

Afin de garantir une simulation précise dans Simulink, on a intégré les masses des différents composants du robot (châssis, moteurs, roues, jantes et supports) au modèle. Ces paramètres influencent directement la dynamique du système, notamment l'inertie et la répartition du centre de masse, essentielles pour des simulations réalistes. On a utilisé les valeurs suivantes:

- **Châssis** : Masse de [0,26 kg], incluant les plaques et les supports de moteur, mesurée à partir des spécifications des matériaux dans SOLIDWORKS.
- **Roues, moteurs et jantes** : Masse unitaire de [0,236 kg par roue], déterminée expérimentalement ou via les données du fabricant.
- **Autres composants** : Les éléments électroniques (carte de contrôle) ont une masse estimée à [0,1kg].

Lors de l'importation du modèle 3D, ces masses sont intégrées au module Simscape Multibody Solid File. De plus, le centre de masse de chaque composant est défini avec précision dans SOLIDWORKS et exporté au format STL afin de refléter la répartition réelle des masses. Cette approche améliore la précision du modèle dynamique, notamment lors de

la simulation des interactions châssis-cadre, et optimise le réglage du régulateur PID en tenant compte des effets inertiels réalistes.

➤ **Réglage du contrôleur PID :**

Le réglage des paramètres PID (gains proportionnel, intégral, et dérivatif) est crucial pour obtenir un comportement stable. Deux approches ont été utilisées :

- **Réglage manuel :** Les gains ont été ajustés progressivement pour minimiser les oscillations, réduire l'erreur en régime permanent, et assurer une réponse rapide. Par exemple, un gain proportionnel élevé améliore la réactivité, mais peut induire des oscillations si non équilibré par le gain dérivatif.

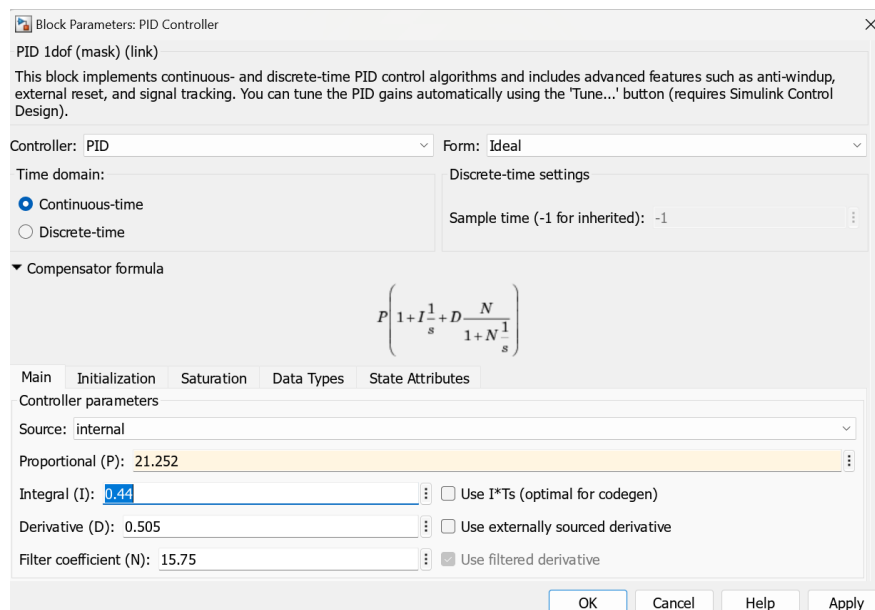


Figure IV.26 : Fenêtre du réglage manuel du PID

- **Réglage Automatique (Autotune) :** La fonction `piddtune` de MATLAB a été utilisée pour générer des gains optimaux en fonction des attributs de réponse spécifiés (par exemple, temps de montée, dépassement).

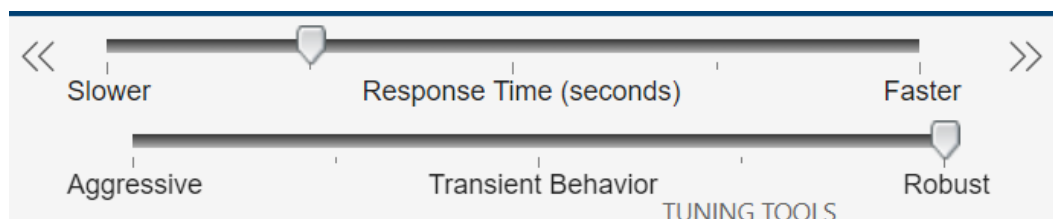


Figure IV.27 : Slider du PID Tuner

Le processus de réglage a été effectué dans Simulink en utilisant l'outil **PID Tuner**, qui permet de visualiser la réponse du système et d'ajuster les gains interactivement.

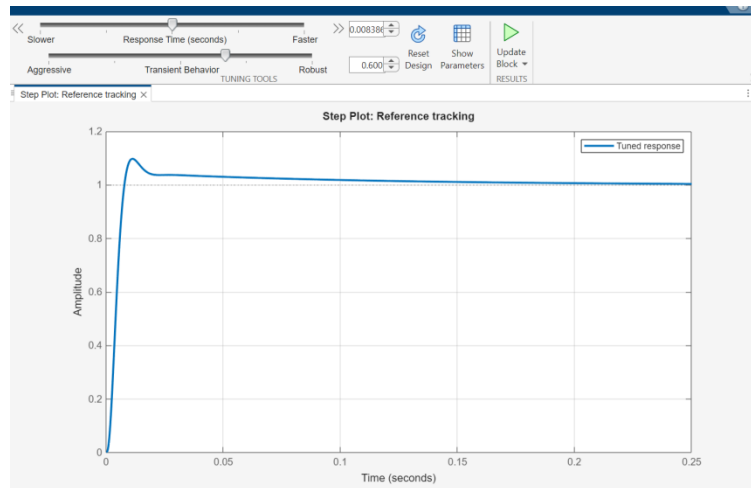


Figure IV.28 : Fenêtre de la réponse de système au PID Tuner

➤ **Résultats de la simulation :**

La simulation a été exécutée pour tester deux scénarios :

1. **Réponse à une consigne nulle** : Le robot est initialisé avec un angle d'inclinaison nul ($\theta_{\text{réf}} = 0$). Le contrôleur PID maintient l'équilibre, et la réponse est visualisée via un bloc **Scope**.
2. **Réponse à une perturbation** : Une perturbation (par exemple, un changement d'angle de 0,1 rad à $t = 2$ s) est appliquée pour tester la capacité du contrôleur à ramener le système à l'équilibre.

4.20 Régulateur linéaire quadratique (LQR) :

Le **LQR (Linear Quadratic Regulator)** est une méthode d'optimisation issue du contrôle optimal, utilisée pour stabiliser les systèmes dynamiques linéaires en minimisant une fonction de coût quadratique. Il est particulièrement bien adapté aux systèmes instables comme un robot auto-équilibrant, où l'objectif est de maintenir la verticalité tout en minimisant les efforts de commande.

➤ **Définition mathématique :**

Un système linéaire d'état peut s'écrire sous la forme :

$$\dot{x}(t) = Ax(t) + Bu(t)$$

où:

- $x(t) \in \mathbb{R}^n$ est le vecteur d'état (angle, vitesse angulaire, position, etc.),

- $u(t) \in R^m$ est le vecteur de commande,
- $A \in R^{n \times n}$ est la matrice d'état,
- $B \in R^{n \times m}$ est la matrice de commande.

Le LQR cherche à minimiser la fonction de coût suivante :

$$J = \int_0^{\infty} (x(t)^T Q x(t) + u(t)^T R u(t)) dt$$

où :

- $Q \geq 0$ est une matrice symétrique pondérant les états (importance de la stabilité),
- $R > 0$ est une matrice symétrique pondérant les efforts de commande.

La commande optimale est donnée par :

$$u(t) = -Kx(t)$$

avec K calculée à partir de la solution P de l'équation de Riccati algébrique :

$$A^T P + P A - P B R^{-1} B^T P + Q = 0$$

et :

$$K = R^{-1} B^T P$$

Ces équations proviennent de la théorie du contrôle optimal présentée dans [34] et [35]

4.20.1 Application à un robot auto-équilibrant

Dans le cas du **robot auto-équilibrant**, le LQR est utilisé pour générer une commande de tension (PWM) afin de maintenir l'équilibre autour de la position verticale.

➤ LQR sur MATLAB :

Le code MATLAB implémente un contrôleur LQR (Linear Quadratic Regulator) pour un robot auto-équilibré :

➤ Définition des paramètres physiques :

Les paramètres du robot sont définis, incluant la masse du corps ,la masse des roues,la distance du centre de masse au centre des roues , l'accélération gravitationnelle ,le moment d'inertie du corps ,et le rayon des roues.

➤ Modélisation du système :

Une représentation en espace d'état linéarisée est créée pour le robot. La matrice d'état A décrit la dynamique du système (angle d'inclinaison, vitesse angulaire, position horizontale, vitesse linéaire). La matrice d'entrée B relie l'effort de contrôle (couple appliqué aux roues) aux états. La matrice de sortie C indique que l'angle et la position sont mesurés, et D est nulle car il n'y a pas d'effet direct de l'entrée sur la sortie.

➤ Conception du contrôleur LQR :

Les matrices de pondération Q et R sont définies pour équilibrer l'importance des états ($Q = \text{diag}([10, 1, 1, 1])$ met l'accent sur le contrôle de l'angle) et l'effort de contrôle ($R = 0.1$). La fonction `lqr` calcule la matrice de gain K qui minimise un critère de performance quadratique. La matrice de gain K possède quatre entrées, correspondant aux quatre variables d'état : l'angle d'inclinaison (θ), la vitesse angulaire ($d\theta/dt$), la position horizontale (x), et la vitesse linéaire (dx/dt). Ces gains sont utilisés pour calculer l'effort de contrôle $u = -K*x$, permettant de stabiliser le robot.

➤ Simulation du système :

Le système en boucle fermée est créé avec $A - B*K$ pour intégrer le contrôleur. Une simulation est effectuée sur 5 secondes avec une condition initiale d'un angle de 0.1 radian, sans entrée externe ($u = 0$). La fonction `lsim` calcule la réponse du système (angle et position).

4.21 Visualisation des résultats :

Deux graphiques sont tracés : l'un montre l'évolution de l'angle d'inclinaison en fonction du temps, l'autre montre la position horizontale des roues. Ces graphiques permettent de vérifier si le robot se stabilise (angle proche de 0) et si la position reste contrôlée.

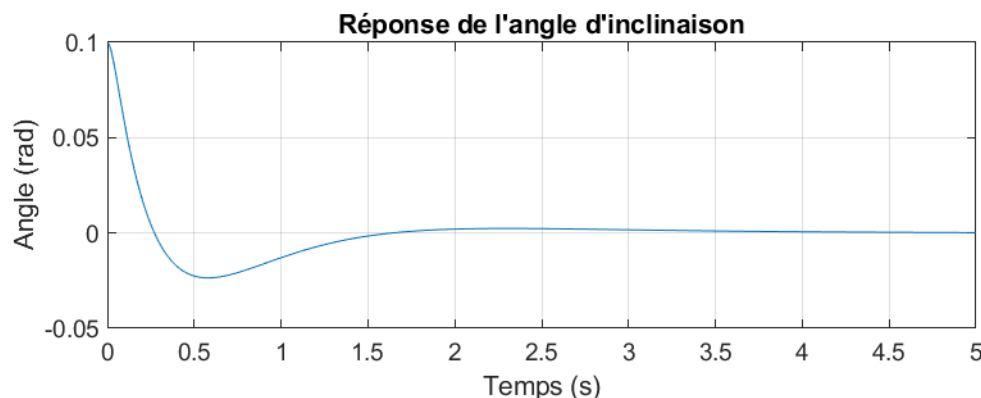


Figure IV.29 : Fenêtre de la réponse de l'angle au LQR.

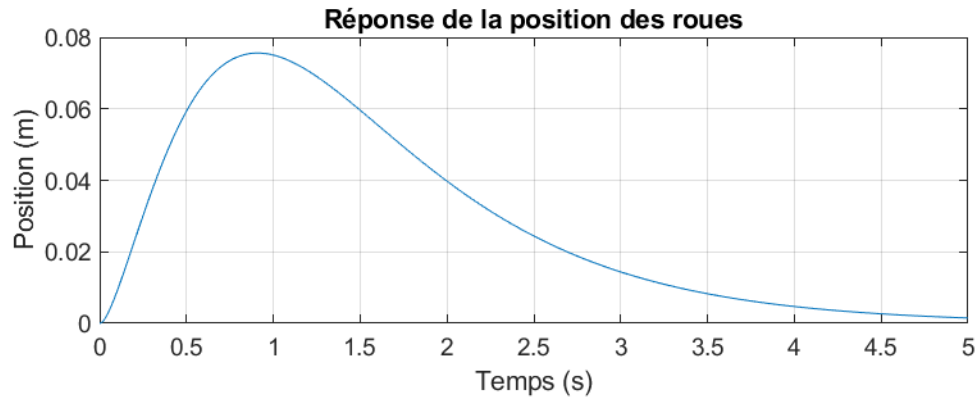


Figure IV.30 : Fenêtre de la réponse de la position au LQR.

Ce code permet donc de concevoir un contrôleur pour stabiliser un robot auto-équilibré et de simuler son comportement pour évaluer les performances du contrôleur. Les gains calculés (K) sont utilisés pour générer un effort de contrôle qui maintient le robot à la verticale tout en limitant les mouvements horizontaux.

4.22 Comparaison entre PID et LQR :

Critère	PID classique	LQR (Linear Quadratic Regulator)
Complexité de conception	Faible (3 gains à ajuster)	Moyenne à élevée (matrices à définir)
Modèle requis	Aucun modèle nécessaire	Modèle dynamique linéaire nécessaire
Performance	Moyenne (dépend du tuning manuel)	Élevée (optimisation mathématique)
Temps de calcul	Très faible	Faible (le gain K est constant une fois calculé)
Robustesse	Moyenne	Bonne (pondération des états)
Implémentation sur ESP32	Très facile (intégration directe)	Facile si K est pré-calculé
Adaptabilité multivariable	Faible	Excellente (systèmes MIMO)

Tableau IV .3 : Comparaison entre PID et LQR.

4.22.1 Facilité d'implémentation sur ESP32 :

Pour un **ESP32**, le **PID** est **plus simple à implémenter**.

Cependant, si le **gain K du LQR** est calculé à l'avance sous **MATLAB**, il est également **facile à intégrer** dans le programme Arduino sous forme de multiplication matricielle. Ainsi, **LQR devient presque aussi simple que PID, à condition de calculer les gains hors ligne**.

4.23 Conclusion :

La simulation dans Simulink et Matlab, combinée à des modèles 3D précis et à un contrôleur PID, a permis de valider le comportement dynamique du robot auto-équilibré. La modélisation 3D détaillée a assuré une représentation fidèle des propriétés mécaniques, tandis que le réglage du PID et LQR a garanti la stabilité et la robustesse du système face à des perturbations.

Chapitre V

Tests et Résultats

Ce chapitre aborde la phase d'optimisation du système robotique, en se concentrant sur l'étalonnage du capteur MPU6050 et l'optimisation des paramètres de contrôle. Pour obtenir des performances optimales, diverses méthodes ont été utilisées. Ce chapitre aborde également le processus de test du robot auto-équilibré, évaluant ses performances et sa stabilité sur différentes surfaces.

Note : amélioration de la stabilité du robot auto-équilibré

Pour développer le robot auto-équilibré, On a utilisé un microcontrôleur ESP32 et l'avons programmé en C++ avec l'IDE Arduino afin de tirer parti de sa vitesse d'exécution, de sa compatibilité avec des bibliothèques telles que `Wire.h`, `MPU6050.h` et `BluetoothSerial.h`, et de sa capacité à gérer des tâches complexes (communication Bluetooth/Wi-Fi, régulation PID, traitement des capteurs). Comparé à MicroPython, C++ offre une meilleure efficacité mémoire et un contrôle périphérique plus précis, malgré une syntaxe plus complexe. La plateforme prend en charge l'intégration robuste des algorithmes électroniques et de contrôle décrit au chapitre 3.

L'ESP32 gère le capteur MPU6050 pour la mesure de l'inclinaison, le capteur à ultrasons HC-SR04 pour la détection d'obstacles, le module GPS GY-NEO6MV2 pour le positionnement et le pont en H TB6612FNG pour le contrôle des moteurs CC avec encodeurs. La batterie 18650, associée au régulateur LM2596 et aux condensateurs (0,1 μ F céramique, 10 μ F électrolytique et 100 μ F), assure une alimentation stable. Le réseau neuronal déployé sur l'ESP32 via TensorFlow Lite permet une navigation autonome basée sur les données du HC-SR04. Ces choix améliorent considérablement la stabilité et l'adaptabilité du robot, lui permettant de maintenir son équilibre face aux perturbations extérieures.

5.1. Étalonnage

5.1.1. MPU6050 :

Pour étalonner le capteur MPU6050 dans notre projet de robot auto-équilibré, on a suivi ces étapes, en utilisant la bibliothèque `MPU6050.h` et la communication I2C (broches SCL sur D22, SDA sur D21) :

1. Instancier l'objet MPU6050 et configurer la communication I2C pour établir une connexion avec le capteur (adresse 0x68 ou adresse 0x69 si AD0 est connecté).
2. Initialiser les variables pour le filtre de kalman et le processus d'étalonnage.
3. Créer une boucle principale pour lire en continu les données du capteur via `Wire.h`.

4. Lire les valeurs brutes de l'accéléromètre (axes X, Y, Z) et du gyroscope (vitesse angulaire).
5. Calculer l'angle d'inclinaison à partir de l'accéléromètre à l'aide de la trigonométrie et convertir les données du gyroscope en degrés par seconde.
6. Appliquer un filtre complémentaire (fonction `Yiorderfilter`) pour combiner les données, réduisant le bruit et la dérive.
7. Cumuler les angles dans une variable de somme et incrémenter un compteur pour suivre le nombre d'itérations.
8. Après 180 secondes, calculer la moyenne des angles pour obtenir les valeurs étalonnées, stockées pour une utilisation dans les boucles PID.

5.1.2. Test du filtre de Kalman pour la sélection des axes :

Pour valider l'étalonnage et sélectionner les axes appropriés, un filtre de Kalman a été testé en complément du filtre complémentaire. Les données brutes des axes X (ax) et Y (ay) de l'accéléromètre, ainsi que du gyroscope X (gx), ont été comparées aux angles filtrés (pitch pour l'équilibre avant/arrière). Le filtre de Kalman, avec les paramètres $Q_angle=0.001$, $Q_gyro=0.003$, $R_angle=0.5$, et $dt=0.005$, a réduit le bruit, confirmant que l'axe X (ax, gx) est optimal pour l'équilibre. Les tests de basculement ont montré une meilleure précision sur l'axe X, avec un écart maximal de 0.5° .

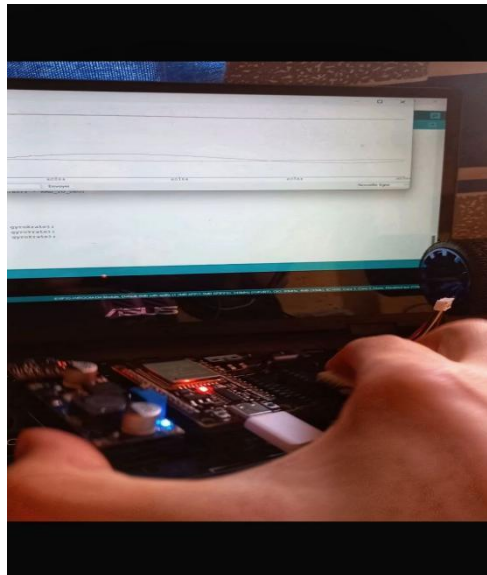


Figure 5.1 : Test du filtre de Kalman pour la sélection des axes

5.2. Optimisation et Tests

5.2.1. Optimisation :

Les paramètres PID (k_p , k_i , k_d pour l'angle, la vitesse et la direction) n'ont pas été initialement définis via la bibliothèque Arduino, mais ajustés manuellement via une interface web hébergée par l'ESP32 en mode point d'accès (adresse IP 192.168.4.1). Le robot, alimenté par trois batteries 18650 (11,4 V-12,6 V) et contrôlé par un LM2596, a été testé sur une surface plane. Il a d'abord maintenu l'équilibre quelques secondes avant de dévier, provoquant une chute.

L'optimisation consiste à ajuster les gains PID via l'application afin de minimiser les oscillations et d'améliorer le temps de réponse. La boucle PID d'angle (paramètres k_p , k_i , k_d) utilise les données d'étalonnage du MPU6050 pour maintenir l'équilibre, tandis que les boucles de vitesse (k_{p_speed} , k_{i_speed} , k_{d_speed}) et de direction (k_{p_turn} , k_{i_turn} , k_{d_turn}) contrôlent les moteurs via le TB6612FNG. La communication Bluetooth est implémentée par BluetoothSerial.h, permettant la transmission en temps réel des réglages via l'application mobile développée en Java sur Android Studio. Ces réglages assurent un équilibre stable et une navigation fluide, même en présence de perturbations.



Figure 5.2 : Interface d'application l'ajustement des paramètres PID

5.2.2. Tests :

Le robot a été testé sur différentes surfaces (carrelage, moquette, gazon, pavé) afin d'évaluer sa stabilité en conditions réelles. L'encodeur du moteur CC fournit des données précises pour réguler la vitesse via une boucle PID. Sur moquette, le frottement accru des roues améliore la stabilité, confirmant l'importance de l'adhérence, en accord avec la philosophie de conception de Simulink. Le capteur HC-SR04 détecte les obstacles dans un rayon de 15 cm, émet un signal sonore (broche D23) et arrête le robot pour éviter une collision. Le module GPS GY-NEO6MV2 permet une localisation précise, affichée sur l'application mobile via Google Maps.

Le réseau neuronal, entraîné sur Google Colab et déployé sur un ESP32, a permis une navigation autonome et a pu prédire les actions (arrêt, marche arrière, virage à gauche, virage à droite) avec une précision de 100 % après ajustement de la plage de distance (0-10 cm, 10-20 cm, 20-40 cm). Cependant, sur des surfaces très irrégulières comme l'herbe, le couple moteur était parfois insuffisant, ce qui suggère que les applications futures pourraient nécessiter des moteurs plus puissants.

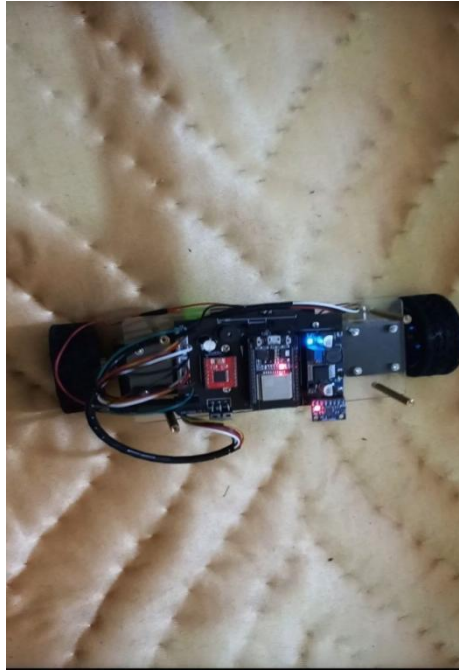


Figure 5.3 : Robot auto-équilibré sur moquette

5.3. Conclusion :

Le processus d'étalonnage du MPU6050, complété par des tests de filtre de Kalman (pour la sélection des axes) et l'optimisation des paramètres PID, est essentiel pour garantir la fiabilité du robot auto-équilibré. L'intégration des capteurs ESP32, TB6612FNG, HC-SR04, GPS GY-NEO6MV2 et des condensateurs (0,1 μ F, 10 μ F, 100 μ F) permet une gestion robuste des capteurs, des moteurs et des communications. Les tests ont démontré une stabilité améliorée sur diverses surfaces, validant l'efficacité de la conception mécanique (SolidWorks), du circuit imprimé (EasyEDA) et de l'application mobile. La navigation autonome, prise en charge par les réseaux neuronaux, améliore les capacités du robot, malgré la nécessité d'un couple moteur plus élevé sur les routes complexes.

Conclusion Générale

Conclusion Générale

Ce mémoire présente une étude approfondie de la conception, de la construction et du contrôle d'un robot auto-équilibré à deux roues, basée sur une approche multidisciplinaire impliquant la mécanique, l'électronique, l'informatique et l'intelligence artificielle. L'objectif principal de la thèse est de résoudre le problème du pendule inversé, grâce à une méthodologie rigoureuse, de la modélisation théorique à la vérification expérimentale.

Ce projet a démontré la faisabilité d'un robot auto-équilibré capable de rester stable dans diverses conditions dynamiques tout en intégrant des fonctionnalités avancées telles que la détection d'obstacles et le positionnement GPS. La modélisation mathématique détaillée au chapitre 2 fournit une base solide pour comprendre la dynamique complexe du système en linéarisant les équations non linéaires autour du point d'équilibre instable. Cette approche permet de concevoir des contrôleurs robustes, notamment des régulateurs PID et des filtres de Kalman, essentiels pour gérer les incertitudes des capteurs et les perturbations externes. L'innovation réside également dans l'intégration de l'intelligence artificielle (voir chapitre 3 pour plus de détails), qui permet au robot de prendre des décisions en temps réel pour éviter les obstacles. Le réseau neuronal entraîné et déployé sur ESP32 améliore non seulement la robustesse du système, mais ouvre également la voie à des applications dans des environnements complexes. La conception mécanique est réalisée sous SolidWorks et la fabrication du circuit imprimé sous EasyEDA, ce qui garantit une intégration optimale des composants et améliore la modularité et la répétabilité du prototype.

Les tests expérimentaux présentés au chapitre 5 ont validé les choix techniques du robot sur différents revêtements routiers, démontrant sa capacité d'adaptation aux environnements réels. Ces tests ont également mis en évidence des défis pratiques, tels que l'optimisation des gains PID pour minimiser les oscillations et la gestion des irrégularités du terrain, qui ont été résolus par un réglage itératif. Ces résultats confirment le potentiel du robot pour des applications pédagogiques (comme outil d'apprentissage des principes de contrôle et de robotique) ainsi que pour des utilisations industrielles (par exemple, la logistique ou la surveillance autonome).

À l'avenir, plusieurs pistes d'amélioration méritent d'être explorées. L'intégration de capteurs supplémentaires, tels que le LiDAR, pour une cartographie plus précise, pourrait améliorer l'autonomie du robot. De plus, l'utilisation de techniques de contrôle avancées, comme le contrôle prédictif par modèle (MPC), pourrait améliorer la gestion de la dynamique non linéaire à grande vitesse. Enfin, l'optimisation énergétique grâce à des algorithmes de gestion

de batterie ou à des matériaux plus légers pourrait accroître l'autonomie du système, le rendant ainsi plus adapté aux applications à long terme.

En résumé, cette étude démontre comment réaliser des systèmes robotiques performants grâce à une approche intégrée combinant théorie, simulation et expérimentation. Le robot auto-équilibré développé dans ce travail résout non seulement un problème académique classique, mais pose également les bases d'innovations futures dans les domaines de la robotique mobile et de l'intelligence artificielle appliquée, et pourrait avoir un impact sur tous les aspects de la société.

Références

Références

- [1] Spong, M. W. (2020). Robot Modeling and Control. Wiley.
- [2] Siciliano, B. (2009). Robotics: Modelling, Planning and Control. Springer.
- [3] Sawatzky, B. et al. (2007). The Segway as a Mobility Device. Arch Phys Med Rehabil.
- [5] Leobotics (2022). Segway Loomo Specifications. Document Technique.
- [6] Segway Robotics (2024). ServeBot S1: Technical Manual.
- [7] Piaggio Fast Forward (2023). Gita Robot Launch Video. YouTube.
- [8] Heater, S. (2017). Gita: Personal Cargo Robot. TechCrunch.
- [9] Piaggio Fast Forward (2019). Gita SLAM Technology. Technical White Paper.
- [10] Porter, J. (2019). Gita Commercial Release. The Verge.
- [11] Boston Dynamics (2017). Handle Robot: Technical Overview. YouTube.
- [12] Alsaadi, A. et al. (2023). Mobile Manipulators in Industry 4.0. Sensors.
- [13] Robotnik (2024). AMR for Industrial Automation. Product Documentation.
- [14] McKinsey & Company (2019). Automation in Logistics. Industry Report.
- [15] Running Brains (2024). GR100 Surveillance Robot. Technical Datasheet.
- [16] TBC-France (2023). Jack Surveillance Robot Specifications.
- [17] France 2030 (2024). NOVA 2027 Project. Ministère de la Recherche.
- [18] Park, J. & Cho, B. (2013). STEM Robotics with Arduino. IEEE Conference Paper.
- [19] Li, X. et al. (2019). Lower Limb Rehabilitation Exoskeletons. Chinese Journal of Mechanical Engineering.
- [20] HARMONY Project (2024). Medical Logistics Robots. EU Horizon 2020 Report.
- [21] University of Sydney (2018). Digital Farmhand for Food Security. Research Bulletin.
- [22] F. L. Lewis and D. L. Vrabie, Feedback Control of Dynamic Systems (Pearson Education, 2012).
- [23] T. Bräunl, Sensors and Actuators, available at:
<https://cseweb.ucsd.edu/classes/wi17/cse237A-a/handouts/Braunl.Sensors.Actuators>, p. 42.
- [24] The University of Western Australia, School of Mechanical Engineering, Balancing a Two-Wheeled Autonomous Robot, Final Year Thesis (2003), pp. 23–25
- [25] IEEE Transactions on Robotics, "Aerodynamic Effects in Micro Robotics," 34(2), 215–230 (2018).
- [26] Robotics: Science and Systems (RSS) Conference Proceedings, "Nonlinear Wheel-Ground Interaction" (MIT Press, 2020).
- [27] ASME Journal of Dynamic Systems, Measurement, and Control, "Structural Flexibility in Mobile Robots," 143(5), 051004 (2021).
- [28] IEEE/ASME Transactions on Mechatronics, "DC Motor Modeling for Control," 24(3), 1021–1032 (2019).
- [29] IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), "Power Supply Effects on Control Stability," pp. 5507–5514 (2021).
- [30] Nature Machine Intelligence, "Grey-Box Modeling," 7(4), 332–341 (2022), doi:10.1038/s42256-022-00505-4.
- [31] L. Ljung, *System Identification: Theory for the User* (2nd ed., Prentice-Hall, 1999).

- [32] S. Skogestad and I. Postlethwaite, *Multivariable Feedback Control* (2nd ed., Wiley, 2005).
- [33] IEEE International Conference on Robotics and Automation (ICRA) Workshop, "Benchmarking Methods for Legged Robots" (IEEE Press, 2023).
- [34] L. B. Lau and N. S. Ahmad, Self-balancing robot: modeling and comparative analysis between PID and linear quadratic regulator, *International Journal of Reconfigurable and Embedded Systems* 12, 351–359 (2023)
- [35] Anonymous, Modeling, Simulation, and Optimal Control for Two-Wheeled Self-Balancing Robot (ResearchGate, 2024).

Annexes

Annexes :

ESP32 :

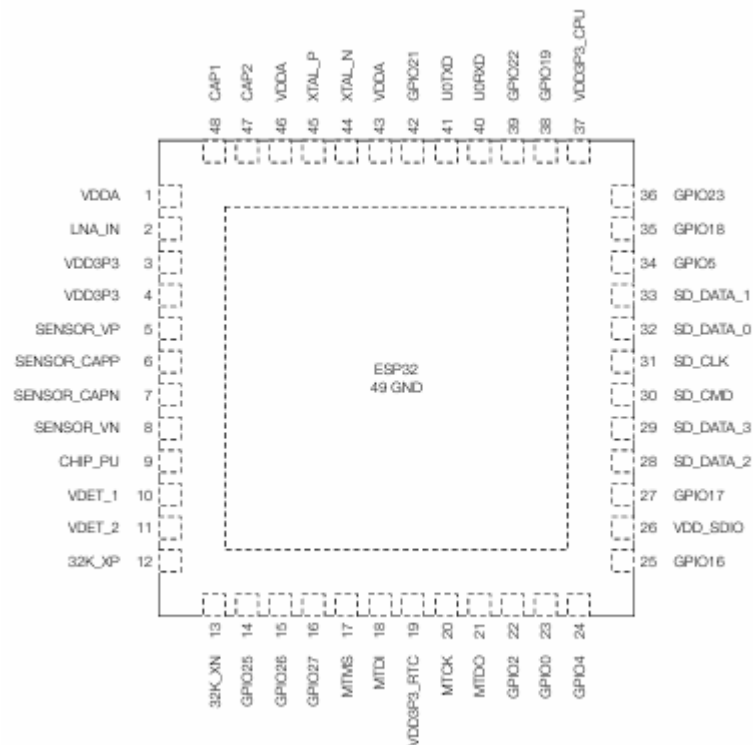
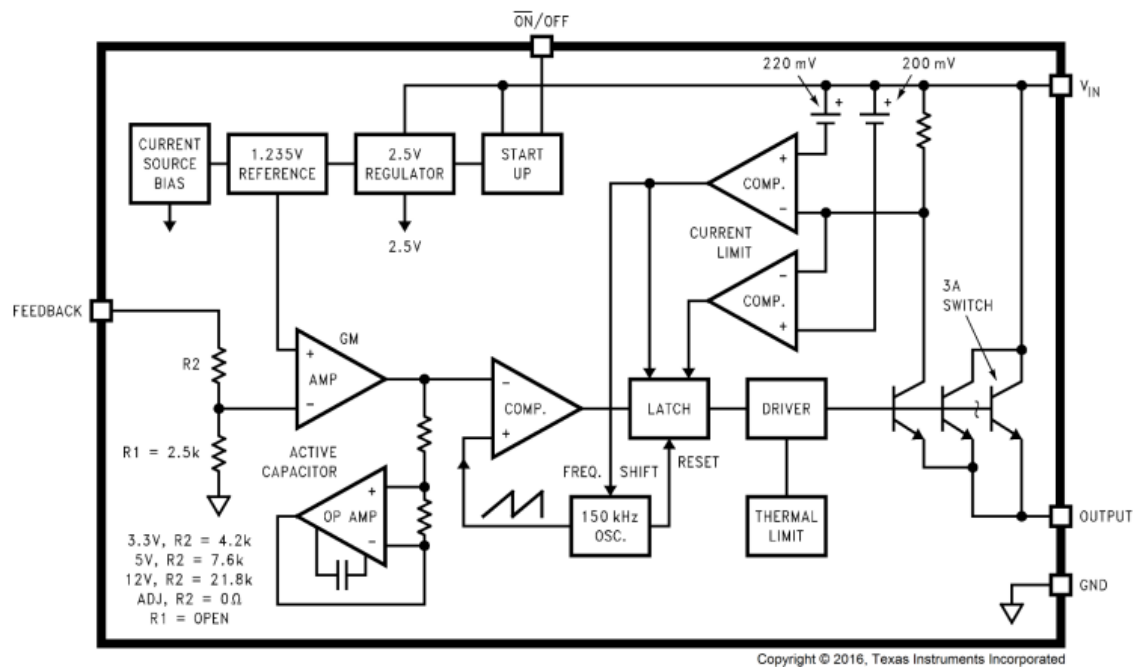
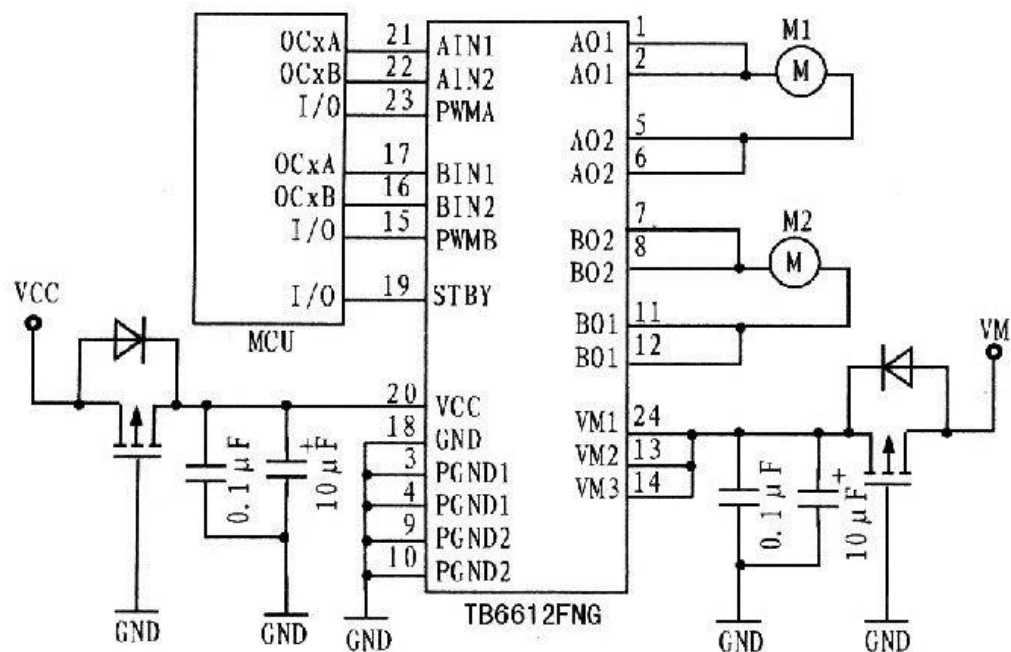


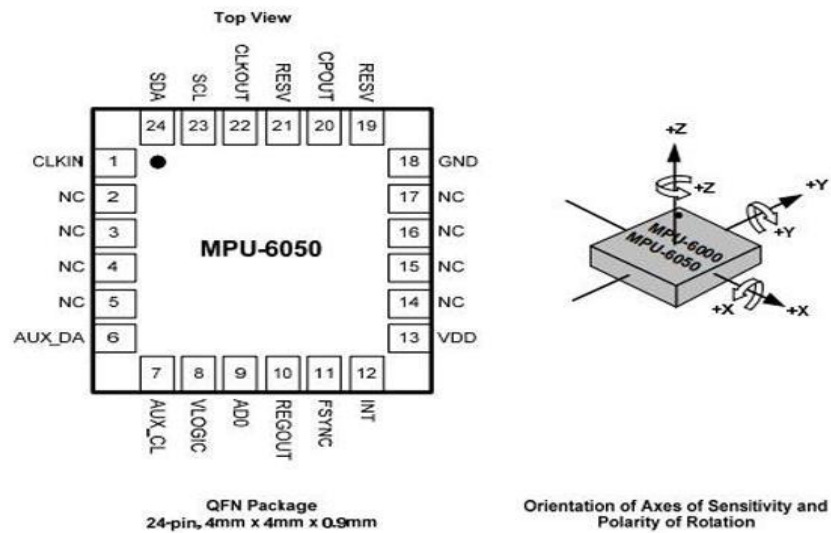
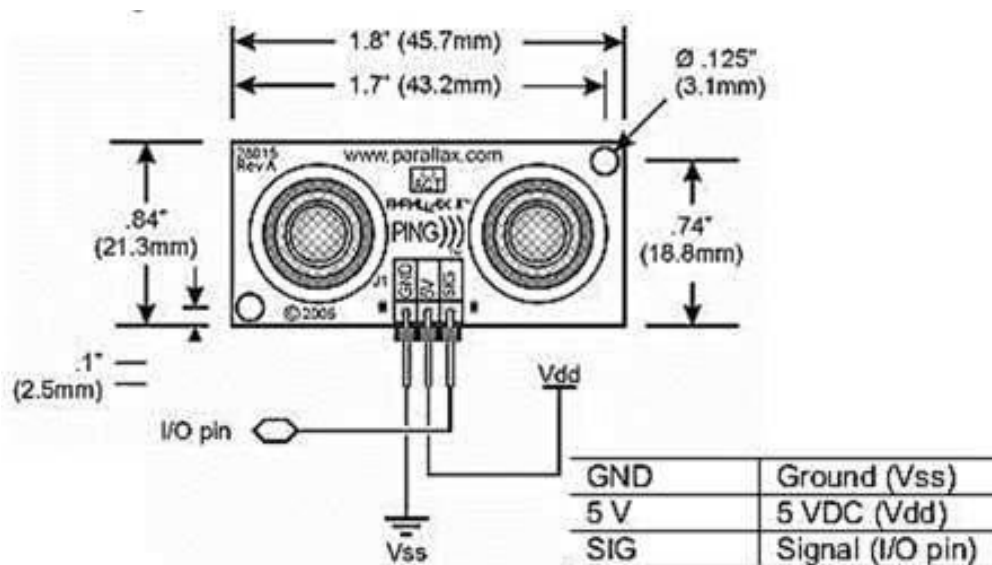
Figure a.1 : Broche de puce ESP32

GPS NEO6M :



Figure a.2 : Broche de puce NEO-6

LM2596 :**Figure a.3 :** Schéma du circuit intégré Régulateur LM2596**TB6612FNG :****Figure a.4 :** Schéma du circuit intégré TB6612FNG

MPU6050 :**Figure a.5 :** Schéma du circuit intégré MPU6050**Capteur HC-SR04 :****Figure a.6 :** Structure de HC-SR04

